Σχεδίαση και Ανάλυση Αλγορίθμων

Ενότητα 10.0

Μετασχηματισμοί Υπολογιστικών Προβλημάτων

Αναγωγές

NP-πληρότητα

Σταύρος Δ. Νικολόπουλος | 2016-17
Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Ιωαννίνων
Webpage: www.cs.uoi.gr/~stavros

Στο Βασίλειο των Ζώων

=

≠

Δαρβίνος

Κοινωνικά ζώα
Φροντίζουν τα μικρά τους
Έχουν τρίχωμα

Πράγματα φαινομενικά εντελώς διαφορετικά, συχνά **Αποδεικνύεται ότι είναι ΙΔΙΑ**, πέρα από επιφανειακές διαφορές !!!

- **Στο Βασίλειο των Προβλημάτων**

Μέγιστη Αύξουσα Υπακολουθία

Μέγιστη Διαδρομή σε DAG

Είδαμε με ποιο τρόπο ένας Αλγόριθμος για τον υπολογισμό της
Μέγιστης Διαδρομής σε DAG
μπορεί, με εκπληκτικό τρόπο, να χρησιμοποιηθεί για την εύρεση μιας
Μέγιστης Αύξουσας Υπακολουθίας

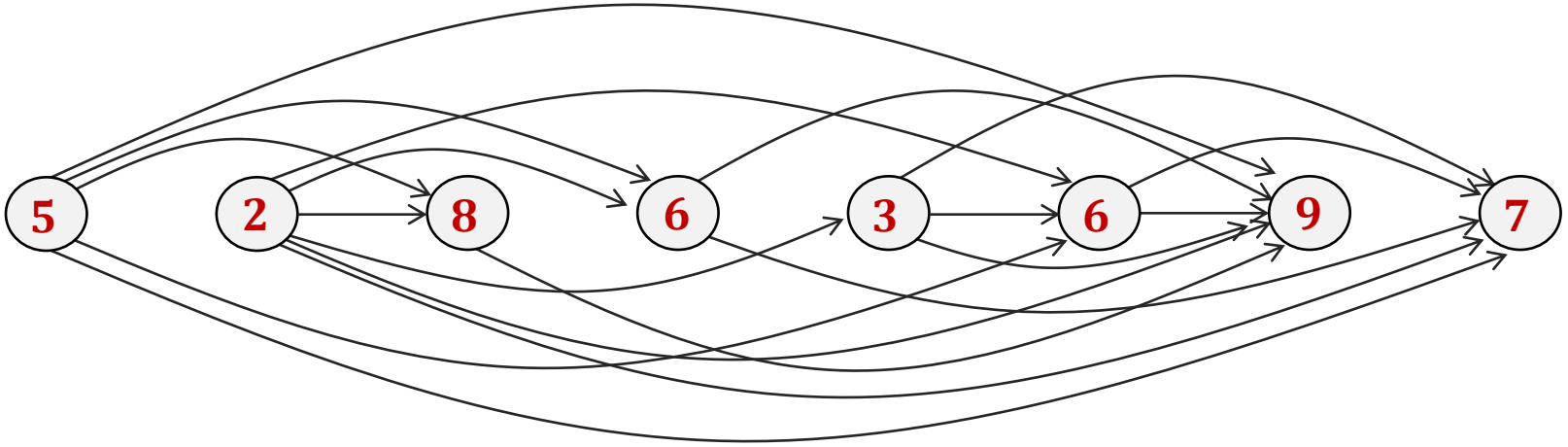
- **Στο Βασίλειο των Προβλημάτων**

Μέγιστη Αύξουσα Υπακολουθία ≤ Μέγιστη Διαδρομή σε DAG

Το Πρόβλημα της Μέγιστης Αύξουσας Υπακολουθίας **ανάγεται** **στο Πρόβλημα** της Μέγιστης Διαδρομής σε DAG

- **ΜΕΓΙΣΤΗ-ΑΥΞΟΥΣΑ-ΥΠΑΚΟΛΟΥΘΙΑ ≤ ΜΕΓΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG**

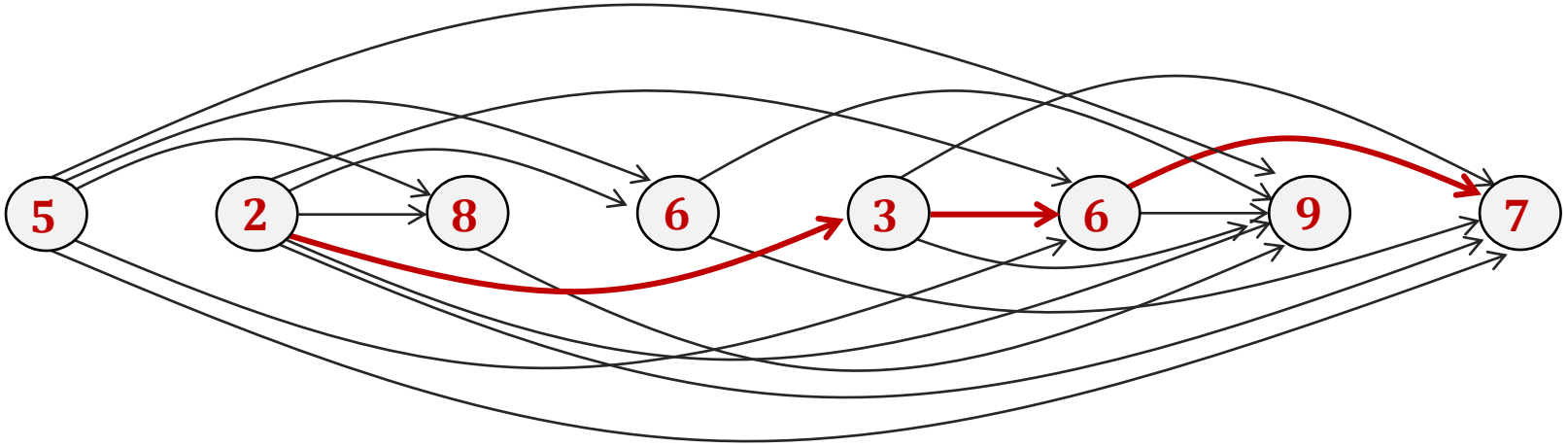
A = (5, 2, 8, 6, 3, 6, 9, 7)



Δημιουργούμε ένα **G** με **ΟΛΕΣ** τις **επιτρεπτές μεταβάσεις** !!!

- **ΜΕΓΙΣΤΗ-ΑΥΞΟΥΣΑ-ΥΠΑΚΟΛΟΥΘΙΑ ≤ ΜΕΓΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG**

A = (5, 2, 8, 6, 3, 6, 9, 7)



Λύση: (2, 3, 6, 7)

Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές του DAG** !!!

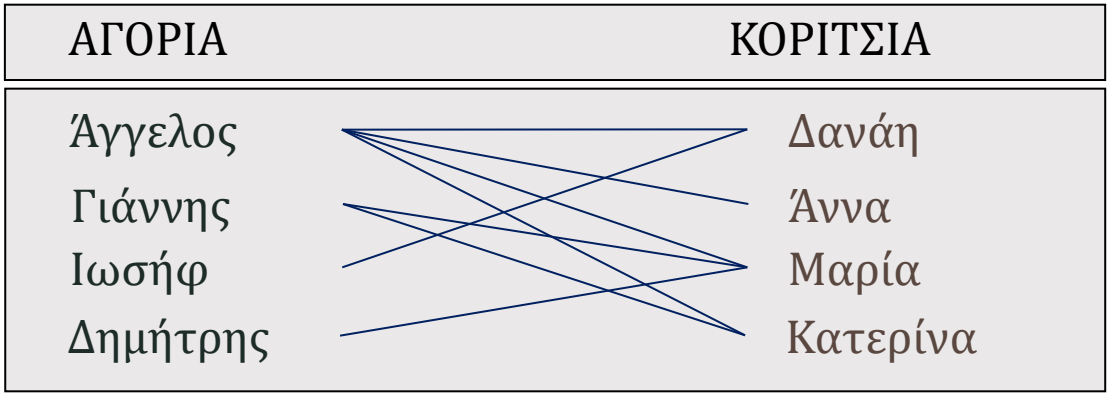
- **Στο Βασίλειο των Προβλημάτων**

Τέλεια Αντιστοίχιση ≤ Μέγιστη Ροή

Πρόσφατα είδαμε ότι:

Μπορούμε να **«ανάγουμε»** το πρόβλημα της Τέλειας Αντιστοίχισης σε πρόβλημα Μεγίστης Ροής !!!

Τελεια-Αντιστοιχιση ≤ Μεγιστη-Ροη



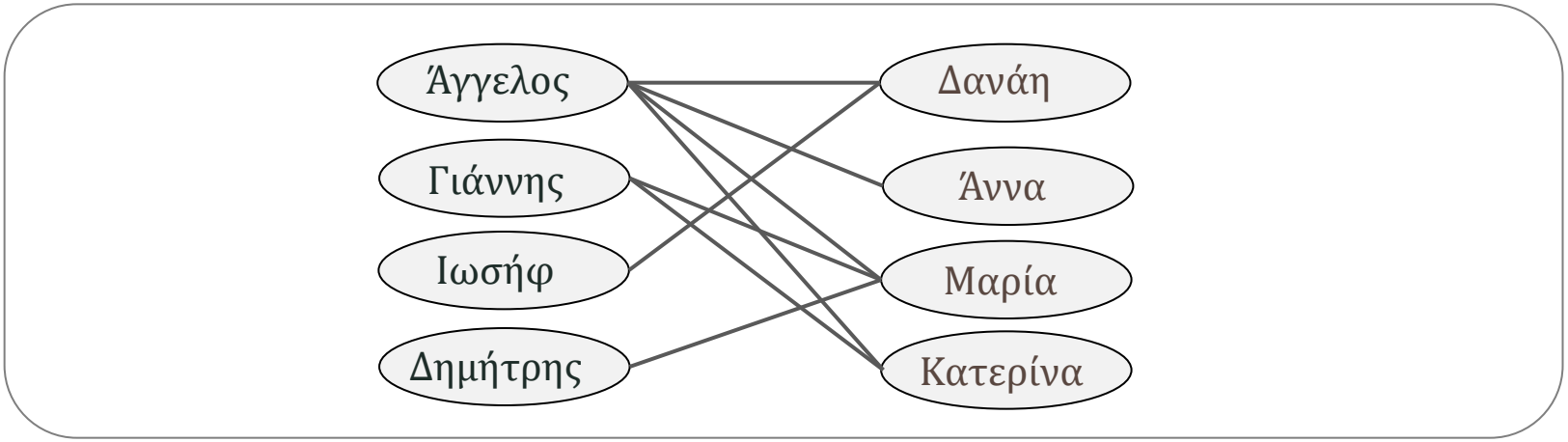
Πρόβλημα: Είναι δυνατόν να επιλεγούν ζευγάρια έτσι ώστε:

- ✓ όλοι να έχουν από ένα μόνο σύντροφο, και
- ✓ να είναι κάποιος που συμπαθούν !!!

ΤΕΛΕΙΑ-ΑΝΤΙΣΤΟΙΧΙΣΗ ≤ ΜΕΓΙΣΤΗ-ΡΟΗ

- Με όρους Γραφοθεωρίας το πρόβλημα διατυπώνεται ως εξής:
  Υπάρχει μια τέλεια αντιστοίχιση (perfect matching) ?

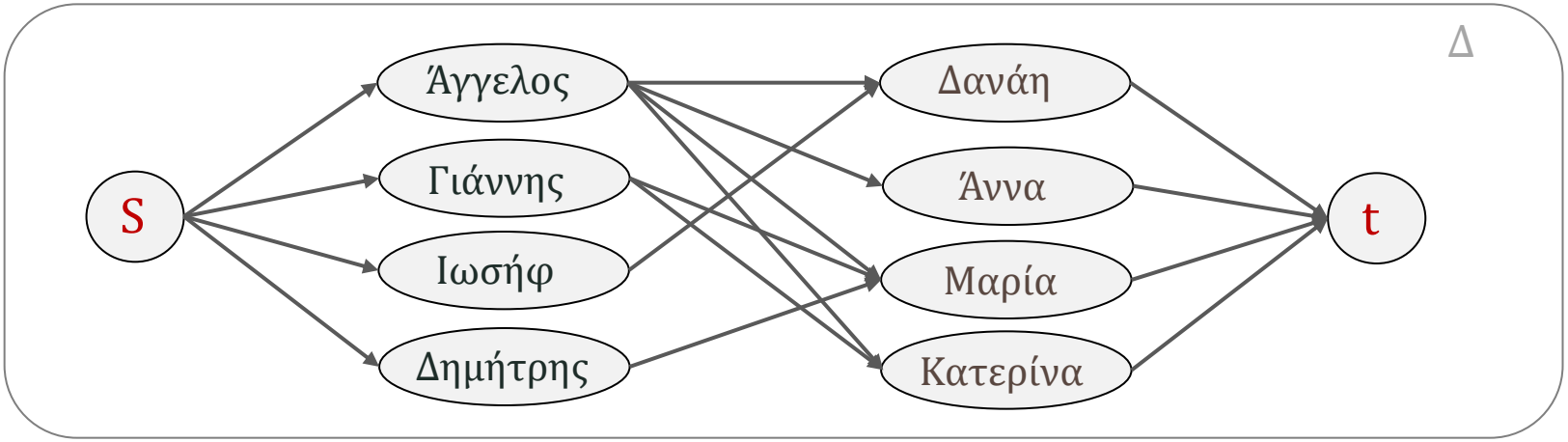
Αναγωγή !!!



Τελεια-Αντιστοιχιση ≤ Μεγιστη-Ροη

- Εισάγουμε 2 κόμβους **s** και **t**

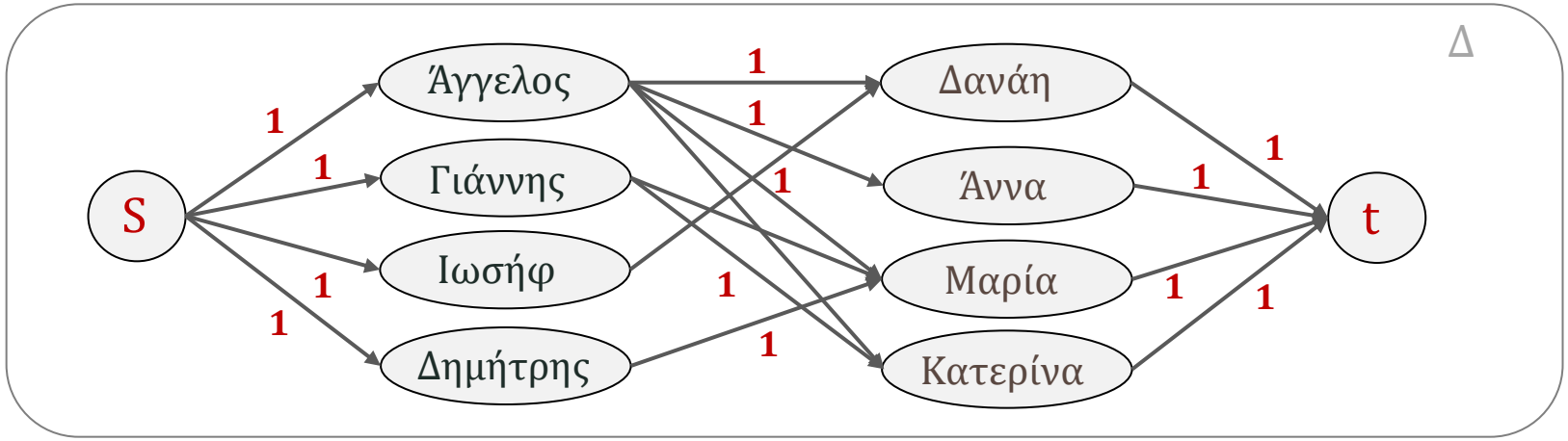
  Εισάγουμε κατευθύνσεις και ορίζουμε το δίκτυο Δ !!!



ΤΕΛΕΙΑ-ΑΝΤΙΣΤΟΙΧΙΣΗ ≤ ΜΕΓΙΣΤΗ-ΡΟΗ

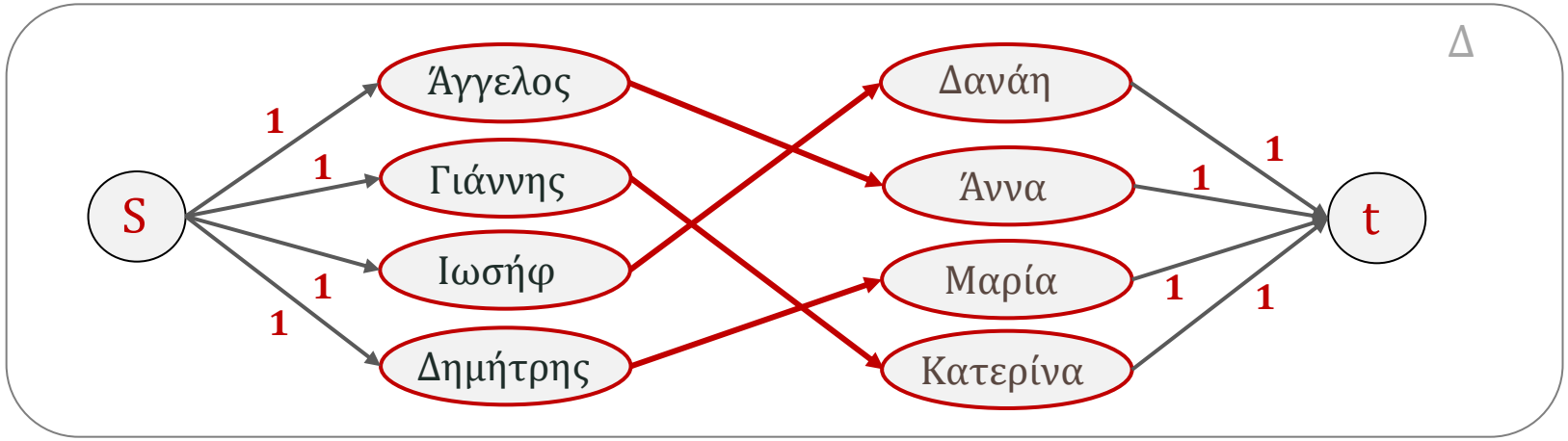
- Δίνουμε σε κάθε ακμή *χωρητικότητα* **1**

  Και τότε, υπάρχει τέλεια αντιστοίχιση **εάν-ν** το δίκτυο Δ δίνει μια ροή f με μέγεθος |f| = πλήθος των ζευγαριών !!!



Τελεια-Αντιστοιχιση ≤ Μεγιστη-Ροη

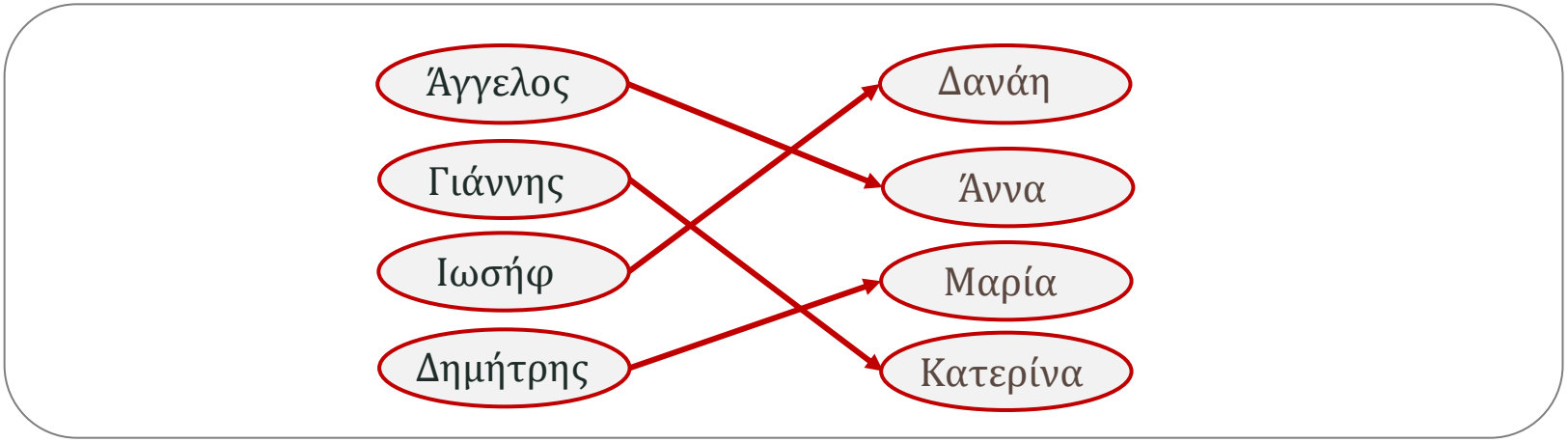
- Υπολογίζουμε μια Μέγιστη Ροή στο δίκτυο Δ



Τελεια-Αντιστοιχιση ≤ Μεγιστη-Ροη

- Υπολογίζουμε μια Μέγιστη Ροή στο δίκτυο Δ

  Και παίρνουμε μια τέλεια αντιστοίχιση !!!



Ιδιότητα: Εάν όλες οι χωρητικότητες των ακμών είναι ακέραιες, τότε η βέλτιστη ροή είναι ακέραιη !!!

- **Στο Βασίλειο των Προβλημάτων**

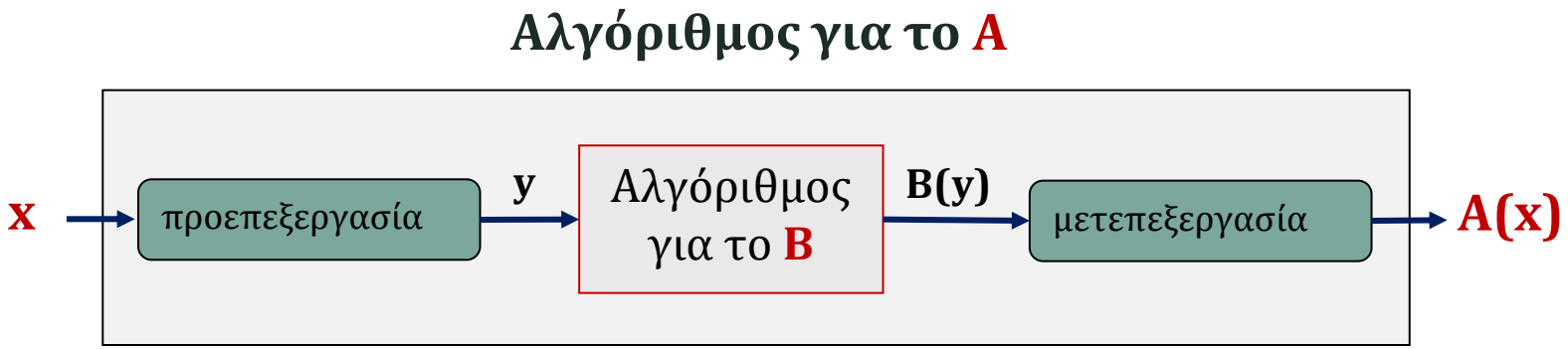
Μέγιστη Διαδρομή σε DAG $\leq$ Ελάχιστη Διαδρομή σε DAG

```
Longest-Path(G)
    for each edge (u,v)∈E(G)
        w(u,v) ← -w(u,v)
    return Shortest-Path(G)
```

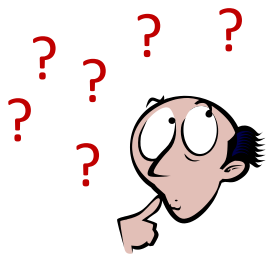
← Αλγόριθμος **Dijkstra**

Μια πιο Τυπική Θεώρηση

- Λέμε ότι **A ≤ B** εάν μπορούμε να διατυπώσουμε έναν Αλγόριθμο πολυωνυμικού χρόνου για την επίλυση του **A** *χρησιμοποιώντας* έναν Αλγόριθμο για την επίλυση του **B** !!!



Σημείωση: Στην πραγματικότητα, μπορεί να *έχουμε* ή και να *μην έχουμε* αλγόριθμο για το **B** !!!



Γιατί Αναγωγές ?

- **Αυξάνουν την ισχύ των αλγορίθμων**:

  εάν **A ≤ B** και έχουμε έναν αλγόριθμο για το **B**, τότε μπορούμε να τον χρησιμοποιήσουμε για να λύσουμε το πρόβλημα **A** !!!

- **Επιτρέπουν συγκρίσεις και κατηγοριοποιήσεις**:

  μια αναγωγή **A ≤ B** μας επιτρέπει να *συγκρίνουμε* τις πολυπλοκότητες των **A** και **B** και να *κατηγοριοποιήσουμε αυτά* ανάλογα με τη δυσκολία τους !!!

Αναγωγές:

- ✓ ΜΕΓΙΣΤΗ-ΑΥΞΟΥΣΑ-ΥΠΑΚΟΛΟΥΘΙΑ ≤ ΜΕΓΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG
- ✓ ΤΕΛΕΙΑ-ΑΝΤΙΣΤΟΙΧΙΣΗ ≤ ΜΕΓΙΣΤΗ-ΡΟΗ
- ✓ ΜΕΓΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG ≤ ΕΛΑΧΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG

Γιατί Αναγωγές ?

- **Αυξάνουν την ισχύ των αλγορίθμων**:

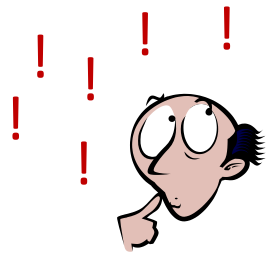
  εάν **A ≤ B** και έχουμε έναν αλγόριθμο για το **B**, τότε μπορούμε να τον χρησιμοποιήσουμε για να λύσουμε το πρόβλημα **A** !!!

- **Επιτρέπουν συγκρίσεις και κατηγοριοποιήσεις**:

  μια αναγωγή **A ≤ B** μας επιτρέπει να *συγκρίνουμε* τις πολυπλοκότητες των **A** και **B** και να *κατηγοριοποιήσουμε αυτά* ανάλογα με τη δυσκολία τους !!!

Γενικά, διευρύνουν:

- ✓ την αλγοριθμική μας ικανότητα !
- ✓ την επιστημονική μας σκέψη !!
- ✓ το πεδίο γνώσης της επιστήμης μας !!!



Γιατί Αναγωγές ?

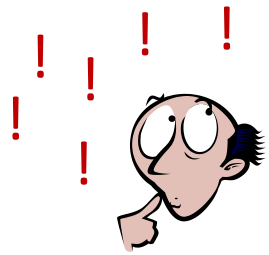
- **Ειδικά, οι αναγωγές είναι χρήσιμες:**

  - ✓ Για *επινόηση* και *δημιουργία ΝΕΩΝ αλγορίθμων* για *ΝΕΑ προβλήματα* (άνω φράγματα)

  - ✓ Για απόδειξη ότι *ΚΑΠΟΙΑ προβλήματα δεν μπορούν να εκτελεστούν γρηγορότερα από ένα πολυωνυμικό χρόνο T(n)*

    Ή για εύρεση ενδείξεων ότι *δεν υπάρχουν αποτελεσματικοί αλγόριθμοι* για *ΚΑΠΟΙΑ άλλα προβλήματα* (κάτω φράγματα)

Άνω και Κάτω Φράγματα



- Έστω η αναγωγή **A ≤ B**

✓ **Άνω φράγμα** Εάν υπάρχει αλγόριθμος πολυωνυμικού χρόνου για το **B**, τότε υπάρχει ένας και για το **A !**

Το A είναι *το πολύ* ίδιας δυσκολίας με το B !

✓ **Κάτω φράγμα** Αντίστροφα, εάν δεν υπάρχει πολυωνυμικός αλγόριθμος για το **A,** τότε δεν υπάρχει ούτε για το **B !**

Το B είναι *τουλάχιστον* ίδιας δυσκολίας με το A !

Επίλυση Προβλημάτων

Επίλυση Γρίφων

Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων

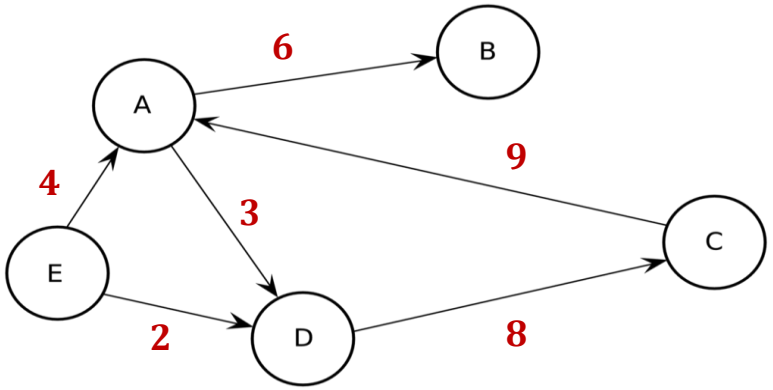
Επίλυση Προβλημάτων

Επίλυση Γρίφων

Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων



Εύρεση Κύκλου Ελάχιστου Κόστους

ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ ≤ ΕΛΑΧΙΣΤΕΣ-ΔΙΑΔΡΟΜΕΣ

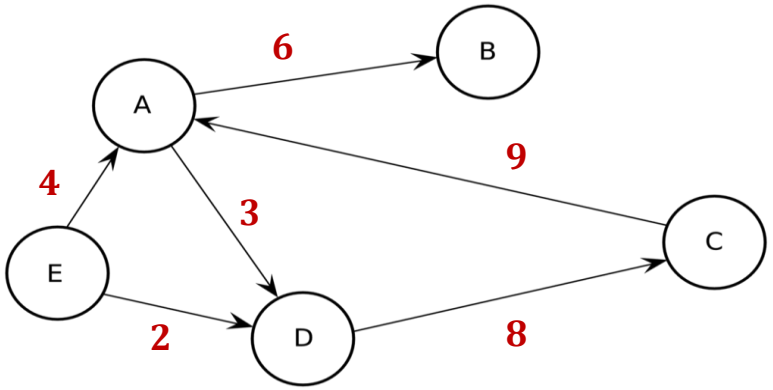
1 Εύρεση Κύκλου Ελάχιστου Κόστους

Έστω κατευθυνόμενο G = (V, E, w), w : E $\rightarrow \Re$

Το πρόβλημα **ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ** είναι το ακόλουθο:

Βρες έναν κύκλο C, εάν υπάρχει, στο G με το Ελάχιστο Κόστος !!!

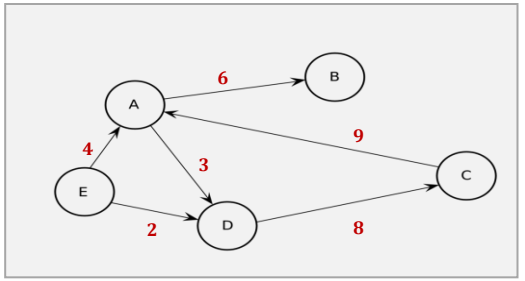
$$Cost(C) = \sum_{i=1}^{k} w(e_i)$$

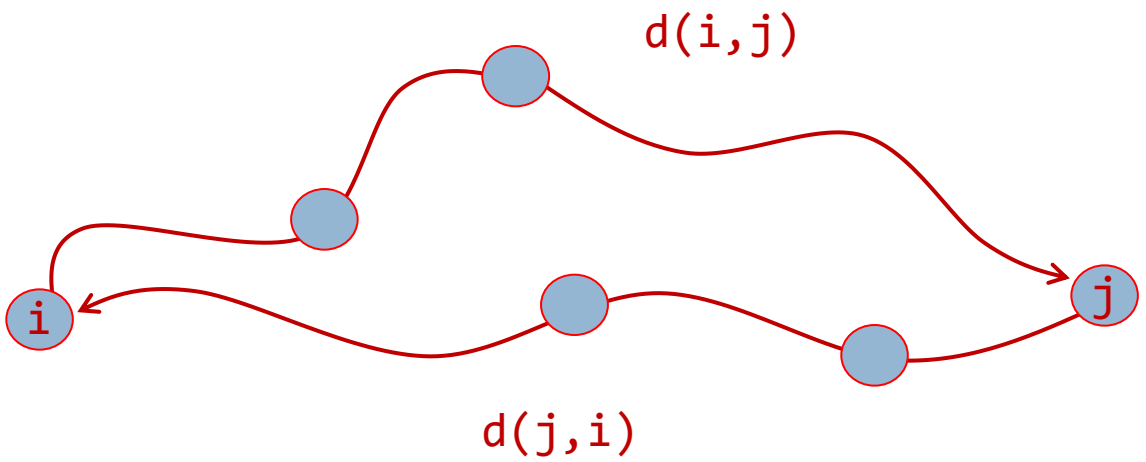


ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ ≤ ΕΛΑΧΙΣΤΕΣ-ΔΙΑΔΡΟΜΕΣ

Υπάρχει απλός τρόπος αναγωγής αυτού του προβλήματος σε πρόβλημα Ελαχίστων Διαδρομών !!!

Βρες την ε.δ $P_1 : i \rightsquigarrow j$ και την ε.δ $P_2 : j \rightsquigarrow i$





Μια Πρώτη Σκέψη !!!

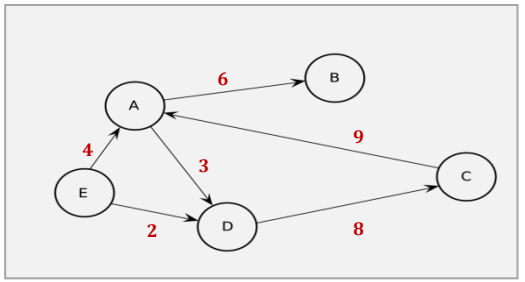
Κύκλος ?... ΝΑΙ

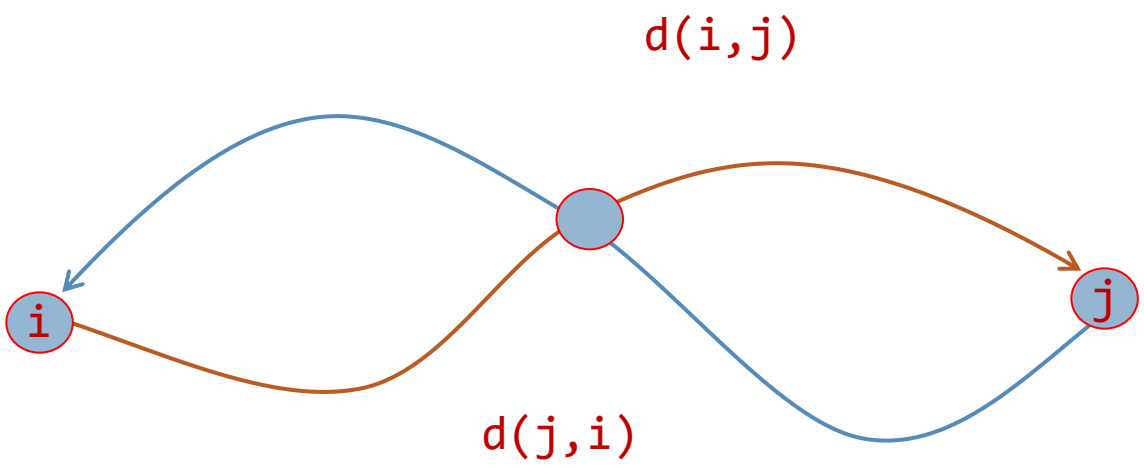
Απλός Κύκλος ?... ΟΧΙ

ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ ≤ ΕΛΑΧΙΣΤΕΣ-ΔΙΑΔΡΟΜΕΣ

Υπάρχει απλός τρόπος αναγωγής αυτού του προβλήματος σε πρόβλημα Ελαχίστων Διαδρομών !!!

Βρες την ε.δ $P_1 : i \rightsquigarrow j$ και την ε.δ $P_2 : j \rightsquigarrow i$





Μια Πρώτη Σκέψη !!!

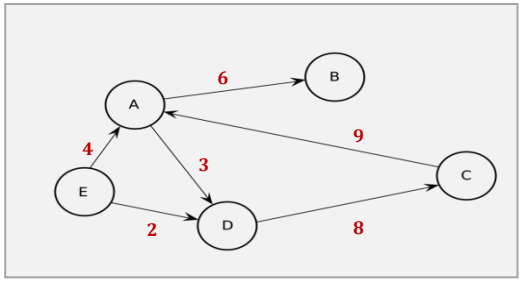
Κύκλος ?... ΝΑΙ

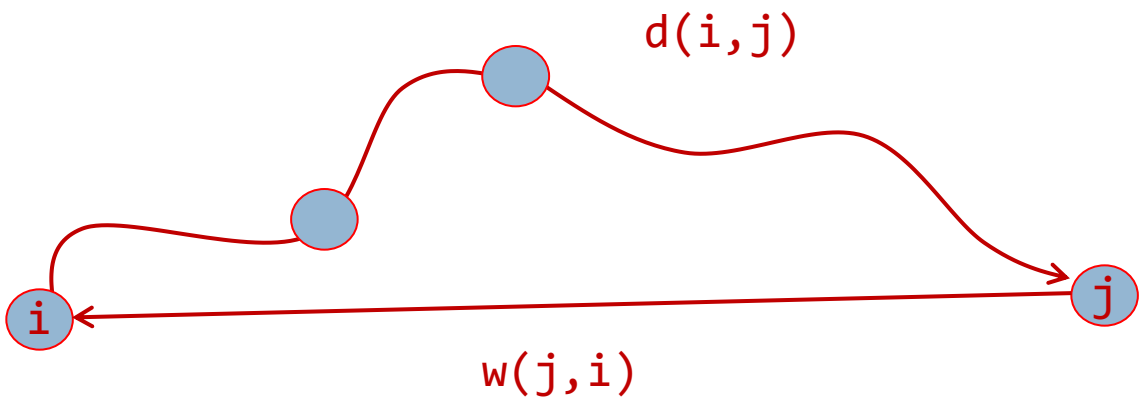
Απλός Κύκλος ?... ΟΧΙ

ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ ≤ ΕΛΑΧΙΣΤΕΣ-ΔΙΑΔΡΟΜΕΣ

Υπάρχει απλός τρόπος αναγωγής αυτού του προβλήματος σε πρόβλημα Ελαχίστων Διαδρομών !!!

Βρες την ε.δ $\mathbf{P_1} : \mathtt{i} \rightsquigarrow \mathtt{j}$ και την ε.δ $\mathbf{w} : \mathtt{j} \longrightarrow \mathtt{i}$





Μια Λύση !!!

Κύκλος ?... ΝΑΙ

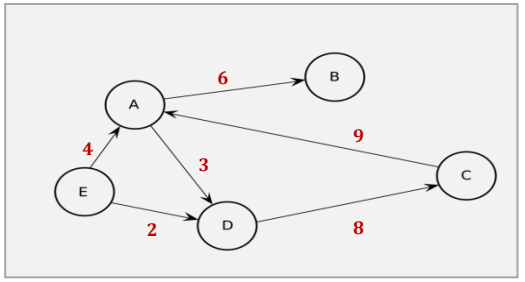
Απλός Κύκλος ?... ΝΑΙ

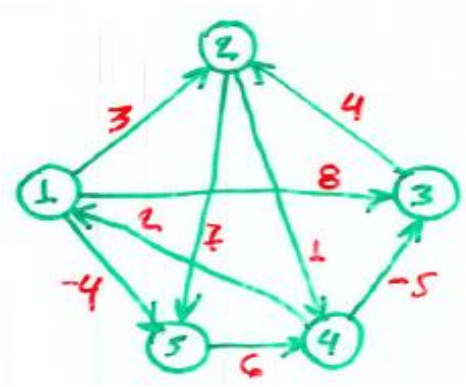
ΕΛΑΧΙΣΤΟΣ-ΚΥΚΛΟΣ ≤ ΕΛΑΧΙΣΤΕΣ-ΔΙΑΔΡΟΜΕΣ

Για κάθε ζεύγος κόμβων $(i,j) \in V$

- ✓ Βρες την ε.δ $P_1 : i \rightsquigarrow j$ και $d(i,j)$
- ✓ Έλεγξε εάν (j,i) είναι ακμή

  Εάν $(j,i) \in E$ τότε $Cost(C_1) = d(i,j) + w(j,i)$





	1	2	3	4	5
1	0	3	-3	2	-4
2	3	0	-4	1	-1
3	6	4	0	5	3
4	2	-1	-5	0	-2
5	7	5	1	6	0

Πανζευκτικές Ελάχιστες Διαδρομές

Floyd-Warshall

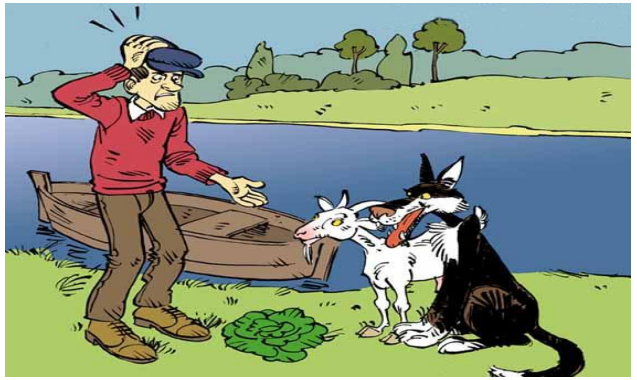
Επίλυση Προβλημάτων

Επίλυση Γρίφων

Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων

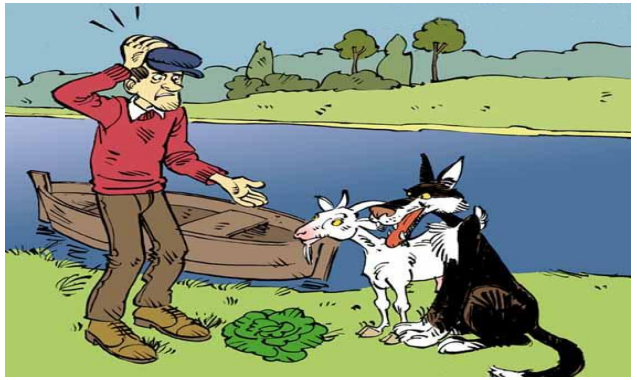


Το Πρόβλημα του Βοσκού

ΠΡΟΒΛΗΜΑ-ΒΟΣΚΟΥ ≤ ΠΡΟΒΛΗΜΑ-ΠΡΟΣΒΑΣΙΜΟΤΗΤΑΣ

2 Το Πρόβλημα του Βοσκού

Ένας βοσκός θέλει να περάσει στην απέναντι όχθη ενός ποταμού έναν **Λύκο**, ένα **Πρόβατο** και ένα **Λάχανο** ή Χορτάρι !



Πως είναι δυνατόν ο βοσκός να τα περάσει απέναντι, έτσι ώστε να μην φάει το πρόβατο το χορτάρι, ή ο λύκος το πρόβατο κ.ο.κ !

Το Πρόβλημα του Βοσκού

Ο Λύκος, το Πρόβατο και το Χορτάρι

Στη Βάρκα:

Βοσκός
Βοσκός & Λύκος
Βοσκός & Πρόβατο
Βοσκός & Χορτάρι

Καταστάσεις Μετάβασης

1	ΛΠΧ-Β	|
2	ΠΧ-Β	| Λ
3	ΛΧ-Β	| Π
4	ΛΠ-Β	| Χ
5	Π-Β	| ΛΧ
6	ΛΧ	| Β-Π
7	Χ	| Β-ΛΠ
8	Π	| Β-ΛΧ
9	Λ	| Β-ΠΧ
10		| Β-ΛΠΧ

Επιτρεπτες Καταστασεις Μεταβασης

Χ Π Λ - Β |

Αρχική Κατάσταση

⇒

| Β – Λ Π Χ

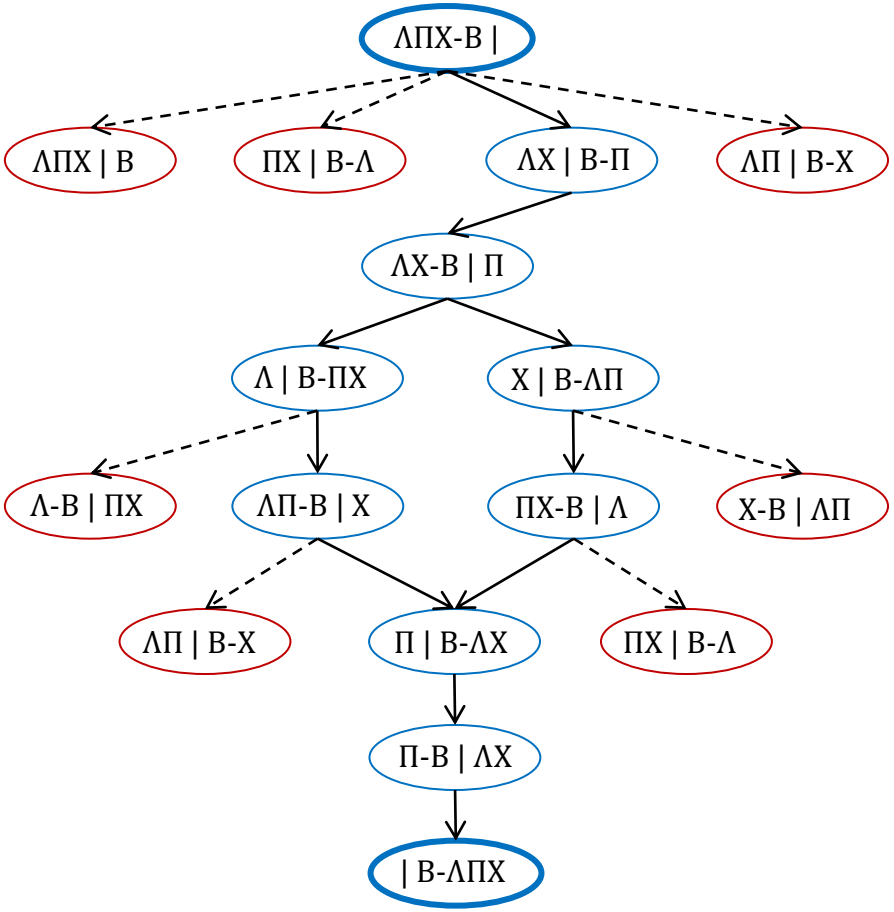
Τελική Κατάσταση

Το Πρόβλημα του Βοσκού

Πρόβλημα-Βοσκού ≤ Πρόβλημα-Προσβασιμοτητας

1	ΛΠΧ-Β	|
2	ΠΧ-Β	| Λ
3	ΛΧ-Β	| Π
4	ΛΠ-Β	| Χ
5	Π-Β	| ΛΧ
6	ΛΧ	| Β-Π
7	Χ	| Β-ΛΠ
8	Π	| Β-ΛΧ
9	Λ	| Β-ΠΧ
10		| Β-ΛΠΧ

Επιτρεπτες Καταστασεις Μεταβασης

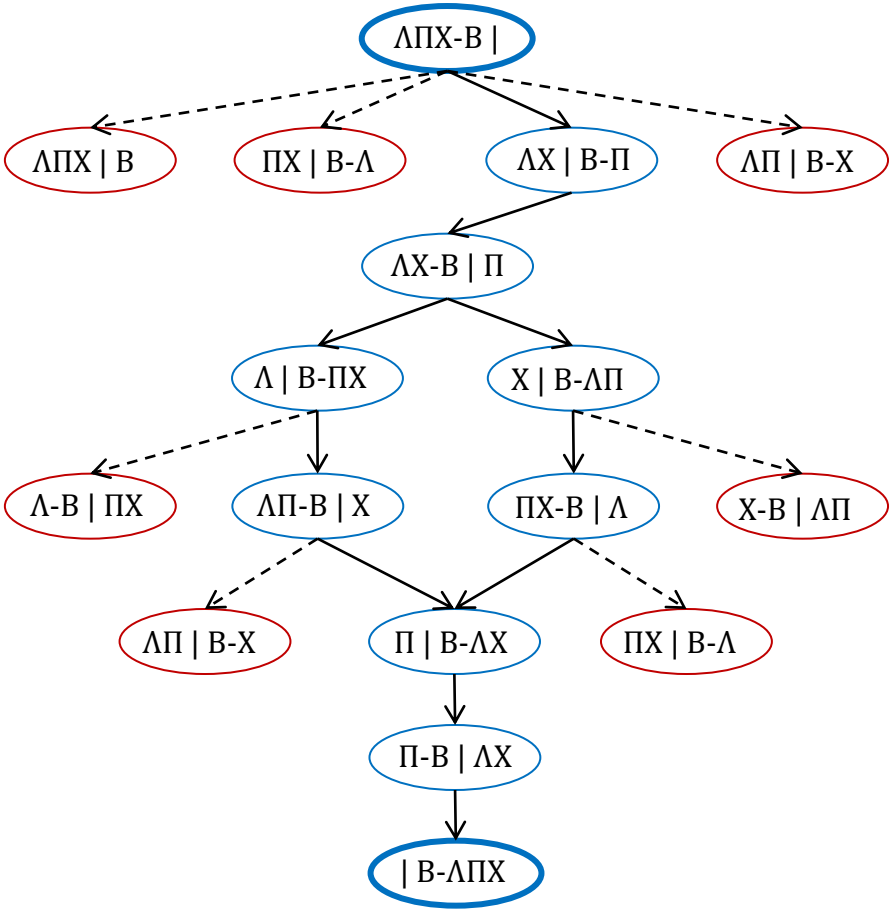


Πρόβλημα-Βοσκού ≤ Πρόβλημα-Προσβασιμοτητας

- ✓ DFS
  Γράφημα καταστάσεων !
- ✓ Κόμβος Εκκίνησης

  ΛΠΧ-Β |
- ✓ Κόμβος Πρόσβασης

  ΛΠΧ-Β |

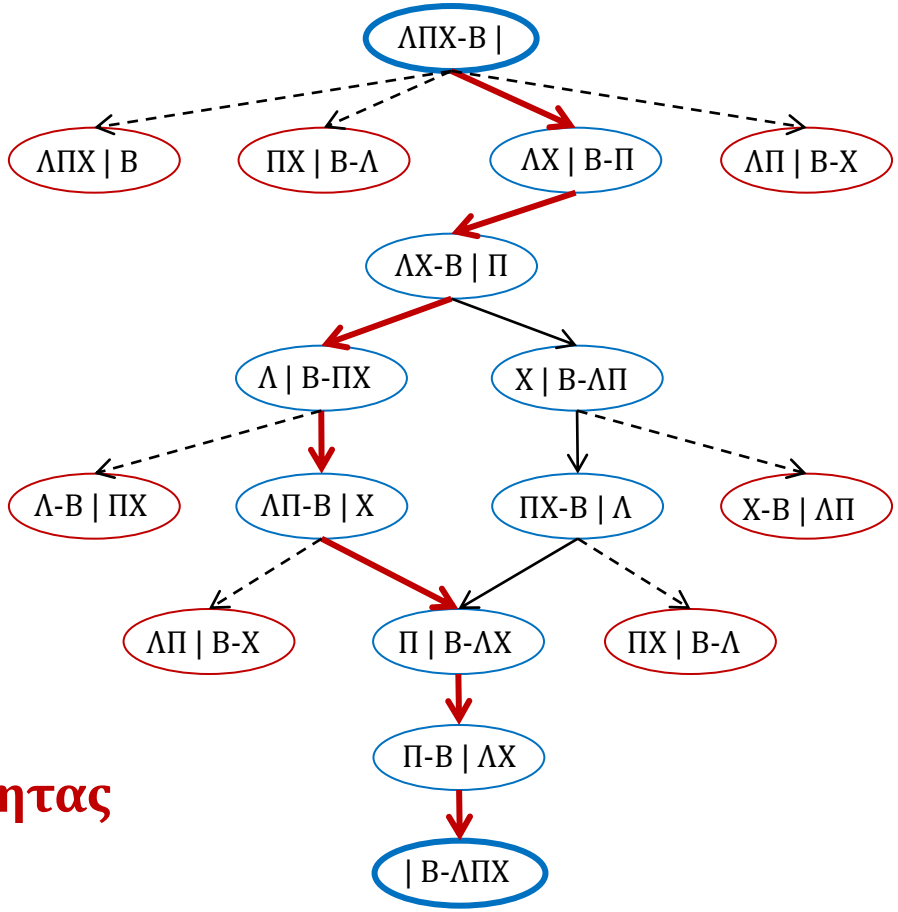


ΠΡΟΒΛΗΜΑ-ΒΟΣΚΟΥ ≤ ΠΡΟΒΛΗΜΑ-ΠΡΟΣΒΑΣΙΜΟΤΗΤΑΣ

- ✓ DFS
  Γράφημα καταστάσεων !
- ✓ Κόμβος Εκκίνησης

  ΛΠΧ-Β |
- ✓ Κόμβος Πρόσβασης

  ΛΠΧ-Β |
- ✓ **Λύση Προβλήματος Προσβασιμότητας**



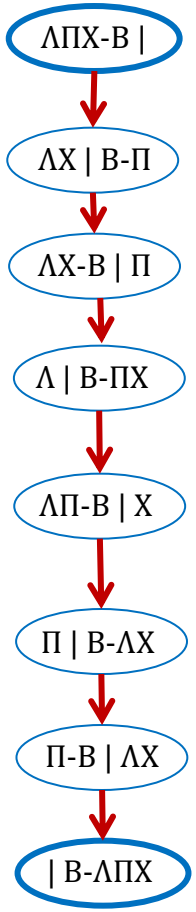
Το Πρόβλημα του Βοσκού

ΠΡΟΒΛΗΜΑ-ΒΟΣΚΟΥ ≤ ΠΡΟΒΛΗΜΑ-ΠΡΟΣΒΑΣΙΜΟΤΗΤΑΣ

- ✓ DFS
  Γράφημα καταστάσεων !
- ✓ Κόμβος Εκκίνησης

  ΛΠΧ-Β |
- ✓ Κόμβος Πρόσβασης

  ΛΠΧ-Β |
- ✓ **Λύση Προβλήματος Βοσκού**



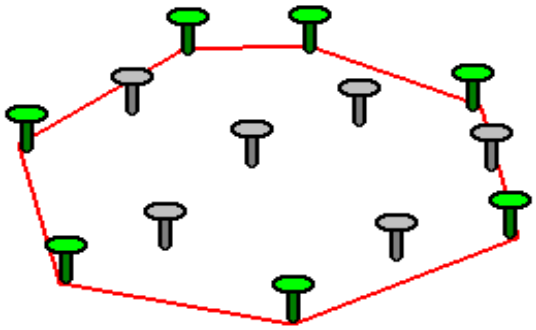
Επίλυση Προβλημάτων

Επίλυση Γρίφων

Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων



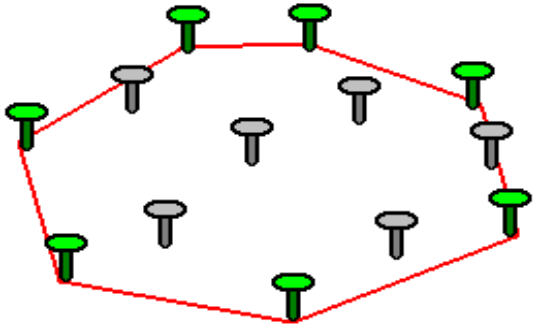
Κάτω Φράγμα για Κυρτό Περίβλημα

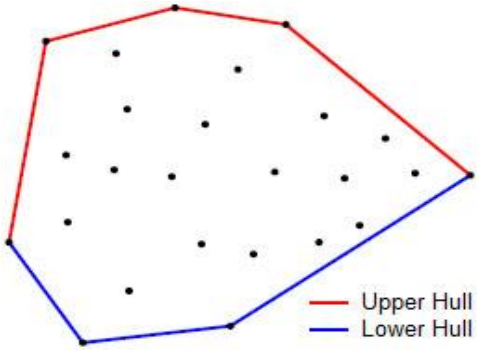
ΤΑΞΙΝΟΜΗΣΗ ≤ ΚΥΡΤΟ-ΠΕΡΙΒΛΗΜΑ

3 Κάτω Φράγμα για Κυρτό Περίβλημα

Ερώτημα !!!...

Πόσο γρήγορα μπορώ να υπολογίσω το **Κυρτό Περίβλημα** Ν σημείων στο επίπεδο?



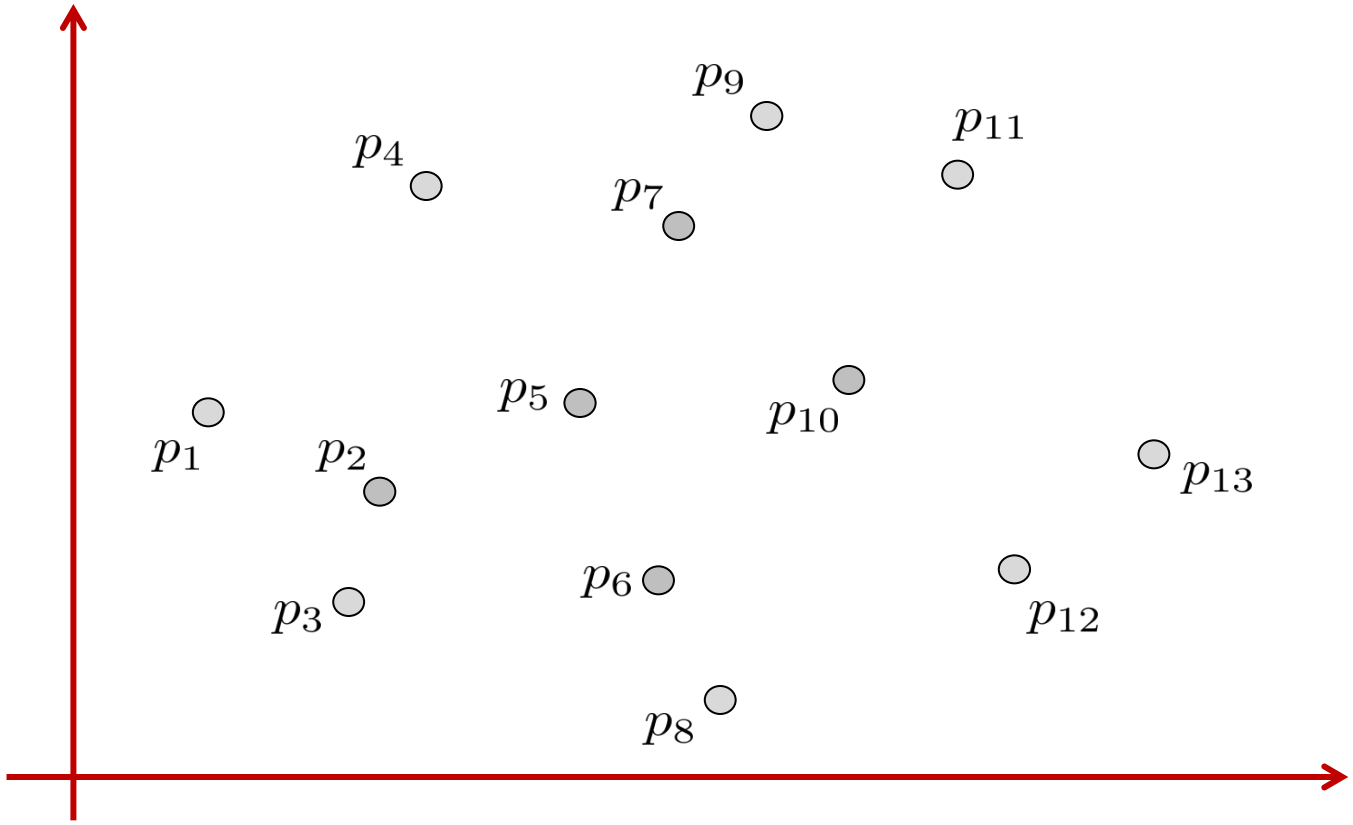


Αρκεί να βρω ένα πρόβλημα **Α**, με γνωστό ένα κάτω φράγμα υπολογισμού του, και να αναγάγω το Α στο Πρόβλημα του Κυρτού Περιβλήματος !!!

Κάτω Φράγμα για Κυρτό Περίβλημα

Ορισμοί

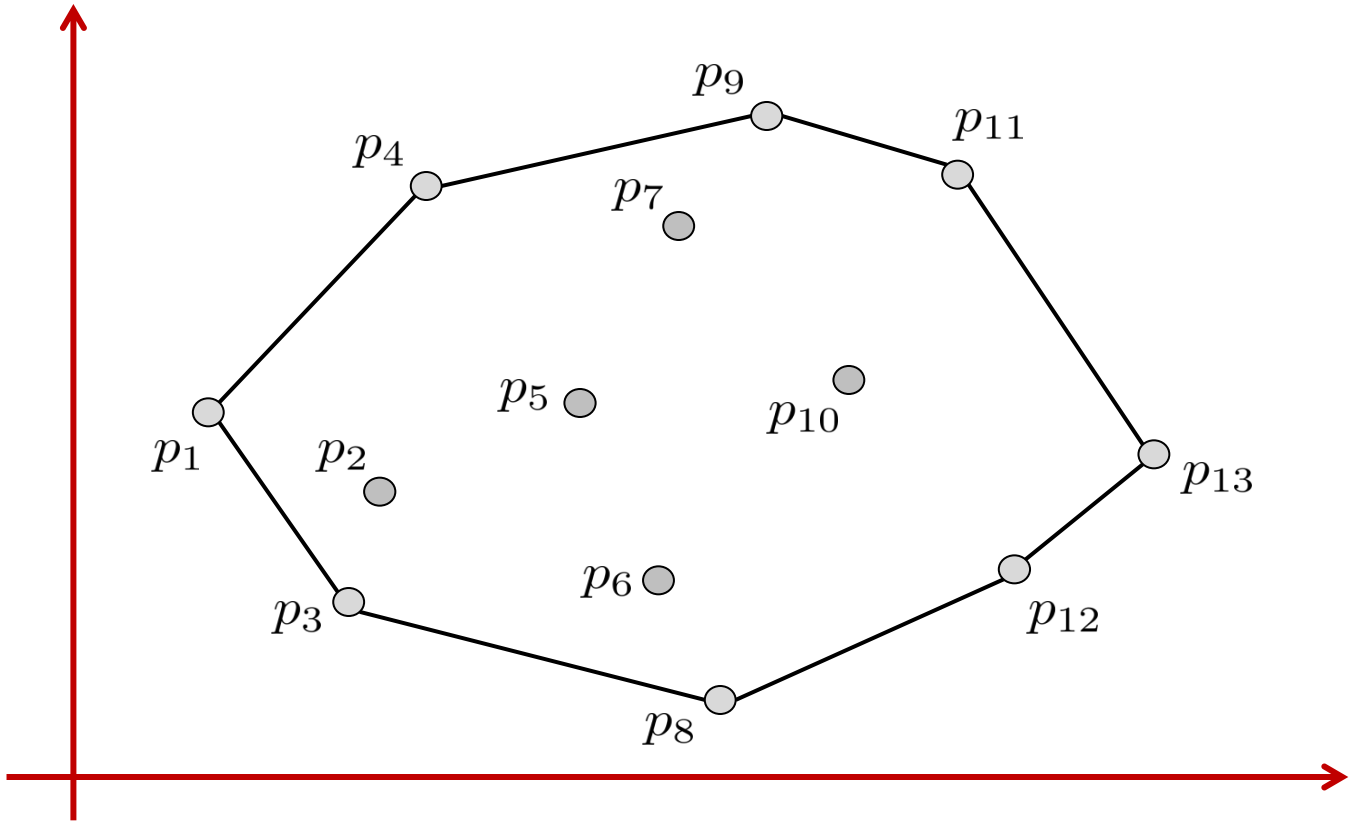
Το **κυρτό περίβλημα** συνόλου σημείων του επιπέδου είναι το κυρτό πολύγωνο με το ελάχιστο εμβαδόν που περιλαμβάνει όλα τα δεδομένα σημεία



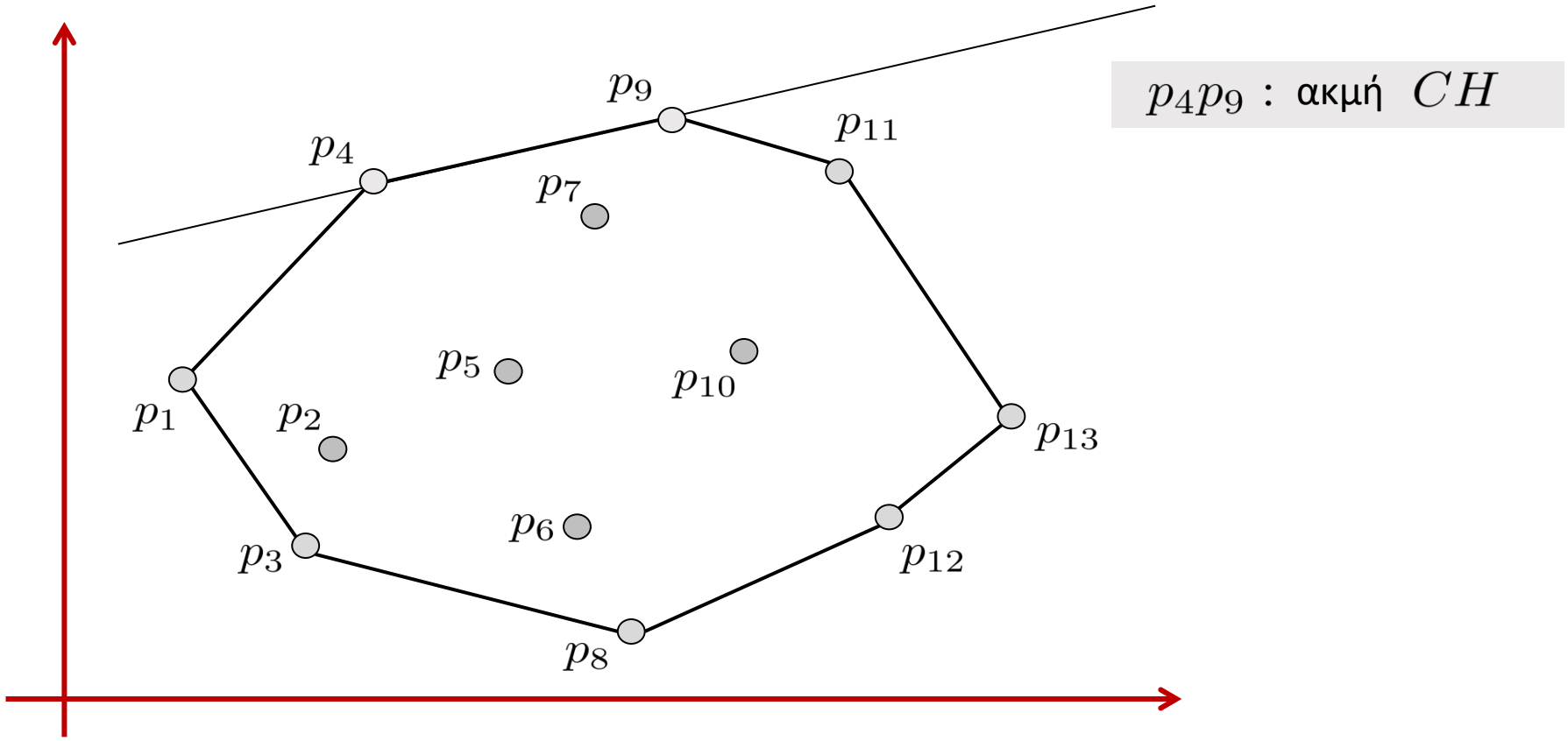
Κάτω Φράγμα για Κυρτό Περίβλημα

Ορισμοί

Το **κυρτό περίβλημα** συνόλου σημείων του επιπέδου είναι το κυρτό πολύγωνο με το ελάχιστο εμβαδόν που περιλαμβάνει όλα τα δεδομένα σημεία

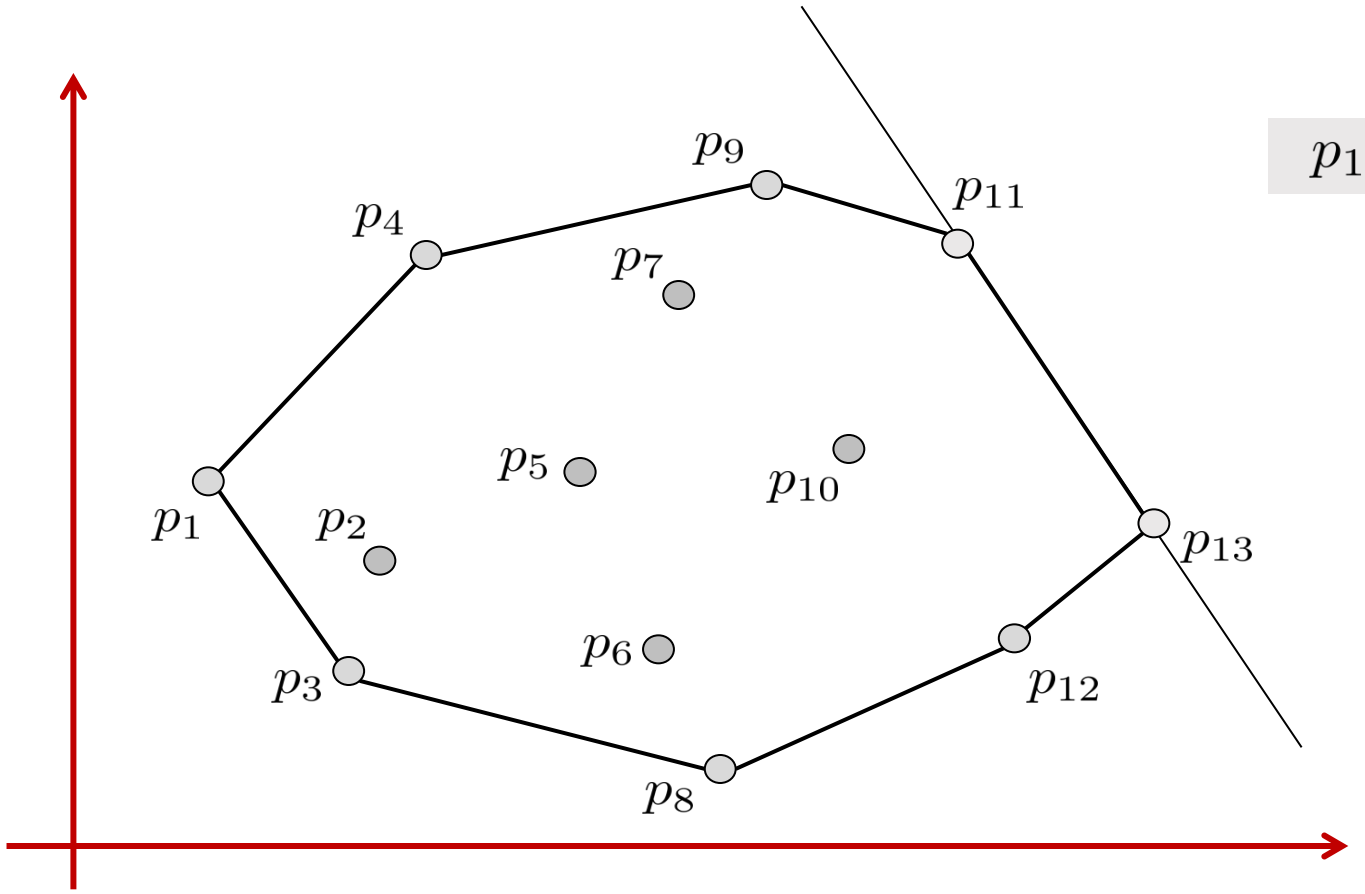


- **Ορισμοί**



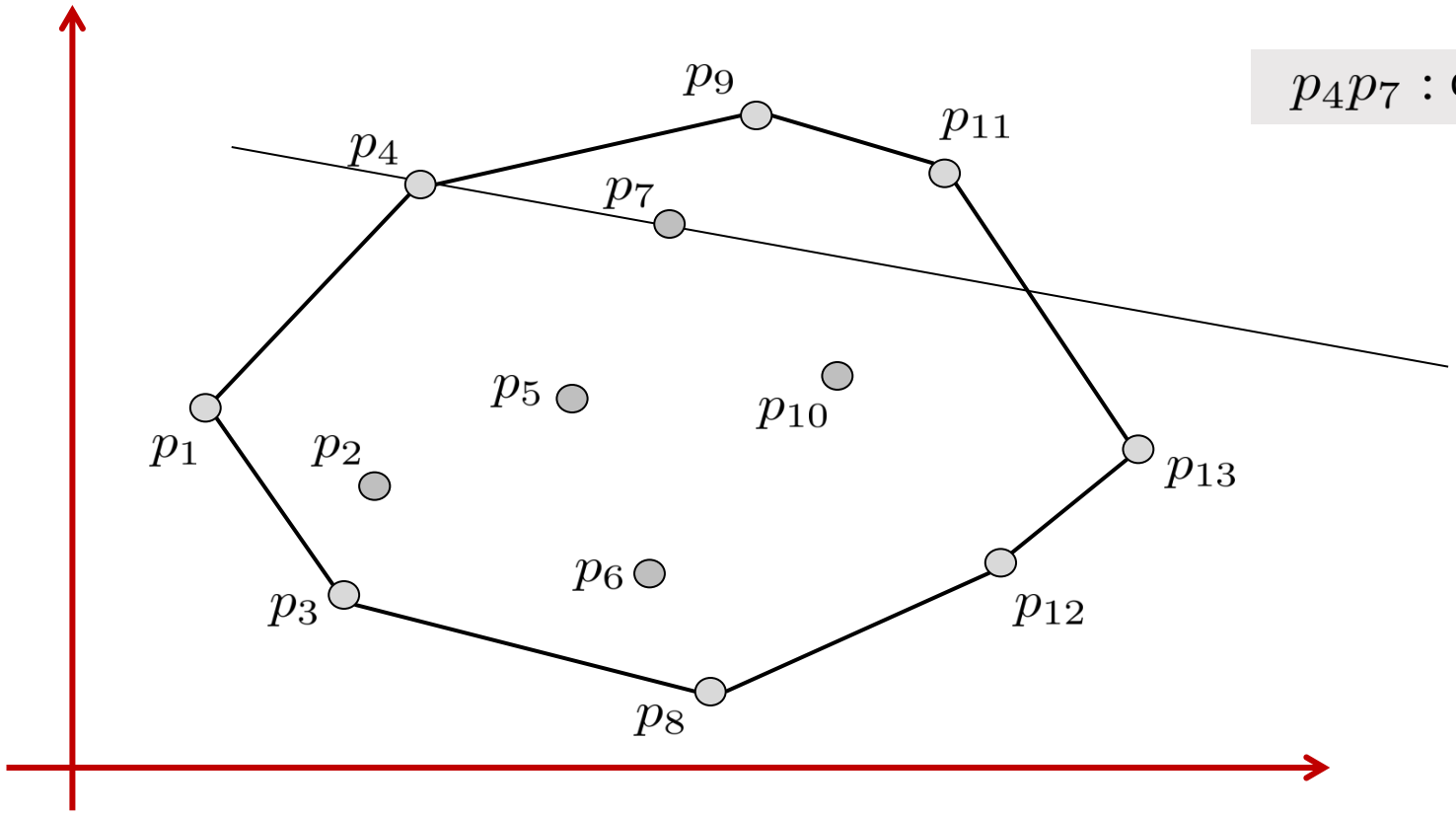
Κάτω Φράγμα για Κυρτό Περίβλημα

- **Ορισμοί**



$p_{11}p_{13}$: ακμή CH

Ορισμοί

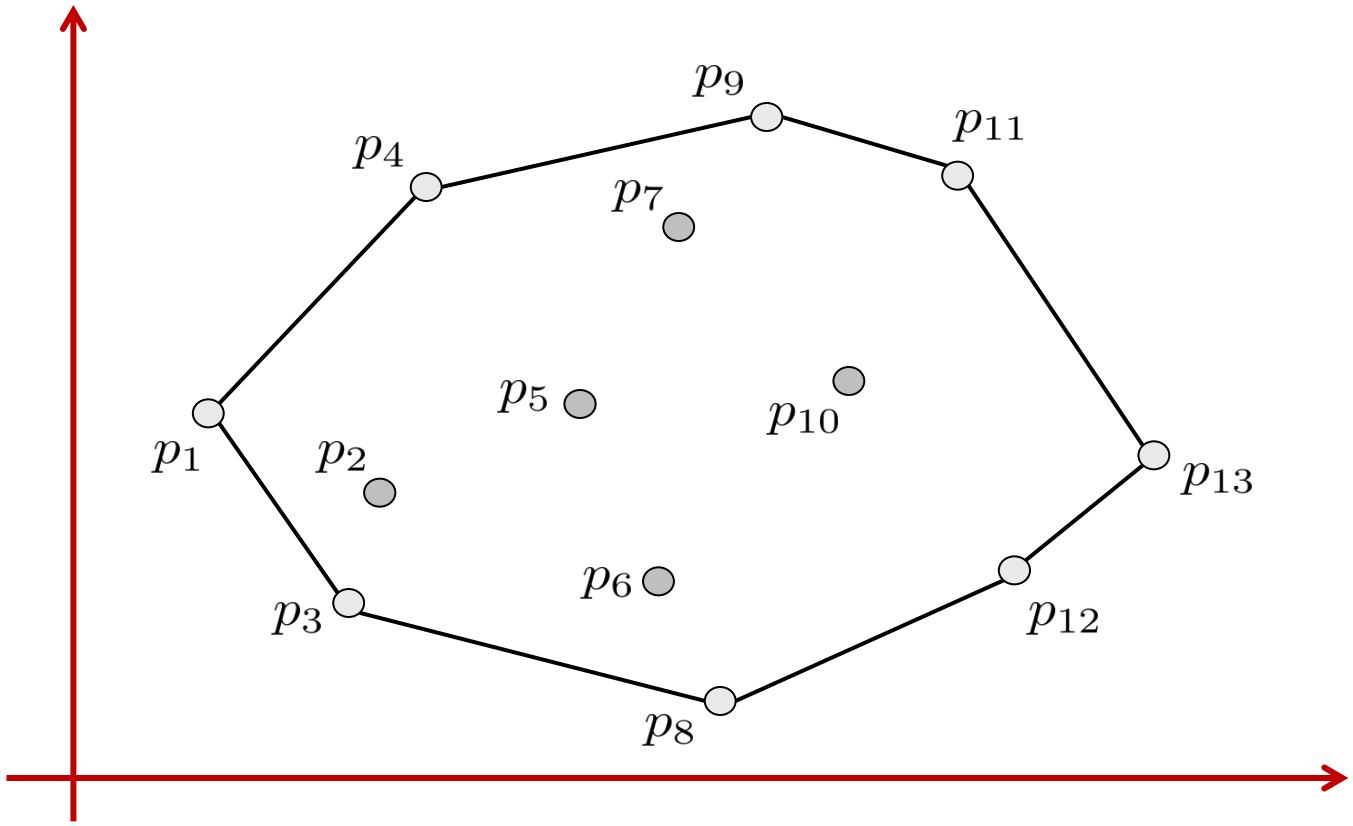


p_4p_7 : όχι ακμή CH

Κάτω Φράγμα για Κυρτό Περίβλημα

Ορισμοί

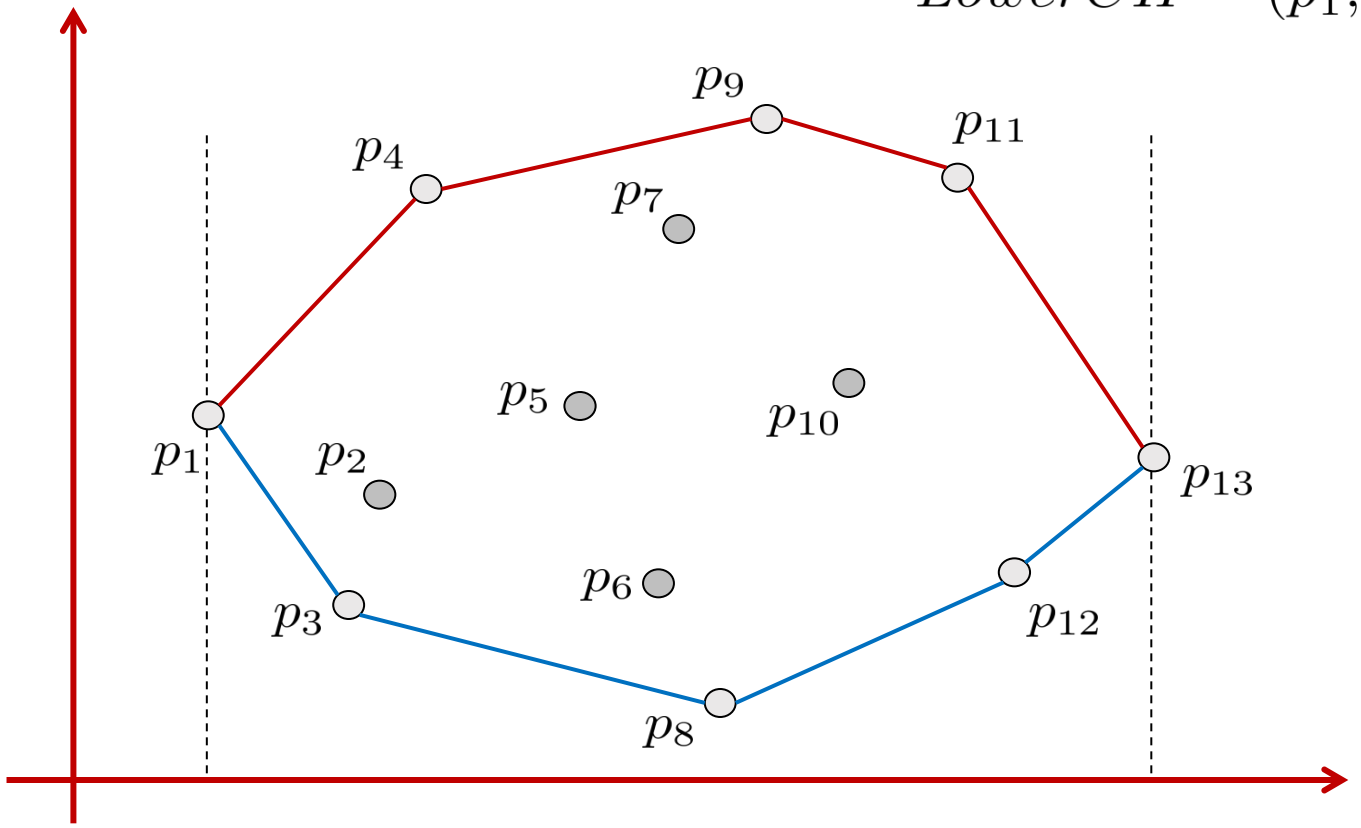
$$CH = (p_1, p_3, p_8, p_{12}, p_{13}, p_{11}, p_9, p_4)$$



Κάτω Φράγμα για Κυρτό Περίβλημα

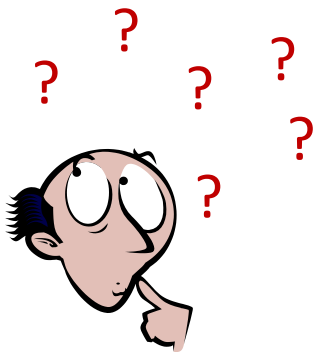
- **Ορισμοί**

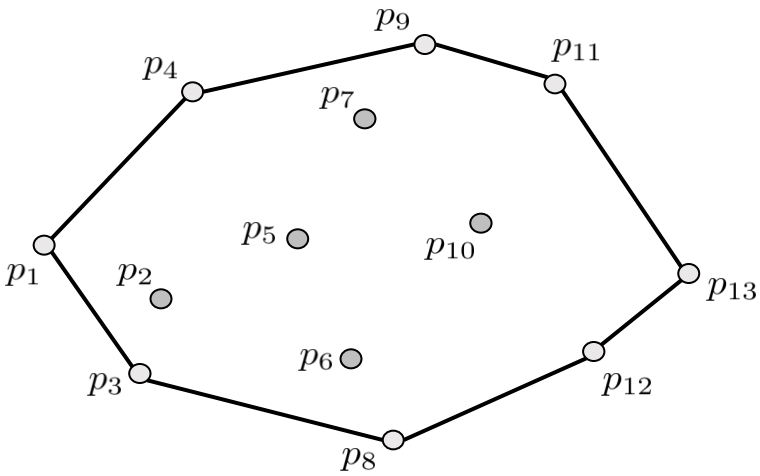
$$UpperCH = (p_1, p_4, p_9, p_{11}, p_{13})$$
$$LowerCH = (p_1, p_3, p_8, p_{12}, p_{13})$$



Κάτω Φράγμα για Κυρτό Περίβλημα

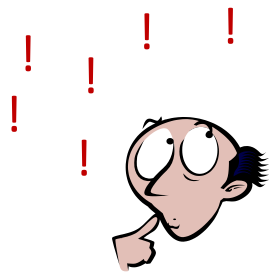
- **Ερώτημα !!!...** Πόσο γρήγορα μπορώ να υπολογίσω το **Κυρτό Περίβλημα N** σημείων στο επίπεδο ?





Κάτω Φράγμα για Κυρτό Περίβλημα

- **Απόδειξη !!!...** ενός κάτω φράγματος για το Πρόβλημα του **Κυρτού Περιβλήματος !**

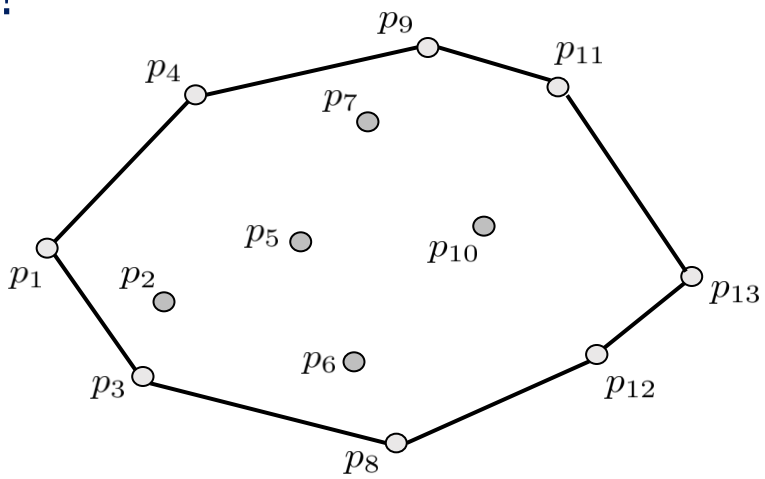


- ✓ Απόδειξη με **αναγωγή** από το πρόβλημα της **Ταξινόμησης** N πραγματικών αριθμών !

- ✓ Το πρόβλημα της Ταξινόμησης έχει πολυπλοκότητα χρόνου

  O(N logN)

  ΚΑΙ



είναι γνωστό ένα **κάτω φράγμα** Ω(N logN) υπολογισμού του !!!

Κάτω Φράγμα για Κυρτό Περίβλημα

ΤΑΞΙΝΟΜΗΣΗ ≤ ΚΥΡΤΟ-ΠΕΡΙΒΛΗΜΑ

- ✓ Έστω $\{x_1, x_2, \ldots, x_n\}$ τα δεδομένα του προβλήματος **ΤΑΞΙΝΟΜΗΣΗ** (*n* στοιχεία που θέλουμε να ταξινομήσουμε)

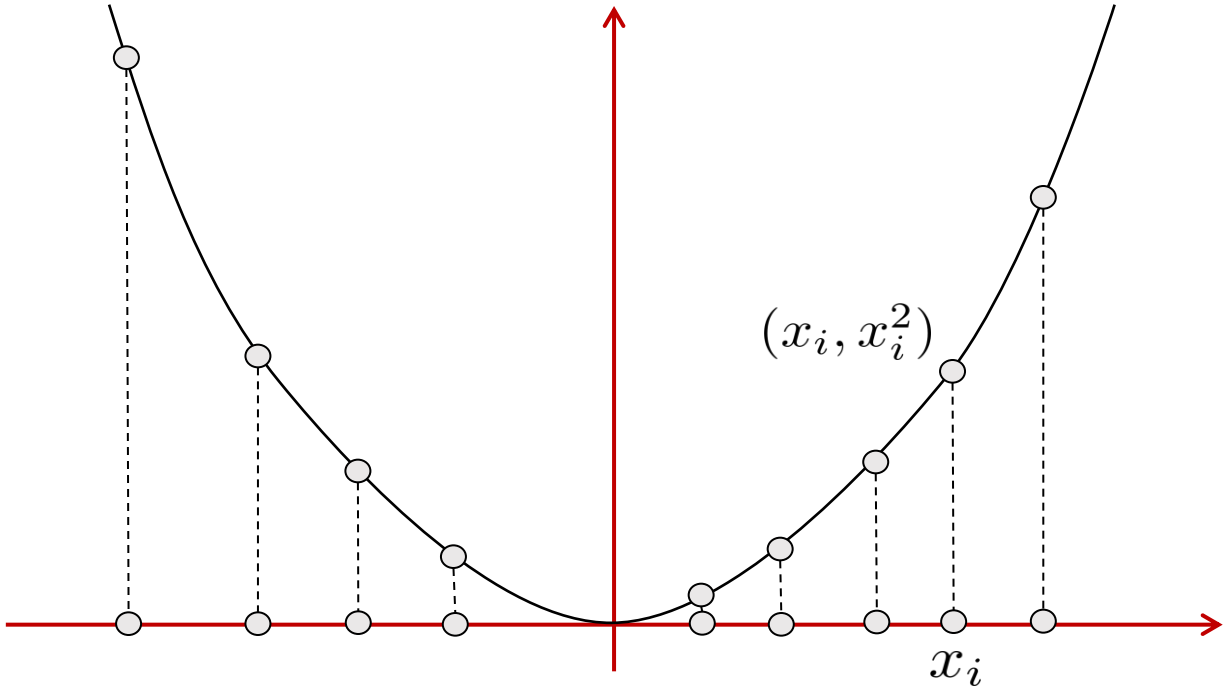
- ✓ **Κανόνας μετατροπής** των δεδομένων $\{x_1, x_2, \ldots, x_n\}$ σε δεδομένα εισόδου στο Πρόβλημα **ΚΥΡΤΟ-ΠΕΡΙΒΛΗΜΑ**:

Για κάθε πραγματικό αριθμό x_i *ορίζουμε στο επίπεδο ένα σημείο* (x_i, x_i^2)

$$x_i \quad \longleftrightarrow \quad (x_i, x_i^2)$$

Κάτω Φράγμα για Κυρτό Περίβλημα

Ταξινομηση ≤ Κυρτο-Περιβλημα



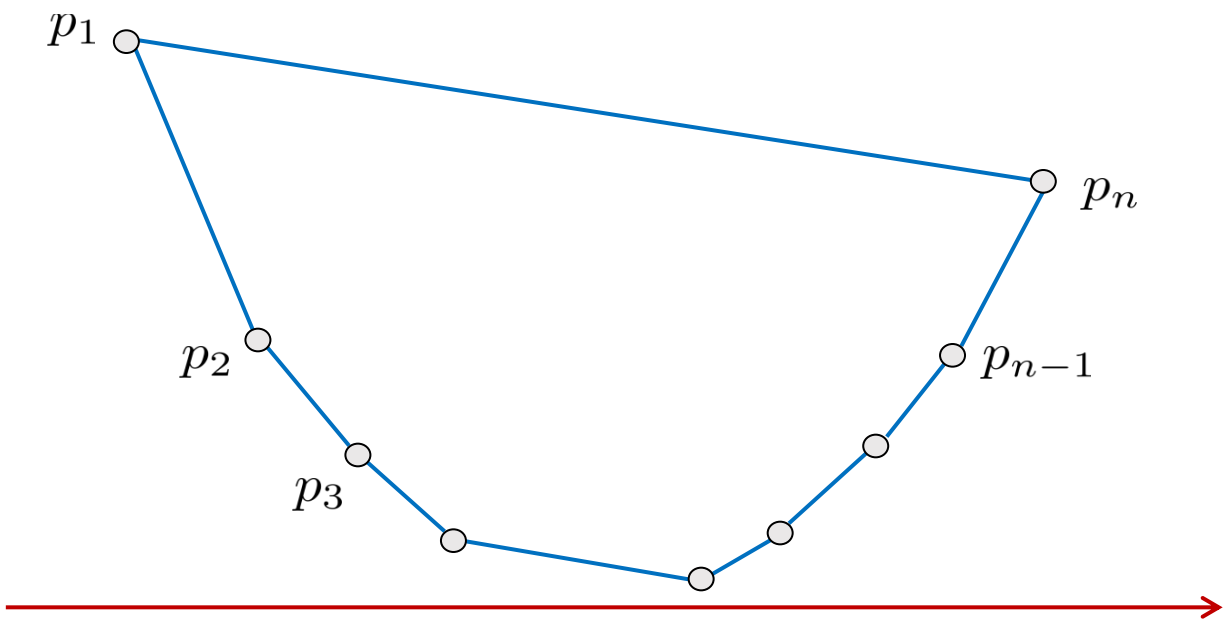
- ✓ Παραβολή

  n σημεία στο επίπεδο

Η κατασκευή των n σημείων στο επίπεδο, από τους n αριθμούς, απαιτεί χρόνο **O(n)**

Κάτω Φράγμα για Κυρτό Περίβλημα

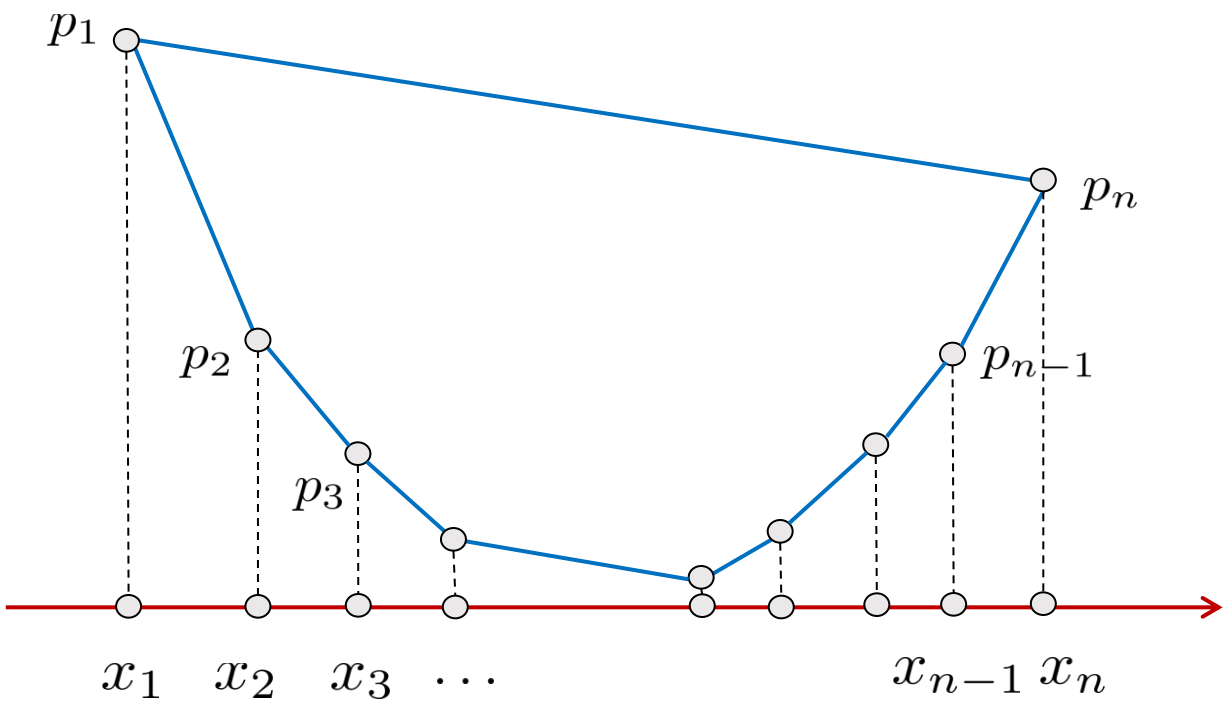
Ταξινομηση ≤ Κυρτο-Περιβλημα



- ✓ Υπολογισμός $CH = (p_1, p_2, \ldots, p_n)$

Κάτω Φράγμα για Κυρτό Περίβλημα

Ταξινομηση ≤ Κυρτο-Περιβλημα



- ✓ Υπολογισμός $CH = (p_1, p_2, \ldots, p_n)$

⇓

Υπολογισμός $S = (x_1, x_2, \ldots, x_n)$

Από CH παίρνουμε τα x_i του συνόλου $\{x_1, x_2, \ldots, x_n\}$ ταξινομημένα κατ' αύξουσα τάξη !!! σε χρόνο **O(n)**

Κάτω Φράγμα για Κυρτό Περίβλημα

Ταξινομηση ≤ Κυρτο-Περιβλημα

Έστω ότι ο υπολογισμός του Κυρτού Περιβλήματος ενός συνόλου n σημείων στο επίπεδο απαιτεί χρόνο **T(n)**

- ✓ Τότε, η Ταξινόμηση n αριθμών μπορεί να γίνει σε χρόνο: **O(n) + T(n)**
- ✓ Όμως, είναι γνωστό ότι το πρόβλημα της Ταξινόμησης έχει κάτω φράγμα υπολογισμού: **Ω(n logn)**

Θεώρημα: Ο υπολογισμός του Κυρτού Περιβλήματος ενός συνόλου n σημείων στο επίπεδο απαιτεί χρόνο Ω(n logn)

Επίλυση Προβλημάτων

Επίλυση Γρίφων

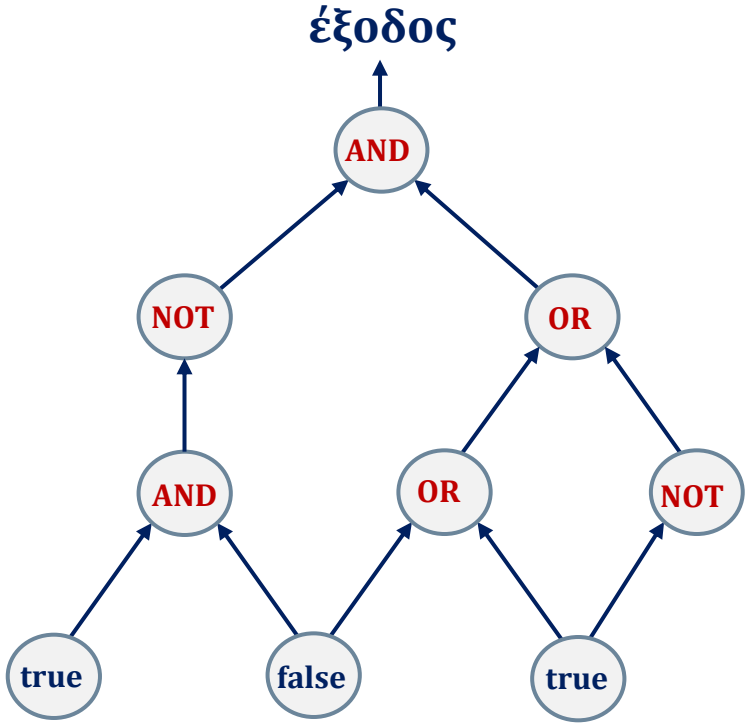
Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων

Αποτίμηση Κυκλώματος

ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ≤ LP

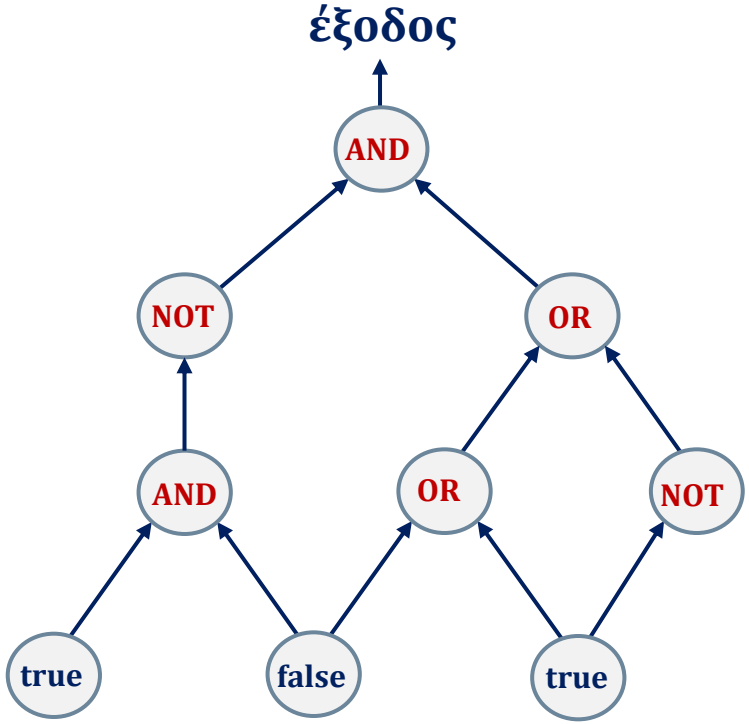


4 Αποτίμηση Κυκλώματος

Μας δίδεται ένα **λογικό κύκλωμα** (Boolean circuit), δηλαδή ένα **DAG**, με πύλες των ακόλουθων τύπων:

- ✓ πύλες **ΕΙΣΟΔΟΥ** (input gates) με βαθμό εισόδου 0 και τιμή true ή false
- ✓ πύλες **AND** και **OR**
- ✓ πύλες **NOT**

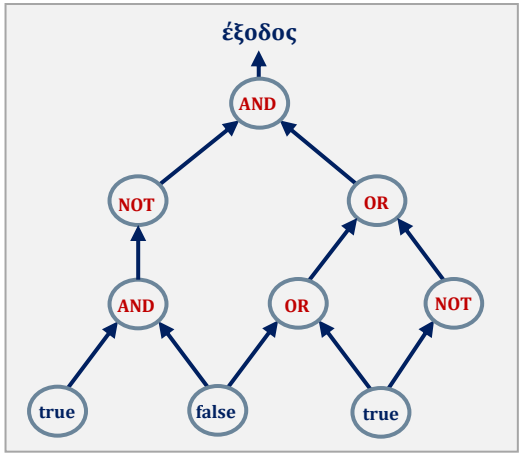
Θέλουμε την αποτίμηση της εξόδου του κυκλώματος σε τιμή true



Αποτίμηση Κυκλώματος

ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ≤ LP

Υπάρχει ένας απλός - αυτόματος τρόπος μετάφρασης αυτού του προβλήματος σε ένα Γραμμικό Πρόβλημα !!!



Δημιουργία μιας μεταβλητής $\mathbf{x_g}$ για κάθε πύλη, μ.τ.π $0 \le x_g \le 1$

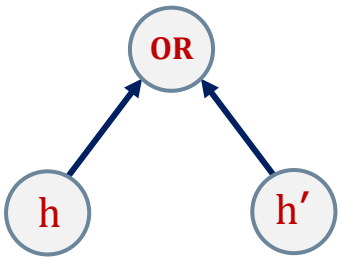
πύλη **g**

$x_g = 1$

g

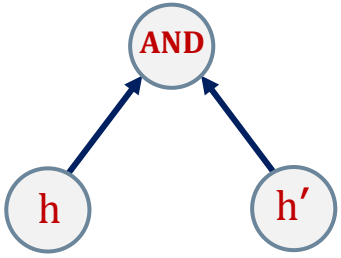
$x_g = 0$

g



$x_g \ge x_h$
$x_g \ge x_{h'}$
$x_g \le x_h + x_{h'}$

g



$x_g \le x_h$
$x_g \le x_{h'}$
$x_g \ge x_h + x_{h'} - 1$

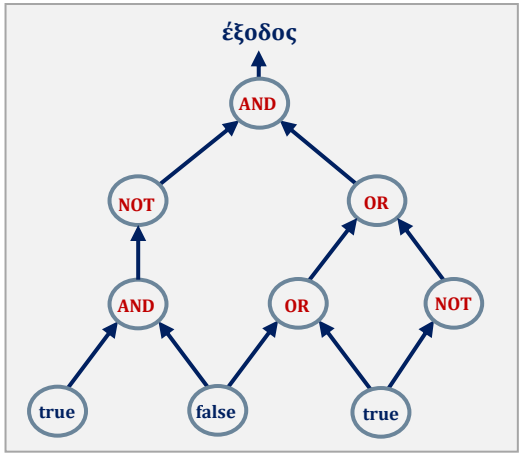
g

$x_g = 1 - x_h$

ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ≤ LP

Οι περιορισμοί αναγκάζουν όλες τις πύλες να πάρουν σωστές τιμές – **0** για **false** και **1** για **true** !!!

Δεν χρειάζεται να μεγιστοποιήσουμε ή ελαχιστ/σουμε κάτι… απλώς να πάρουμε την τιμή της μεταβλητής x_o ⇔ πύλη εξόδου !!!



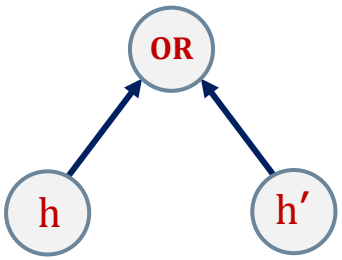
πύλη **g**

$x_g = 1$

g

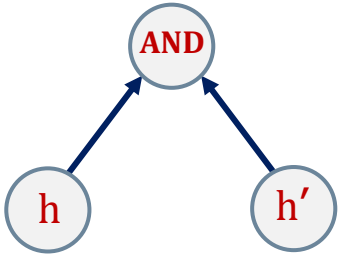
$x_g = 0$

g



$x_g \geq x_h$
$x_g \geq x_{h'}$
$x_g \leq x_h + x_{h'}$

g



$x_g \leq x_h$
$x_g \leq x_{h'}$
$x_g \geq x_h + x_{h'} - 1$

g

$x_g = 1 - x_h$

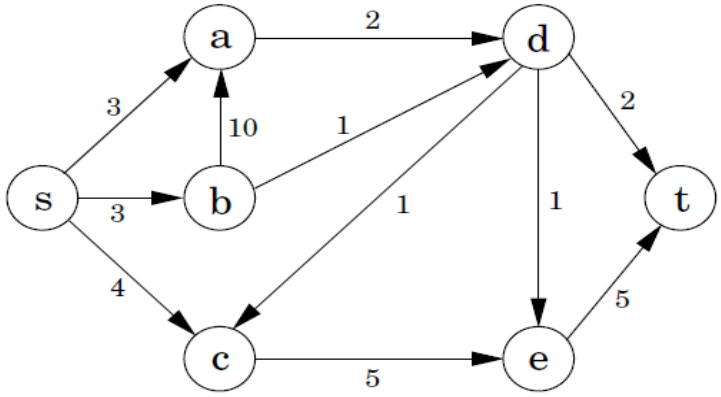
Επίλυση Προβλημάτων

Επίλυση Γρίφων

Απόδειξη Κάτω Φραγμάτων

Χρήση Αλγοριθμικών Τεχνικών

Επινόηση Νέων Αλγορίθμων



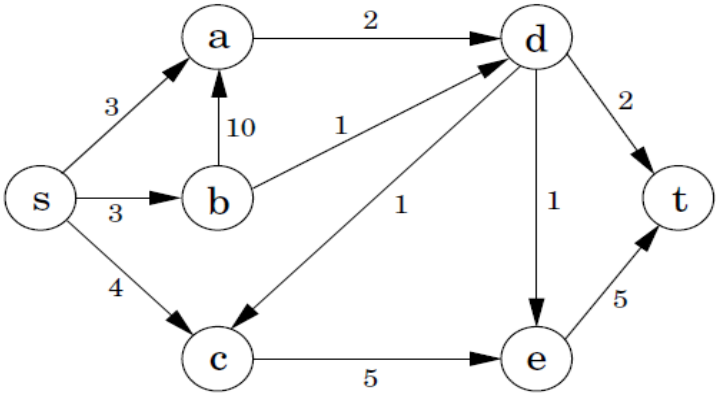
Ροές σε Δίκτυα

Αλγόριθμος των Ford-Fulkerson

5 Ροές σε Δίκτυα

Πρόβλημα

Μετάφερε όσο το δυνατόν περισσότερο πετρέλαιο $\mathbf{s \to t}$ χωρίς να υπερβείς τη *χωρητικότητα* $\boldsymbol{c_e > 0}$ καμιάς ακμής !!!



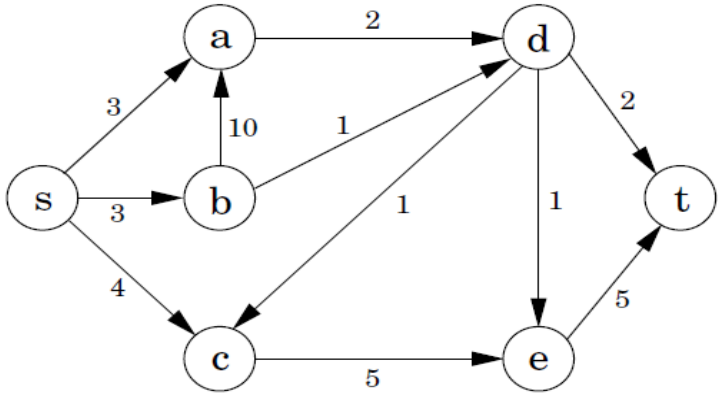
Λύση

Με αναγωγή σε πρόβλημα Γραμμικού Προγραμματισμού (LP) !!!

- Αναθέσαμε τιμές ροών $\boldsymbol{f_e}$, $\forall$ e $\in$ E, οι οποίες:
  - ✓ *ικανοποιούν* ένα σύνολο γραμμικών περιορισμών και
  - ✓ *μεγιστοποιούν* μια γραμμική αντικειμενική συνάρτηση.

Μεγιστη-Ροη ≤ LP

- Στο δίκτυο του σχήματος, το LP ζητά να μεγιστοποιήσει την παράσταση:



$$\max \; f_{sa} + f_{sb} + f_{sc}$$

με 27 περιορισμούς:

- ✓ **11** για *μη-αρνητικότητα* (όπως $f_{sa} \geq 0$)
- ✓ **11** για *χωρητικότητα* (όπως $f_{sa} \leq 3$)
- ✓ **5** για *διατήρηση της ροής* σε κάθε κόμβο $u \in V\text{-}\{s, t\}$ (όπως $f_{sc} + f_{dc} = f_{ce}$)

- Ο αλγόριθμος Simplex επιλύει το πρόβλημα **πολύ γρήγορα** στην πράξη ! δίνοντας ροή 7 που είναι πράγματι βέλτιστη !!!

Μεγιστη-Ροη ≤ LP

- Με την αναγωγή που δώσαμε, τι **πετύχαμε** ?

  Πετύχαμε να λύσουμε **«αποτελεσματικά»** το πρόβλημα της μεγίστης ροής με LP !!!

  Αποκτήσαμε «**στοιχεία**» και «**έμπνευση**» για το σχεδιασμό *άμεσων αλγορίθμων* (direct max-flow algorithms) !!!

- Ας έρθουμε να σχεδιάσουμε έναν **άμεσο αλγόριθμο** για το πρόβλημα της μεγίστης ροής (direct max-flow algorithms) !!!

Πρώτες Παρατηρήσεις !!!

- Πως συμπεριφέρεται (διαισθητικά) ο αλγόριθμος Simplex ?
  - ✓ κάνει τοπικές κινήσεις στην επιφάνεια μιας κυρτής εφικτής περιοχής,
  - ✓ βελτιώνοντας διαδοχικά την αντικειμενική συνάρτηση,
  - ✓ μέχρι τελικά να φθάσει στη βέλτιστη λύση!

- Στοιχειώδης ερμηνεία της συμπεριφοράς του Simplex για τον υπολογισμό της max Ροής :

Ξεκινάει αρχικά με μηδενική ροή !

Επανάληψη: επιλέγει κατάλληλη διαδρομή **P** : s → t, και *αυξάνει* τη ροή κατά μήκος των ακμών της **P** όσο το δυνατόν περισσότερο !

Ιδιότητες Ροών

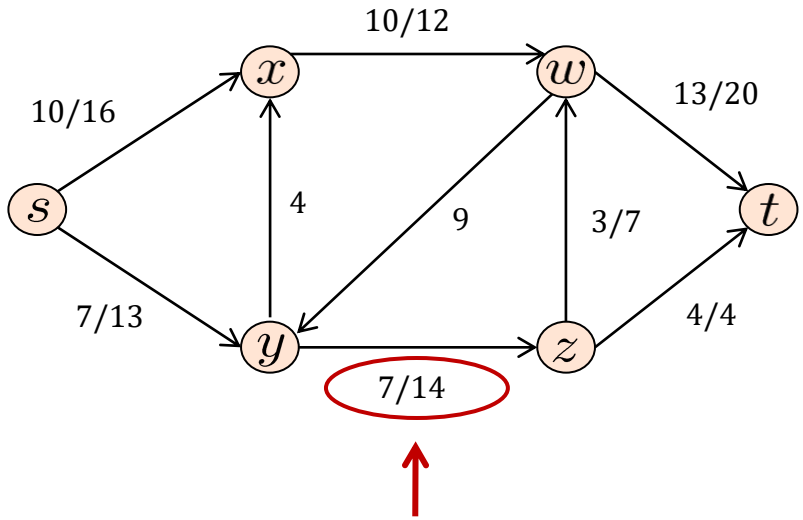
Κατευθυνόμενο γράφημα

$$G = (V, E)$$

Συνάρτηση χωρητικότητας $c : E \to \mathbf{R}$

$$c(u, v) \geq 0, (u, v) \in E$$

$$c(u, v) = 0, (u, v) \notin E$$

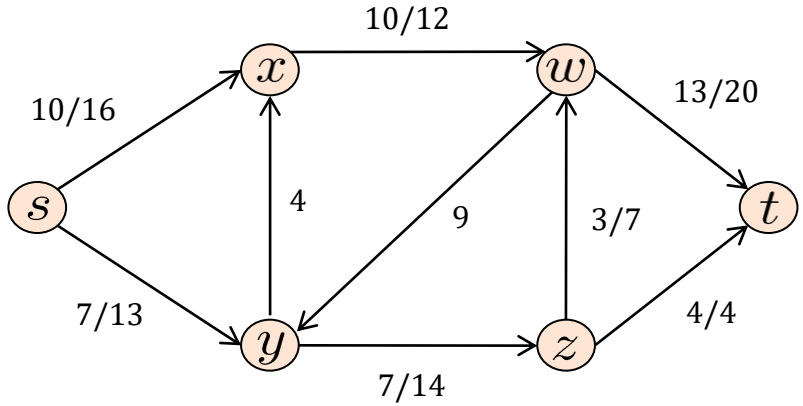


Συμβολισμός **a/b**

Η ακμή $e = (y, z)$ του δικτύου έχει:

- ✓ **a : ροή** $f_e = f(y, z) = 7$
- ✓ **b : χωρητικότητα** $c_e = c(y, z) = 14$

Ιδιότητες Ροών



Κατευθυνόμενο γράφημα

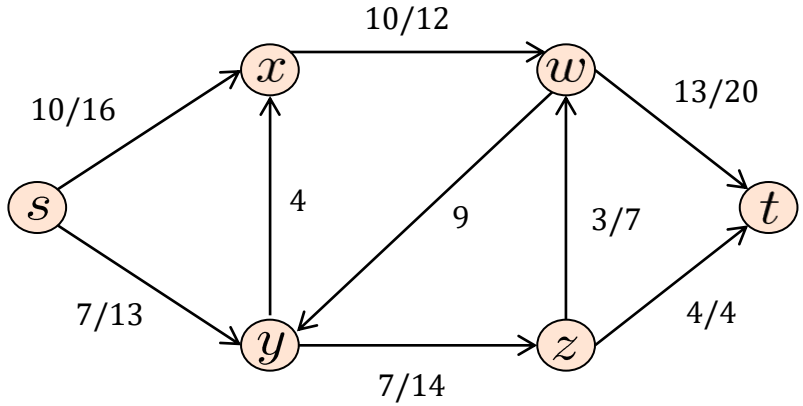
$$G = (V, E)$$

Ροή δικτύου:

Συνάρτηση $f : V \times V \rightarrow \mathbf{R}$ με τις ακόλουθες ιδιότητες:

- **Περιορισμός χωρητικότητας**: $f(u, v) \leq c(u, v)$ για κάθε ζεύγος $u, v \in V$
- **Αντισυμμετρία**: $f(u, v) = -f(v, u)$ για κάθε ζεύγος $u, v \in V$
- **Διατήρηση ροής**: $\sum_{v \in V, f(v,u)>0} f(v, u) - \sum_{v \in V, f(u,v)>0} f(u, v) = 0, u \in V - \{s, t\}$

Ιδιότητες Ροών



Κατευθυνόμενο γράφημα

$$G = (V, E)$$

Τιμή ροής

$$|f| = \sum_{v \in V} f(s, v)$$

Συνολική εκροή από τον κόμβο αφετηρίας *s*

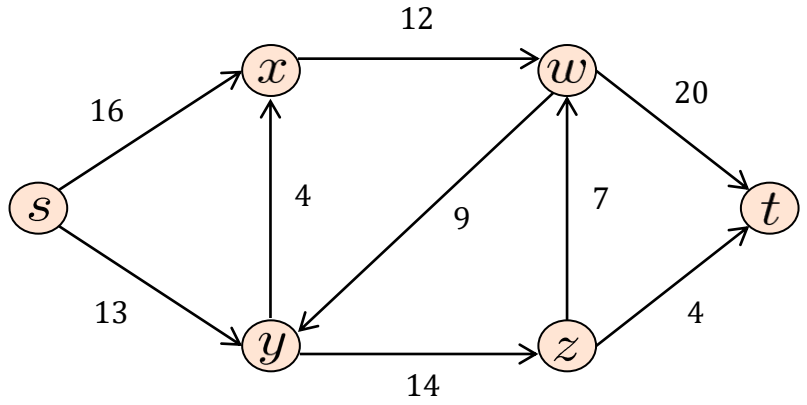
Π.χ. $|f| = f(s, x) + f(s, y) = 10 + 7 = 17$

Αλγόριθμος Μέγιστης Ροής

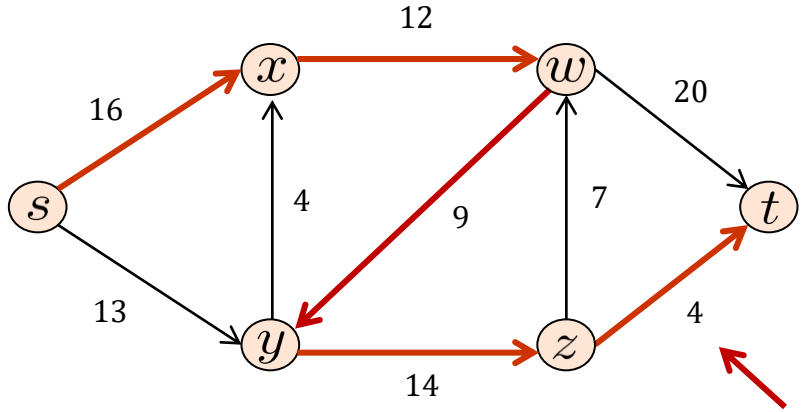
Θέλουμε να βρούμε μια ροή σε ένα δίκτυο με **Μέγιστη Τιμή !!!**

Στοιχεία Αλγόριθμου

- **Αυξητικές Διαδρομές**
- **Υπολοιπόμενα Δίκτυα**
- **Τομές**



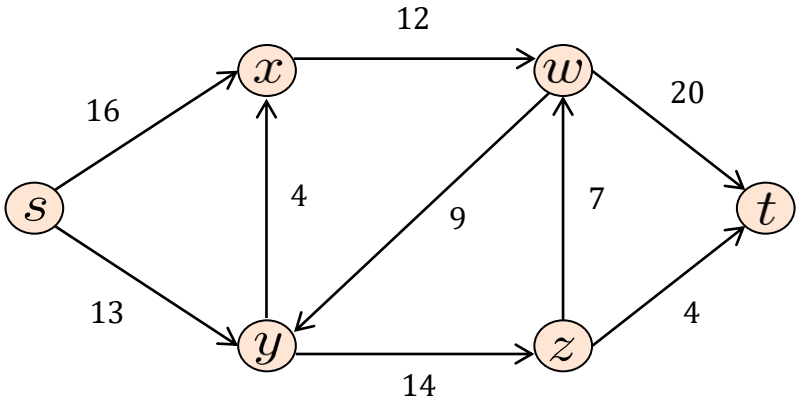
Αυξητική Διαδρομή **P** : s → t

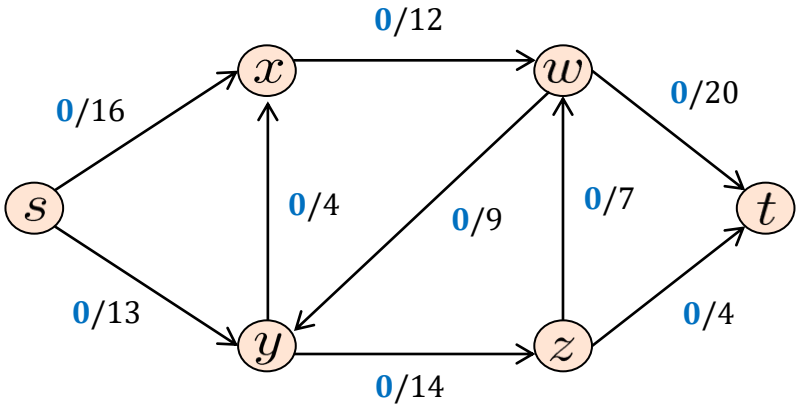


Υπολειπόμενη χωρητικότητα
αυξητικής διαδρομής :

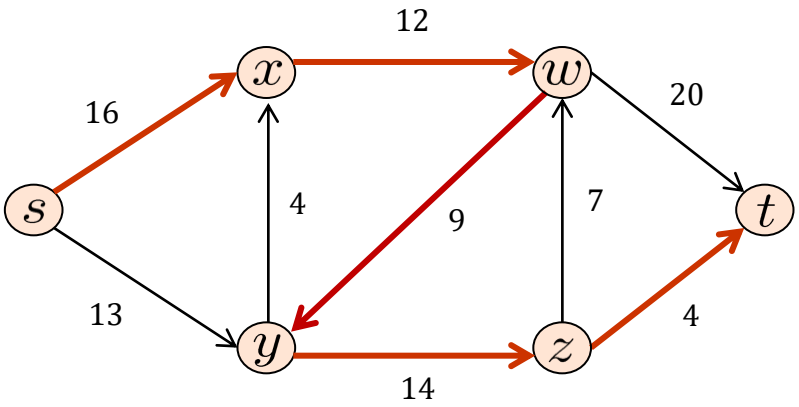
$$c_f(p) = \min\{c_f(u,v) \mid (u,v) \in p\}$$

Σκιαγράφηση Αλγόριθμου



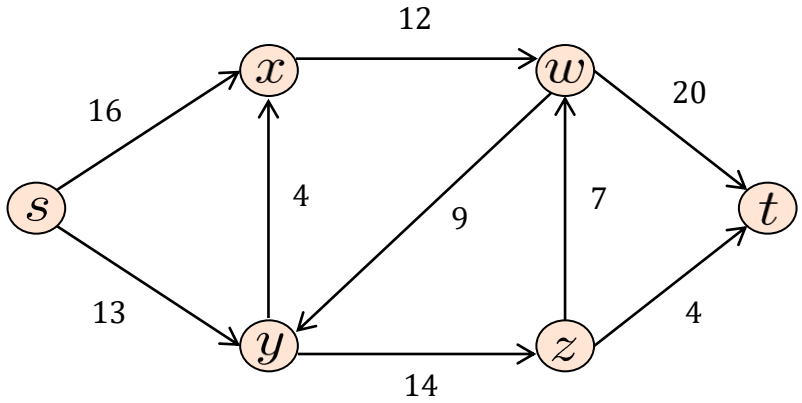


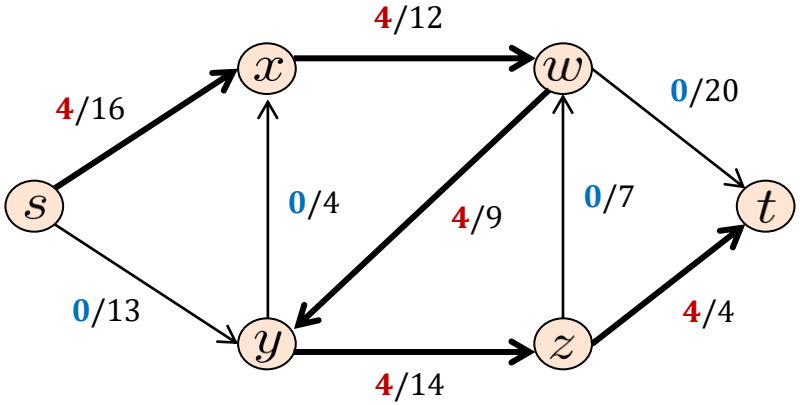
Μηδενική ροή



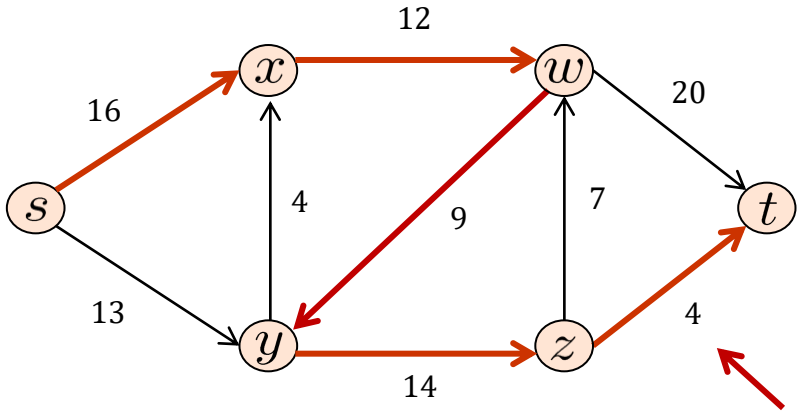
Αυξητική Διαδρομή **P** : s → t

Σκιαγράφηση Αλγόριθμου



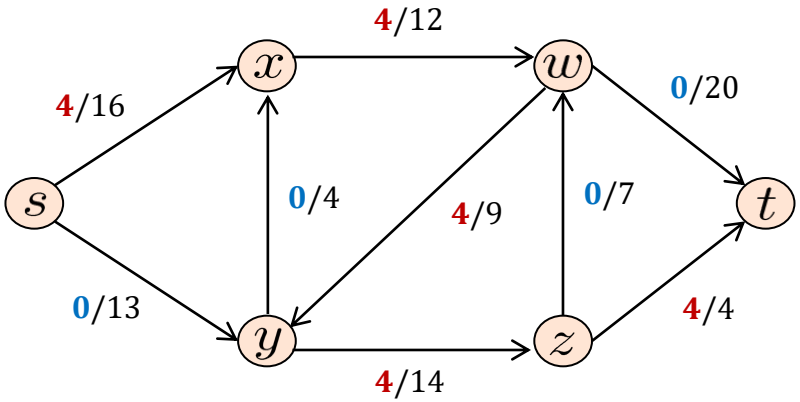


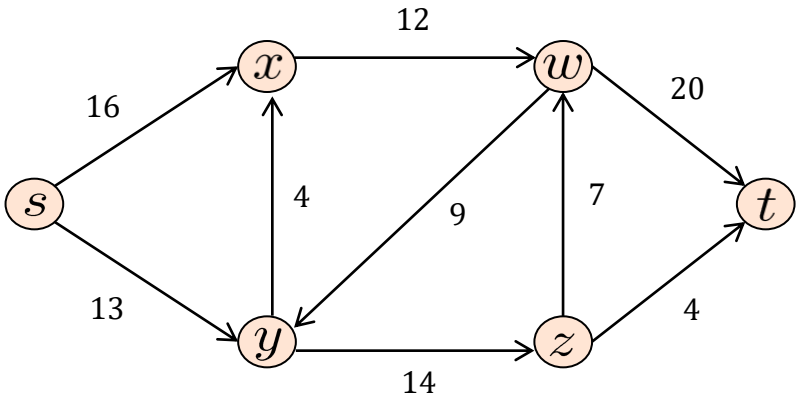
Αύξηση της ροής f
κατά 4 μονάδες στις ακμές της P



Αυξητική Διαδρομή **P** : s → t

Σκιαγράφηση Αλγόριθμου

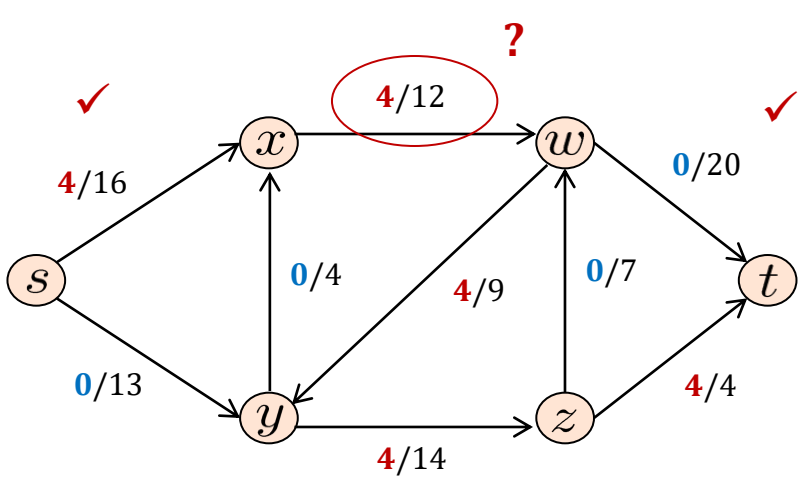




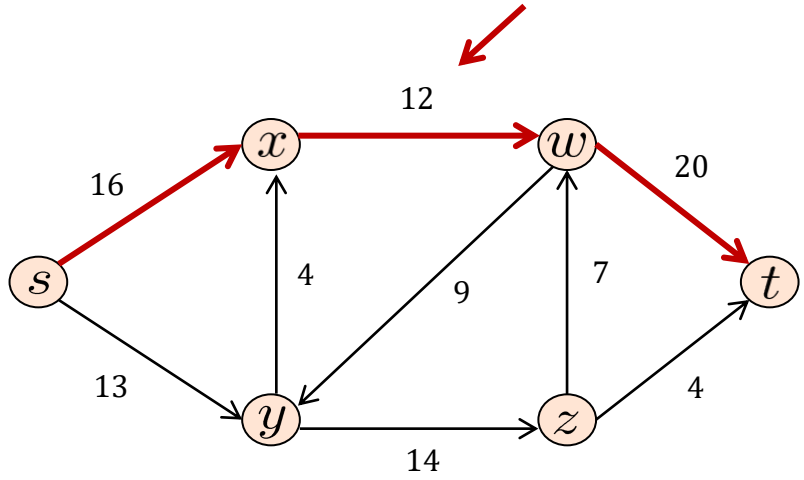
Πως συνεχίζω ?

Επιλέγω αυξητική διαδρομή **P** : s → t

Σκιαγράφηση Αλγόριθμου

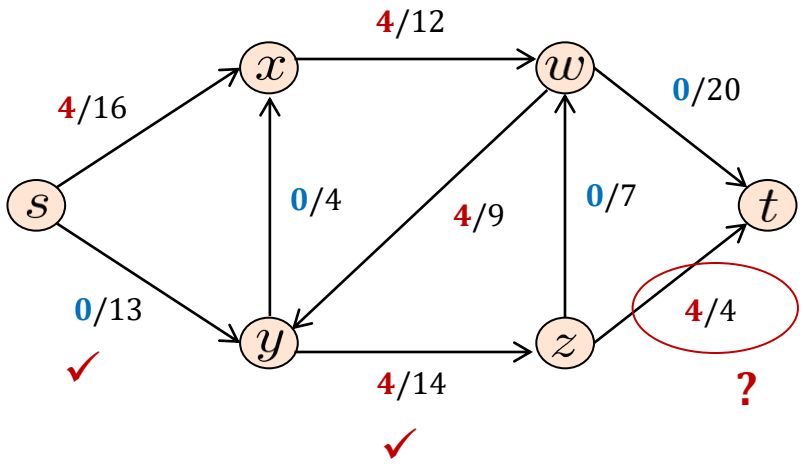


Αύξηση της ροής f
κατά **12** μονάδες στις ακμές της P

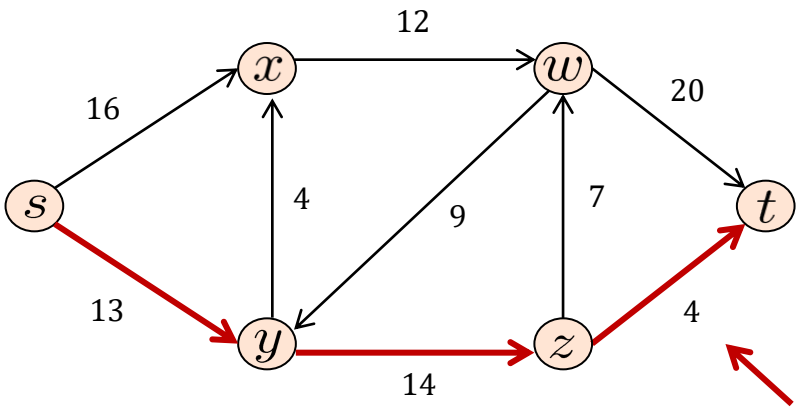


Εάν επιλέξω την **P** = (s, x, w, t) **?**

Σκιαγράφηση Αλγόριθμου

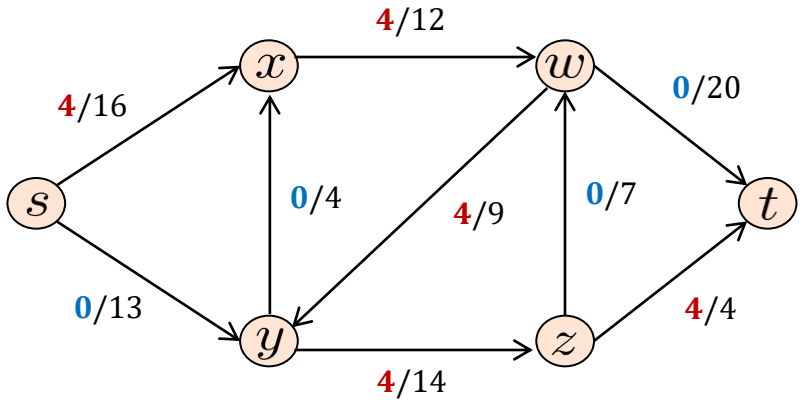


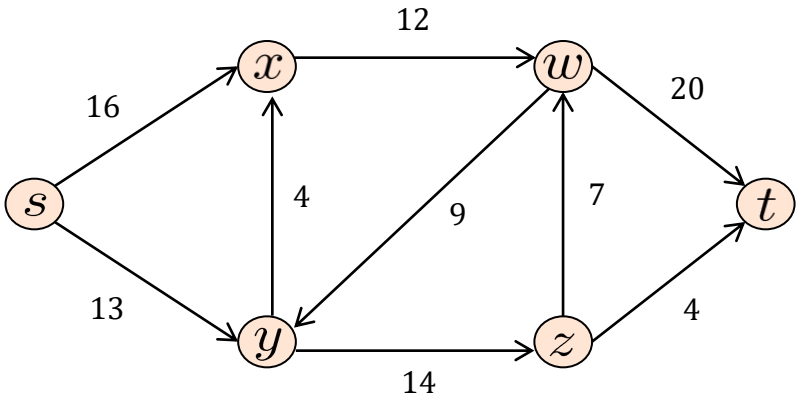
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

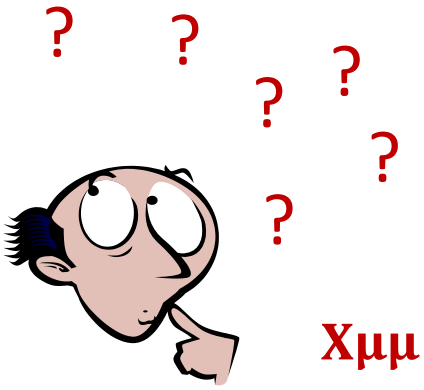


Εάν επιλέξω την **P** = (s, y, z, t) **?**

Σκιαγράφηση Αλγόριθμου

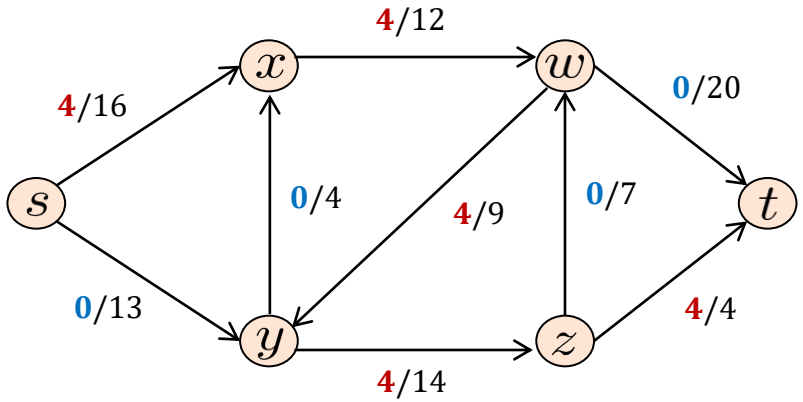


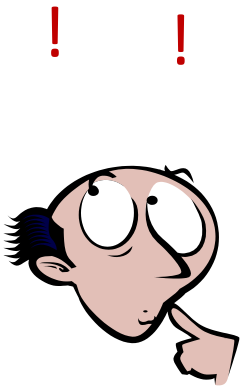




Χμμ !!!... Πως συνεχίσω ?

Σκιαγράφηση Αλγόριθμου



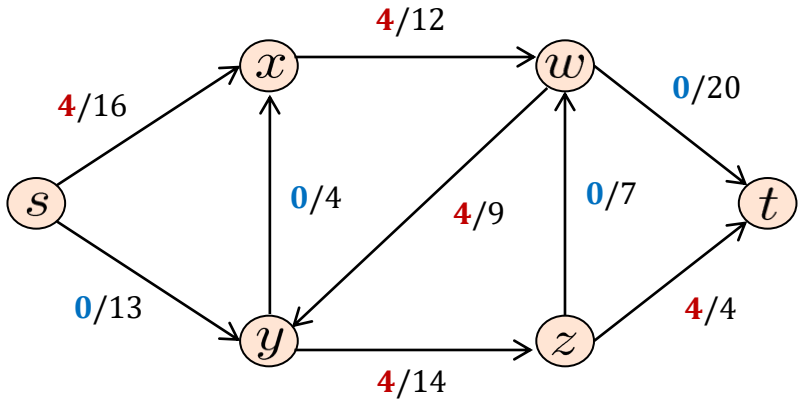


Ιδέα !!!

Ορίζουμε δύο τύπους ακμών:

1. Η ακμή (u, v) υπάρχει στο δίκτυο, και έχει ακόμα υπολειπόμενη χωρητικότητα !
2. Η αντίστροφη ακμή (v, u) δεν υπάρχει στο αρχικό δίκτυο, και υπάρχει κάποια ροή σε αυτή !!

Σκιαγράφηση Αλγόριθμου



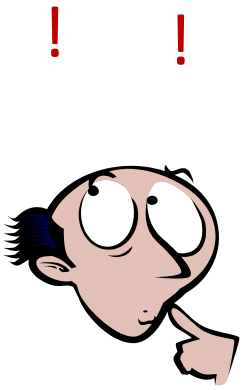
Εάν η τρέχουσα ροή είναι *f*, τότε:

1. Η ακμή (u, v) υπάρχει στο δίκτυο, και έχει ακόμα υπολειπόμενη χωρητικότητα !

$$c_f(u, v) = c(u, v) - f(u, v)$$

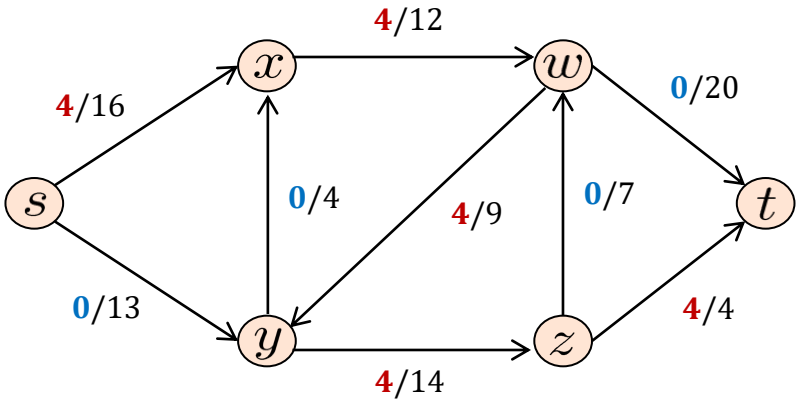
2. Η αντίστροφη ακμή (v, u) δεν υπάρχει στο δίκτυο, και υπάρχει κάποια ροή σε αυτή !!

$$f(u, v)$$



Ιδέα !!!

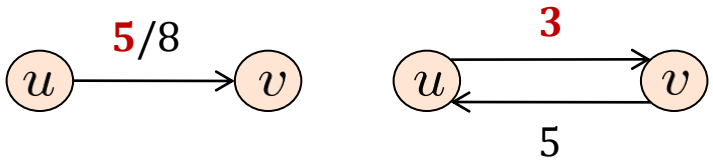
Σκιαγράφηση Αλγόριθμου

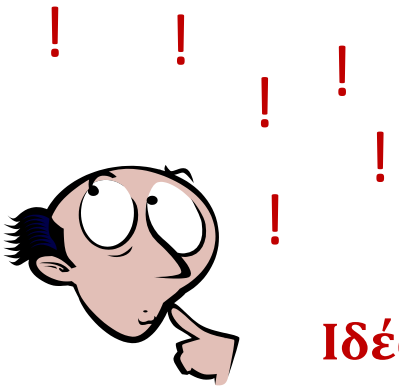


Παράδειγμα:

- ✓ Υπολειπόμενη χωρητικότητα

$$c_f(u,v) = c(u,v) - f(u,v)$$

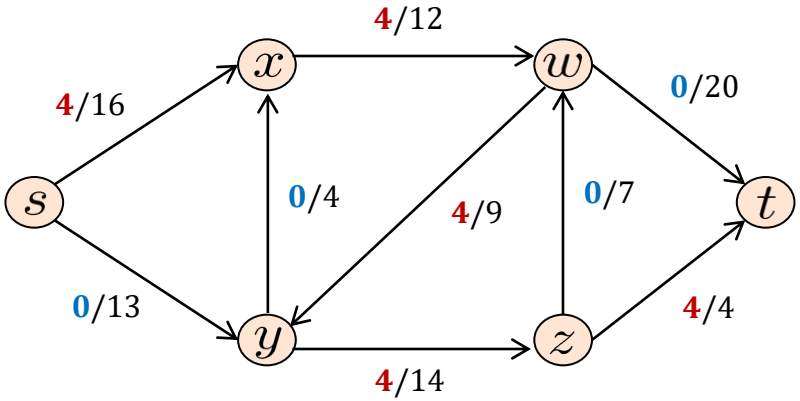




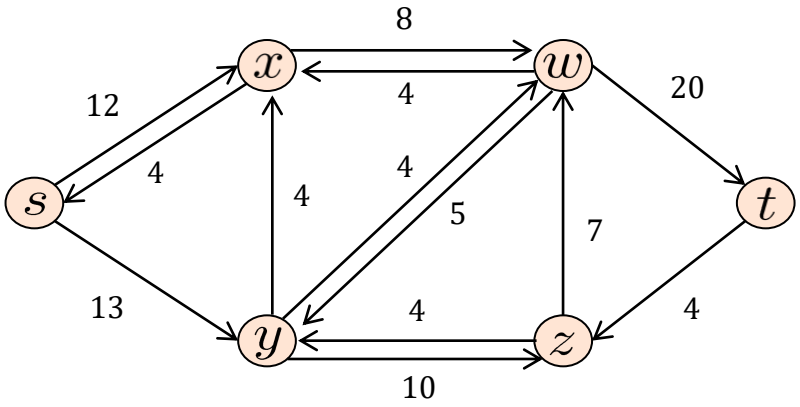
Ιδέα !!!

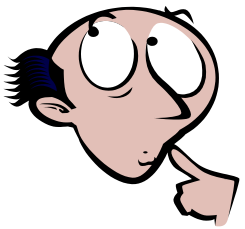
$$c_f(u,v) = 8 - 5 = 3,\ \ c_f(v,u) = 0 - (-5) = 5$$

Σκιαγράφηση Αλγόριθμου



G_f





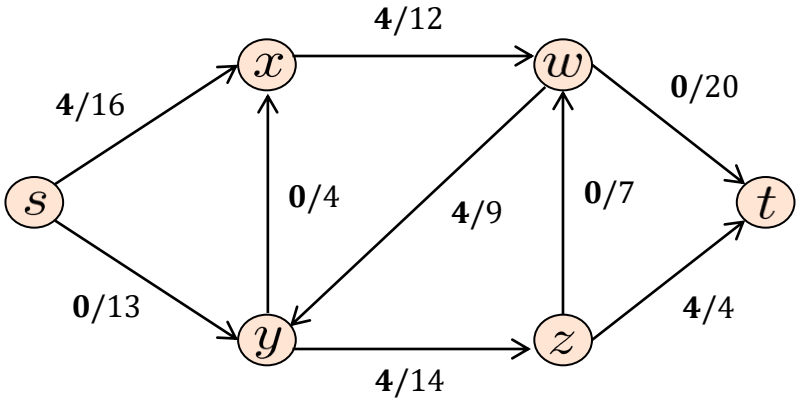
! ! ! ! ! !

Τώρα ξέρω πως να συνεχίζω !!!

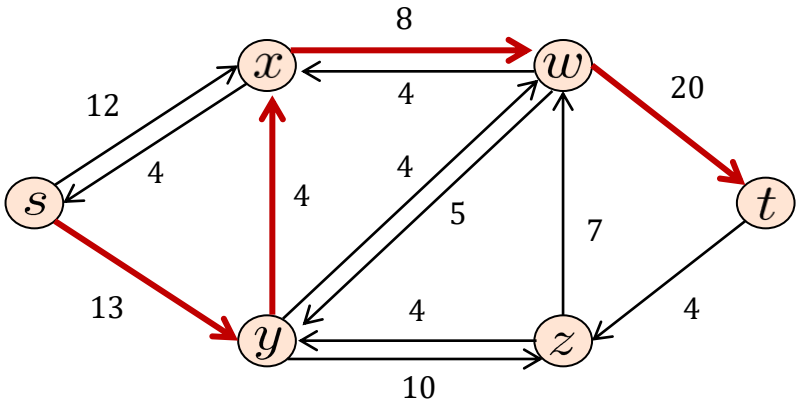
Δίκτυο Υπολοίπων

ή υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



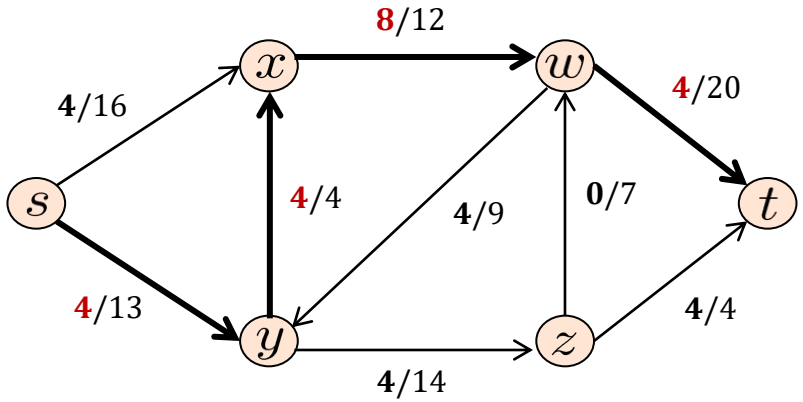
G_f



Επιλέγω αυξητική διαδρομή **P** : s → t

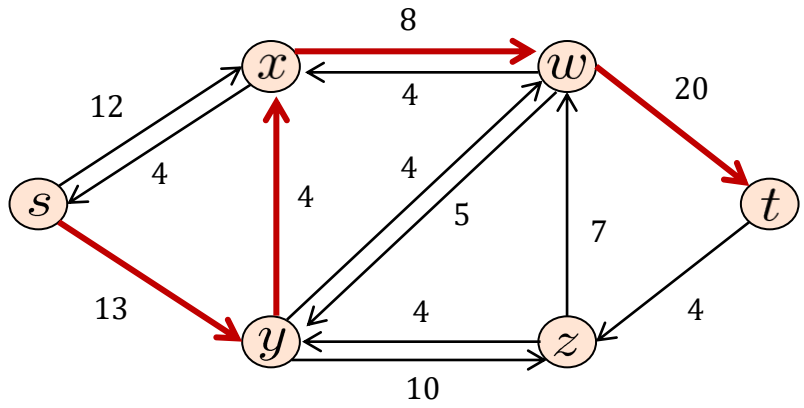
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



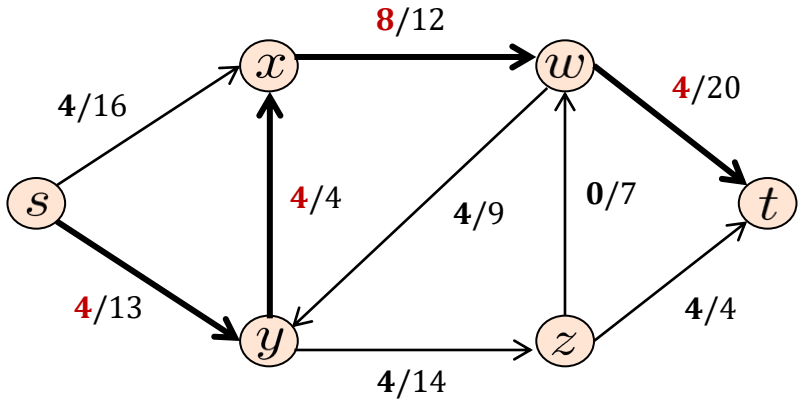
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

G_f



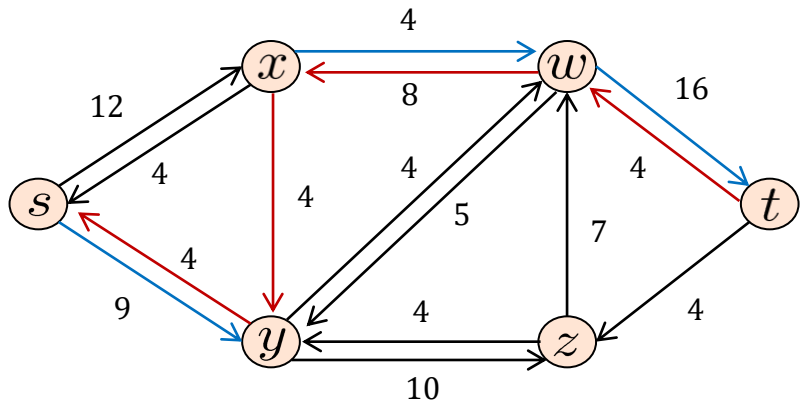
Επιλέγω αυξητική διαδρομή **P** : s → t
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



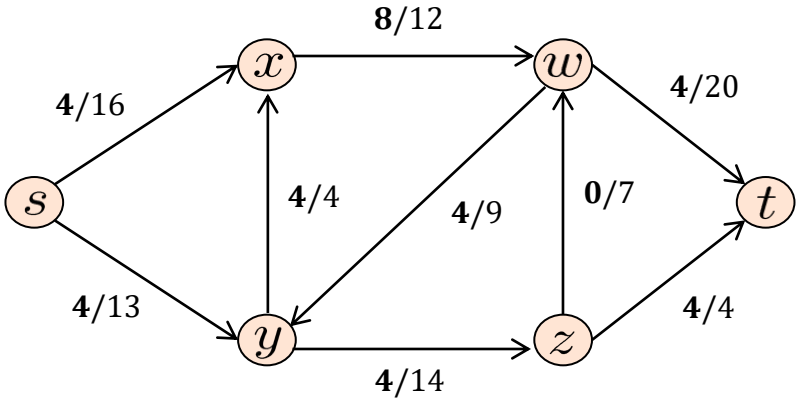
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

G_f

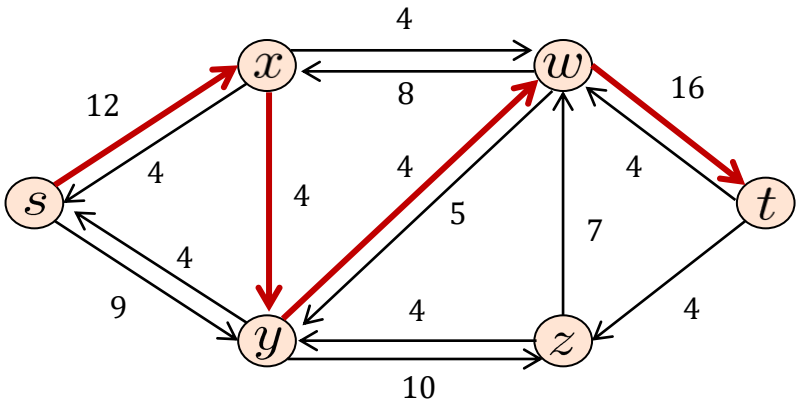


Ενημέρωση
υπολειπόμενου δικτύου G_f

Σκιαγράφηση Αλγόριθμου



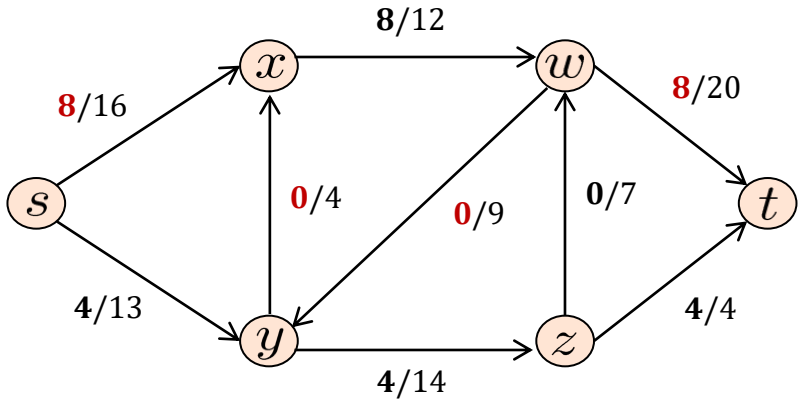
G_f



Επιλέγω αυξητική διαδρομή **P** : s → t

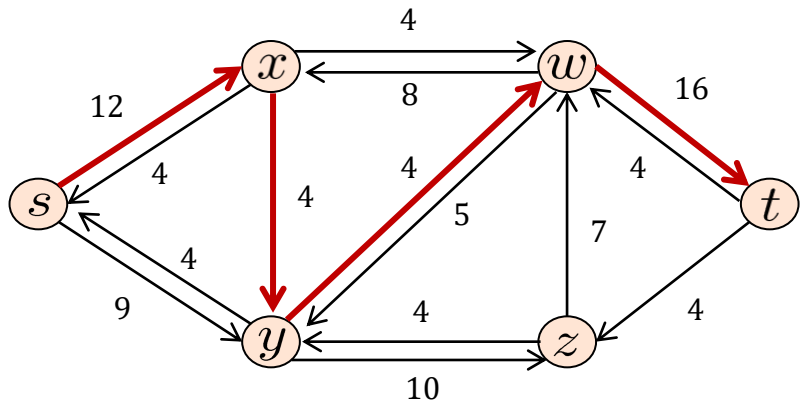
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



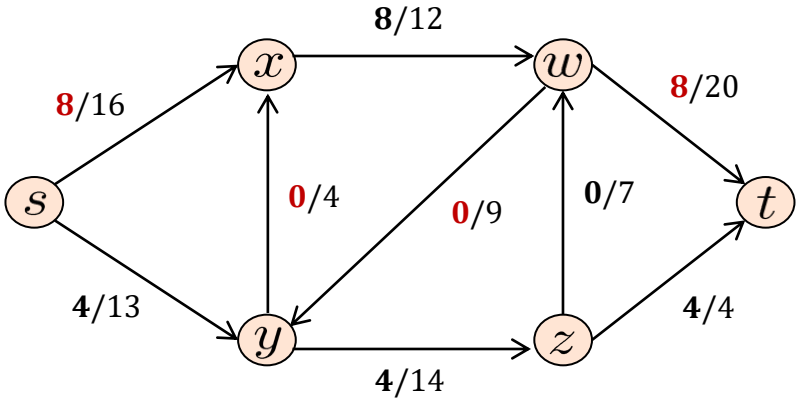
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

G_f



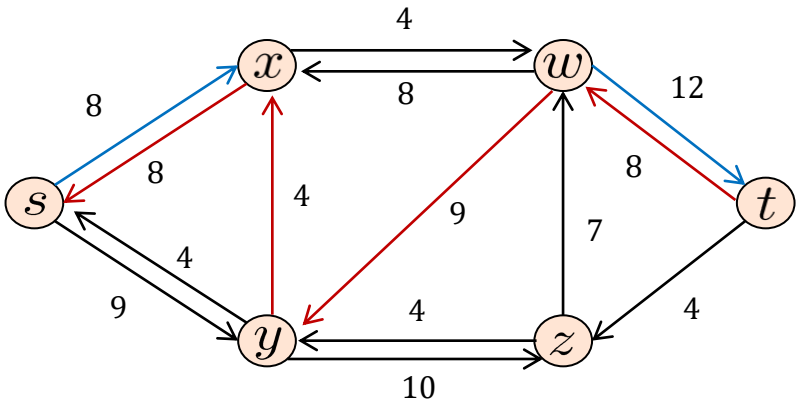
Επιλέγω αυξητική διαδρομή **P** : s → t
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



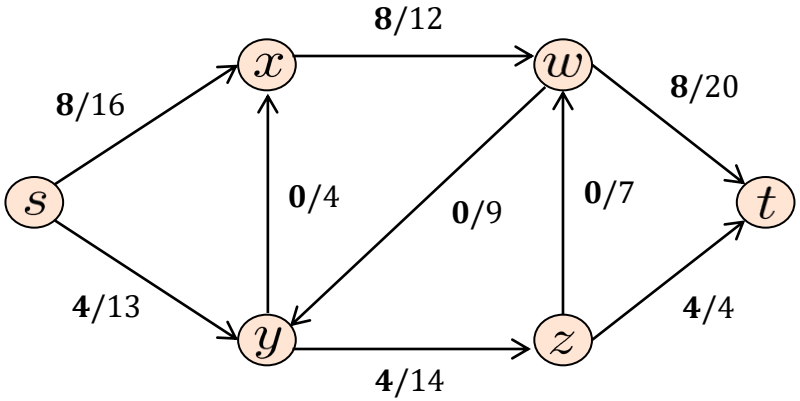
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

G_f

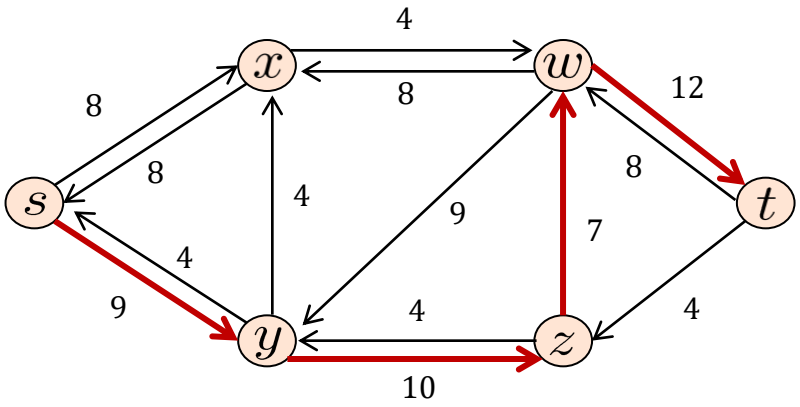


Ενημέρωση
υπολειπόμενου δικτύου G_f

Σκιαγράφηση Αλγόριθμου



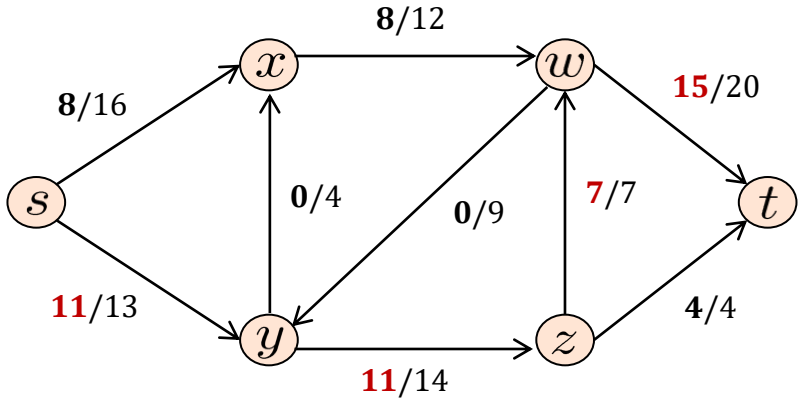
G_f



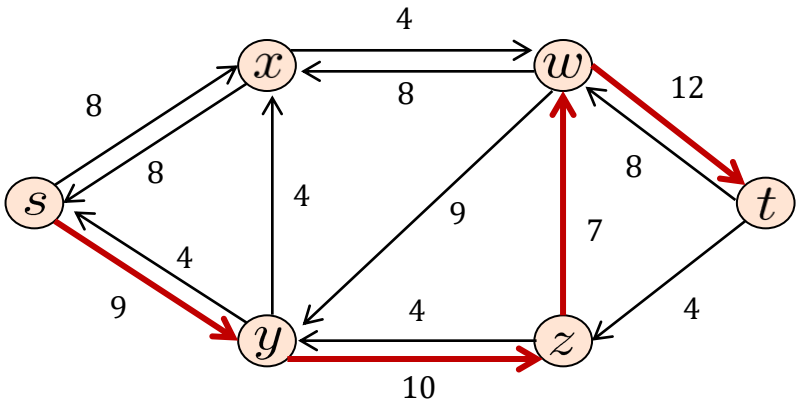
Επιλέγω αυξητική διαδρομή **P** : s → t

στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



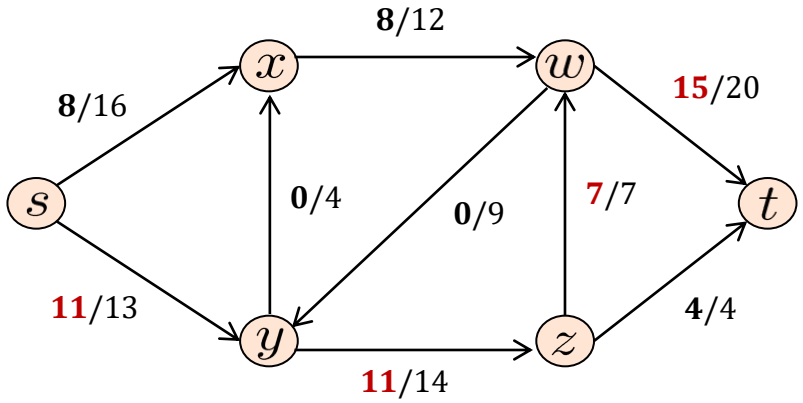
G_f



Αύξηση της ροής f
κατά **7** μονάδες στις ακμές της P

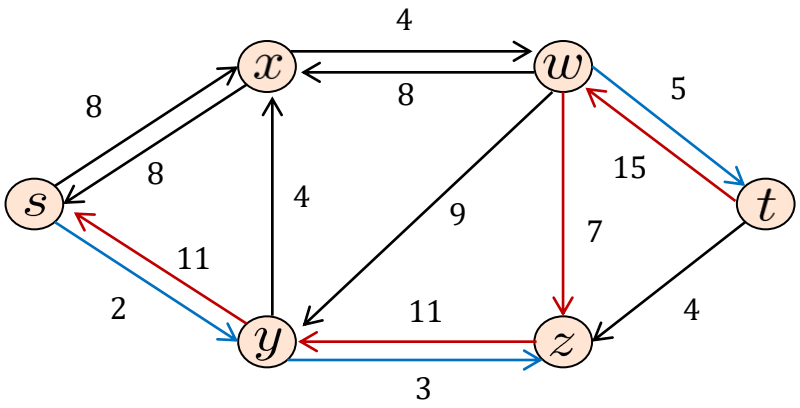
Επιλέγω αυξητική διαδρομή **P** : s → t
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



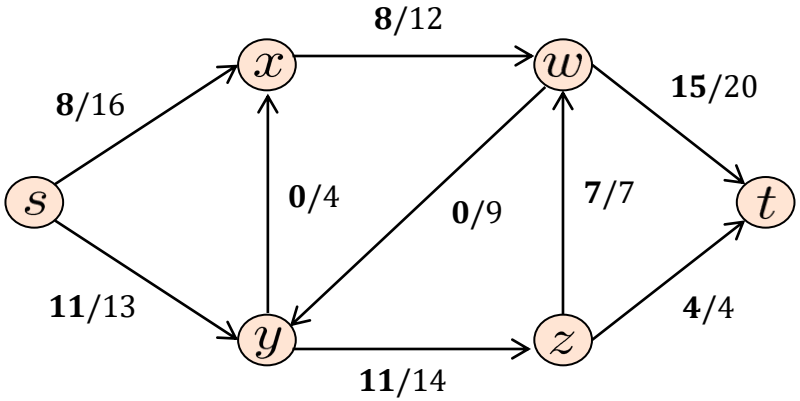
Αύξηση της ροής f
κατά **7** μονάδες στις ακμές της P

G_f

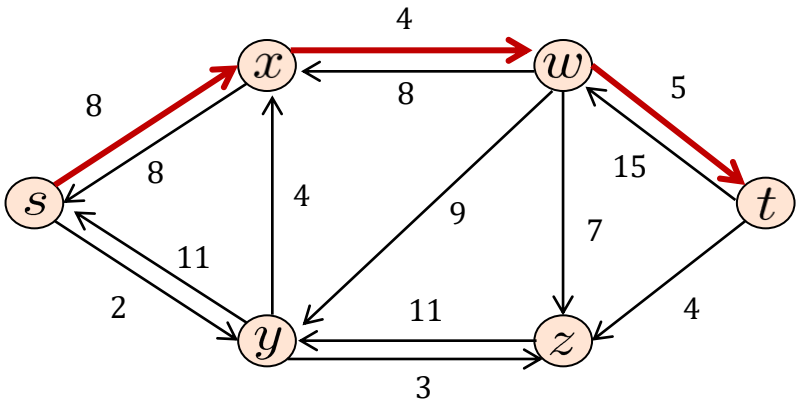


Ενημέρωση
υπολειπόμενου δικτύου G_f

Σκιαγράφηση Αλγόριθμου



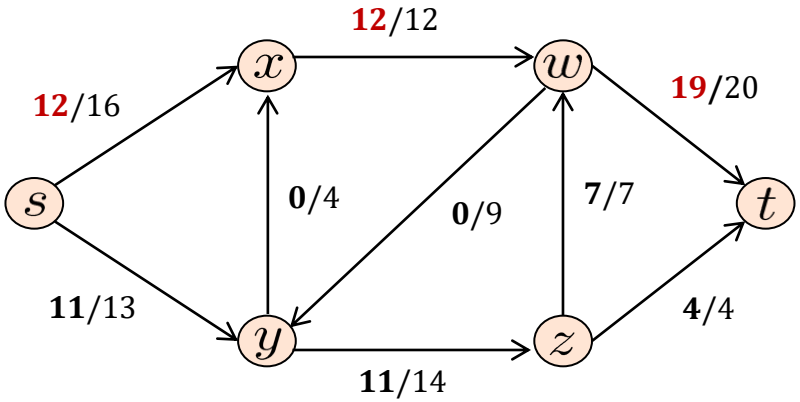
G_f



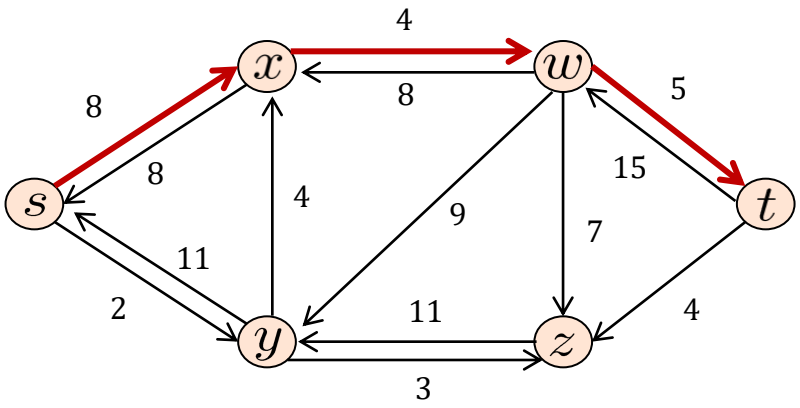
Επιλέγω αυξητική διαδρομή **P** : s → t

στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



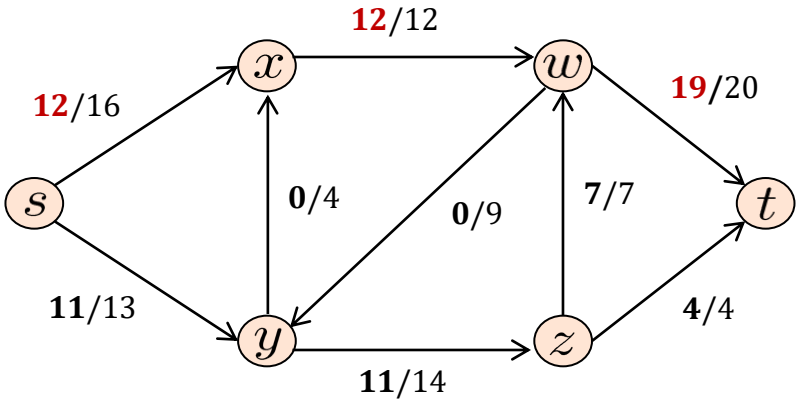
G_f



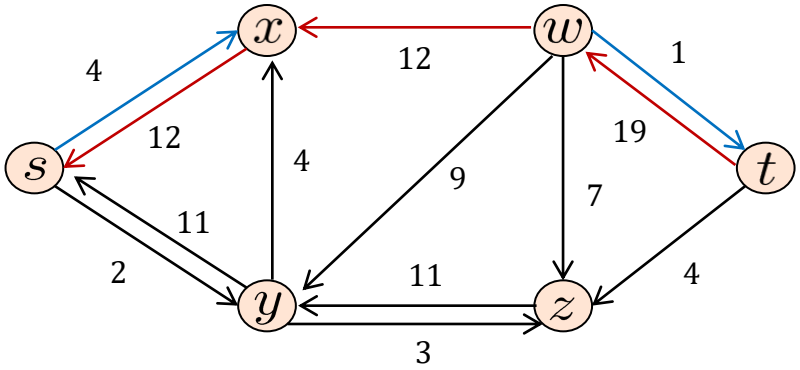
Αύξηση της ροής f
κατά **4** μονάδες στις ακμές της P

Επιλέγω αυξητική διαδρομή **P** : s → t
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου

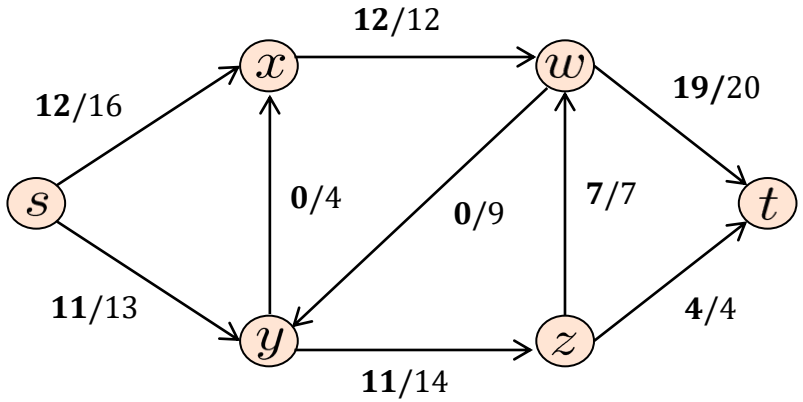


G_f

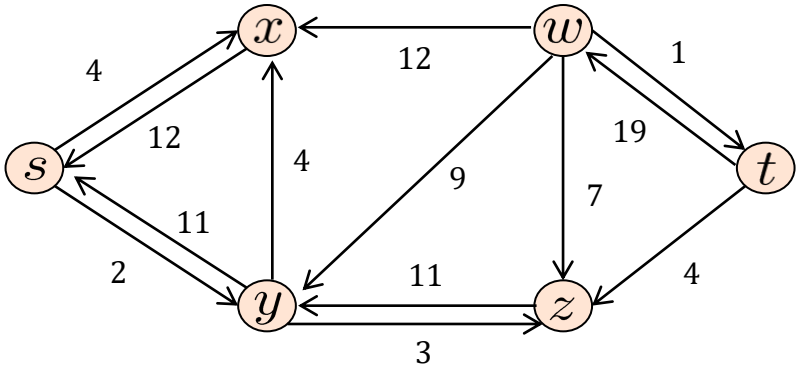


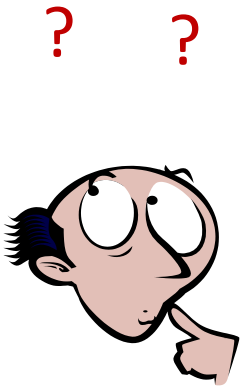
Ενημέρωση
υπολειπόμενου δικτύου G_f

Σκιαγράφηση Αλγόριθμου



G_f





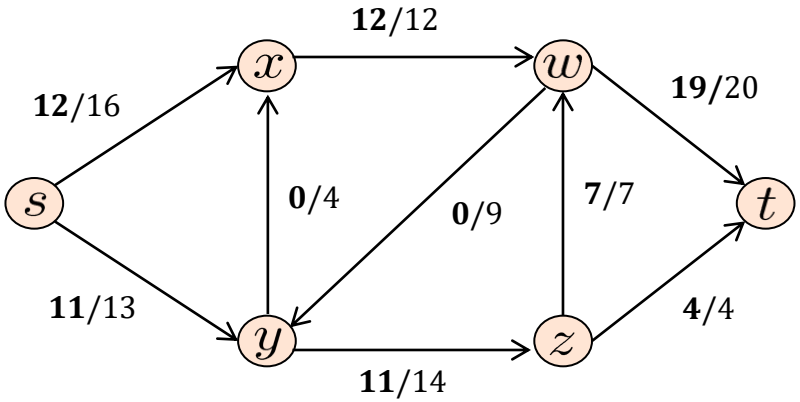
? ? ? ? ? ?

Δεν υπάρχει !!!

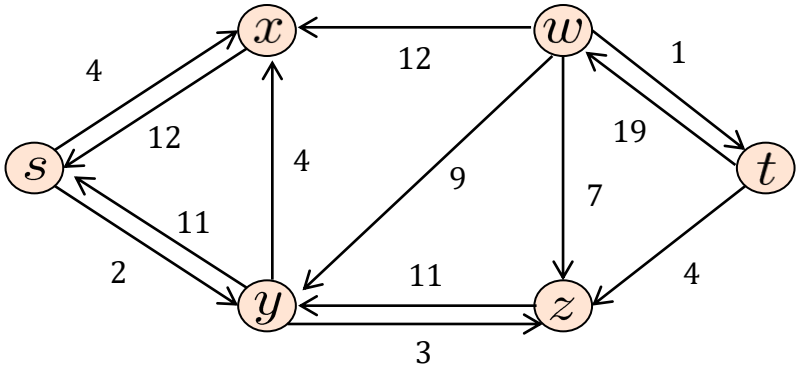
Επιλέγω αυξητική διαδρομή **P** : s → t

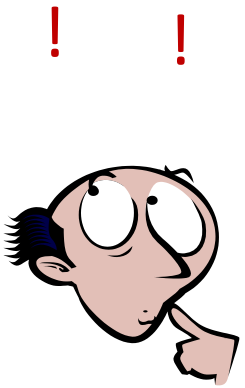
στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



G_f



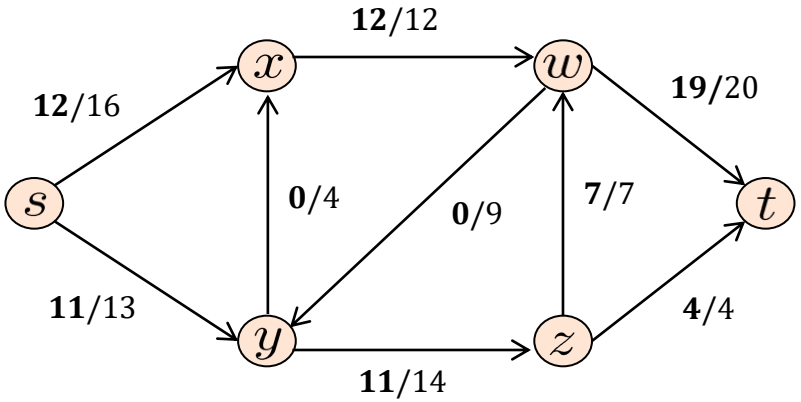


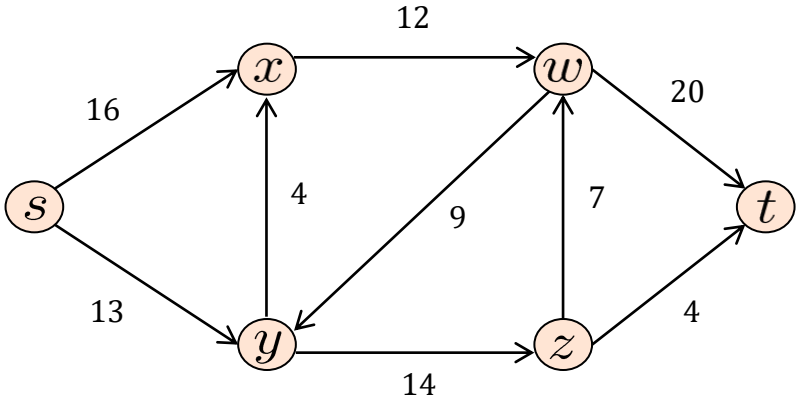
! ! ! ! ! !

Έχω την λύση !!!

Επιλέγω αυξητική διαδρομή **P** : s → t στο υπολειπόμενο δίκτυο G_f

Σκιαγράφηση Αλγόριθμου



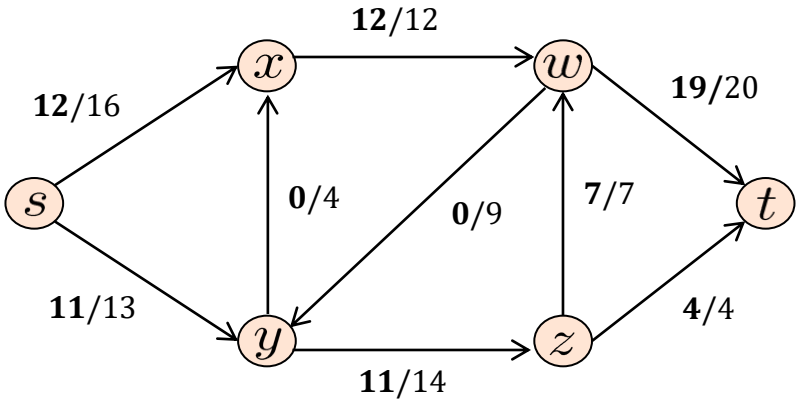


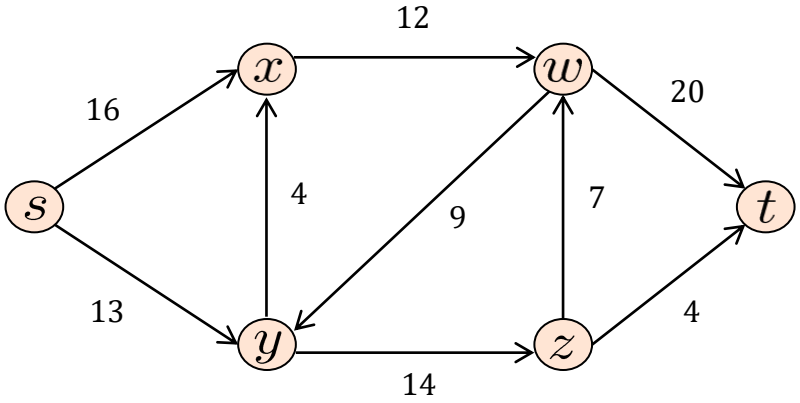
Αρχικό δίκτυο $G = (V, E)$

Τιμή της **μέγιστης ροής** του δικτύου:

$$|f| = \sum_{v \in V} f(s, v) \text{ = } \mathbf{23}$$

Σκιαγράφηση Αλγόριθμου





Αρχικό δίκτυο $G = (V, E)$

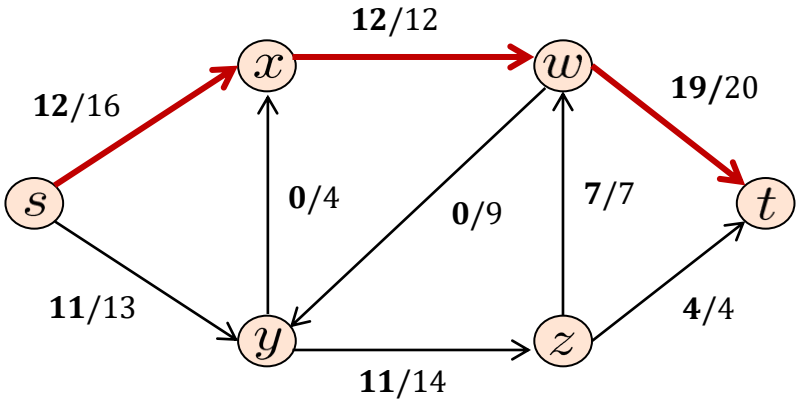
Τιμή της **μέγιστης ροής** του δικτύου:

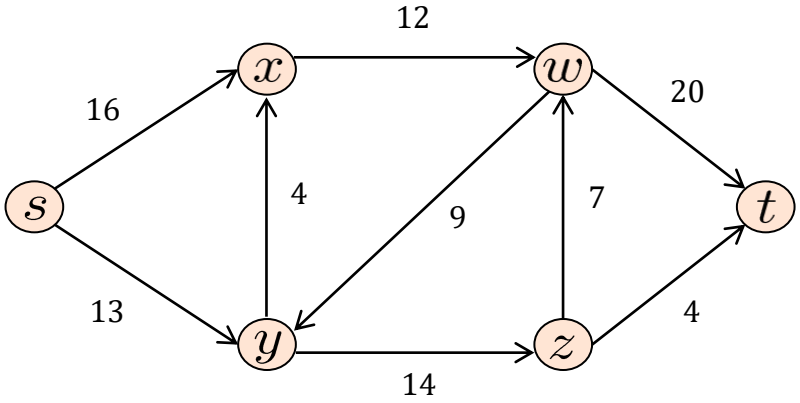
$$|f| = \sum_{v \in V} f(s, v) \quad \textbf{= 23}$$

Επαλήθευση…

Υπολοιπόμενες χωρητικότητες:
4, 4, 4, 7, 4

Σκιαγράφηση Αλγόριθμου





Αρχικό δίκτυο $G = (V, E)$

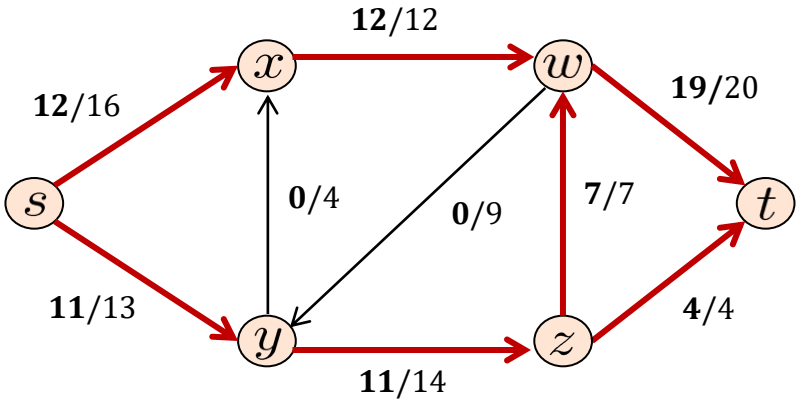
Τιμή της **μέγιστης ροής** του δικτύου:

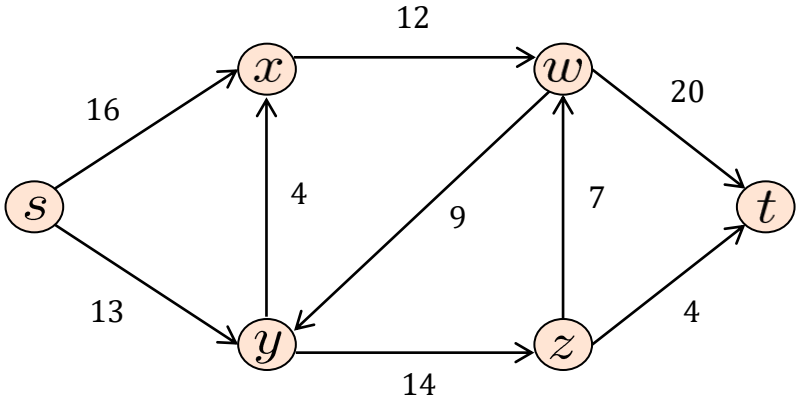
$$|f| = \sum_{v \in V} f(s, v) \text{ = 23}$$

Επαλήθευση…

Υπολοιπόμενες χωρητικότητες:
4, 4, 4, 7, 4

Σκιαγράφηση Αλγόριθμου





Αρχικό δίκτυο $G = (V, E)$

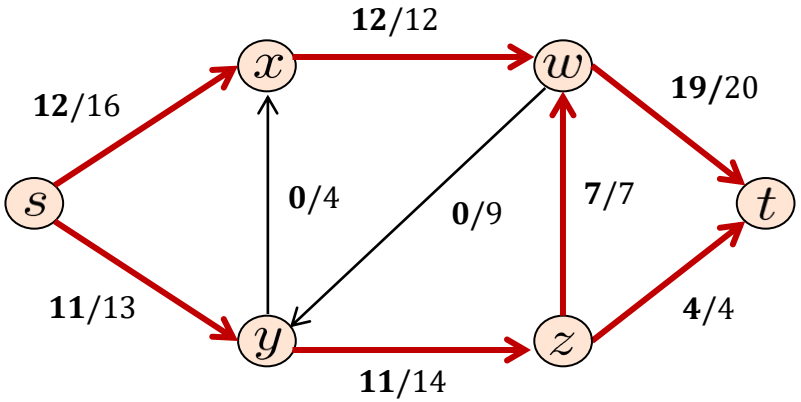
Τιμή της **μέγιστης ροής** του δικτύου:

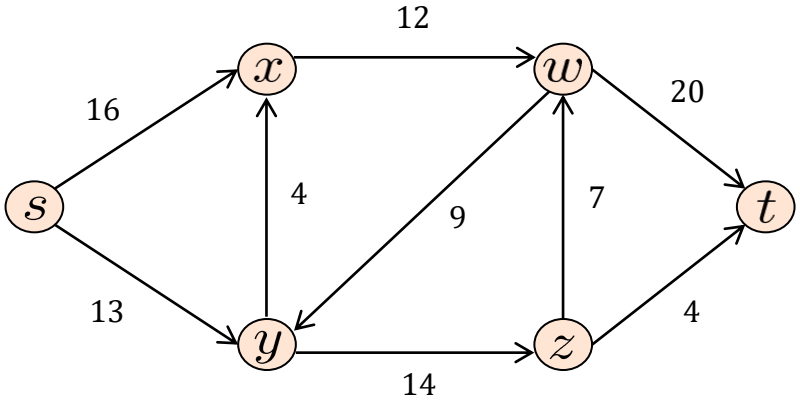
$$|f| = \sum_{v \in V} f(s, v) \text{ = 23}$$

Επαλήθευση…

Υπολοιπόμενες χωρητικότητες:
4, 4, 4, 7, 4

Σκιαγράφηση Αλγόριθμου





Αρχικό δίκτυο $G = (V, E)$

Τιμή της **μέγιστης ροής** του δικτύου:

$$|f| = \sum_{v \in V} f(s, v) \text{ = 23}$$

Αλγόριθμος των Ford-Fulkerson – 1956

Αλγόριθμος των Ford-Fulkerson

αύξηση της ροής κατά 1 μονάδα

1. Θέτουμε τη ροή f ίση με 0
2. **Ενόσω** υπάρχει αυξητική διαδρομή P στο G_f
3. αυξάνουμε τη ροή f κατά μήκος της διαδρομής P
4. Επιστροφή f

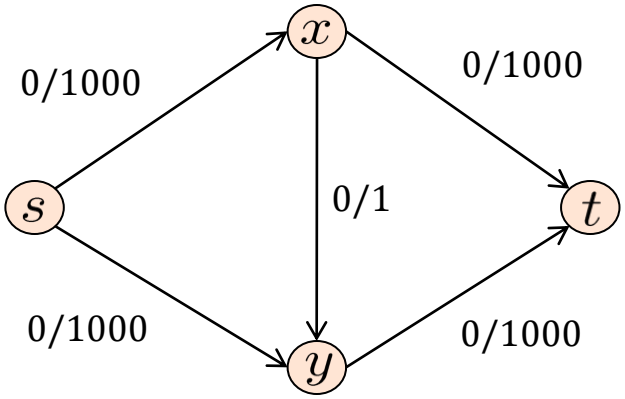
$O(m)$ χρόνος για την εύρεση και επεξεργασία της αυξητικής διαδρομής

Ανάλυση: Υποθέτουμε ακέραιες χωρητικότητες.

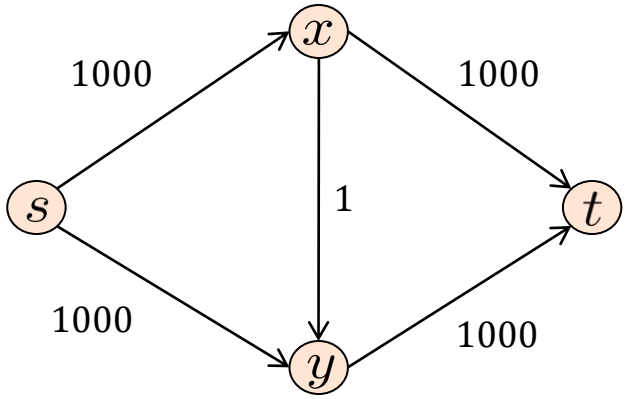
Τότε ο χρόνος εκτέλεσης είναι $O(mf^*)$ όπου f^* η τιμή της μέγιστης ροής.

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



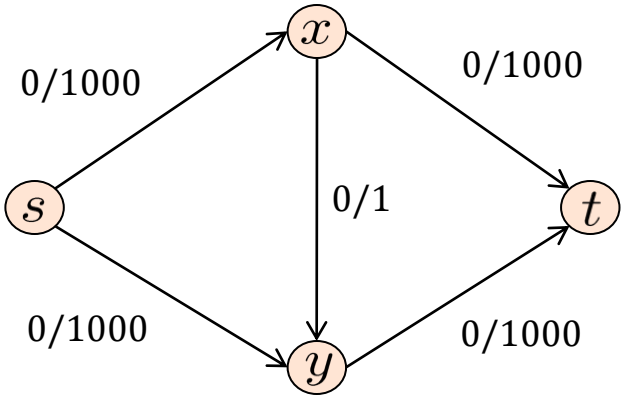
Δίκτυο G



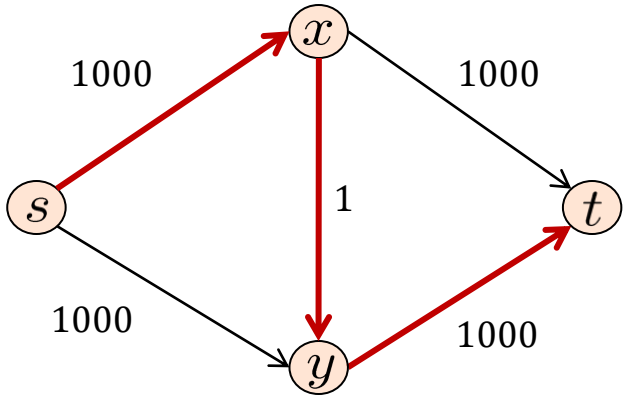
Υπολειπόμενο δίκτυο G_f

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

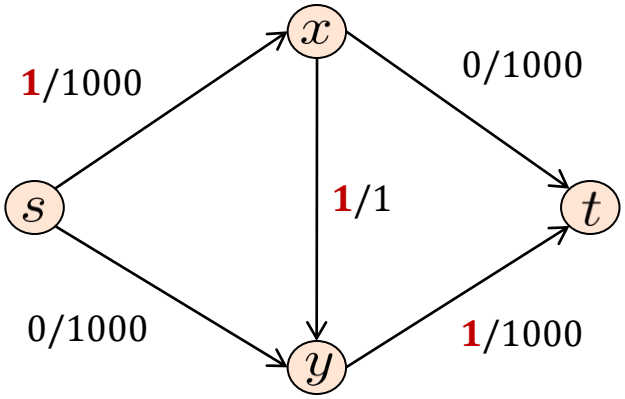


Υπολειπόμενο δίκτυο G_f

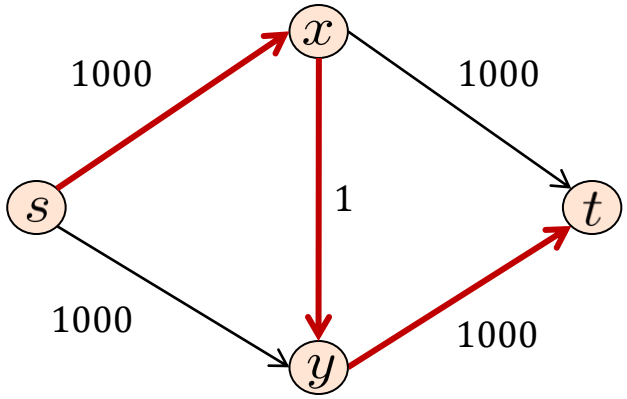
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

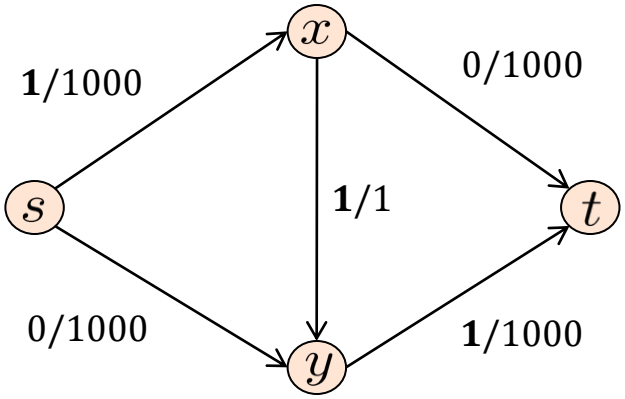


Υπολειπόμενο δίκτυο G_f

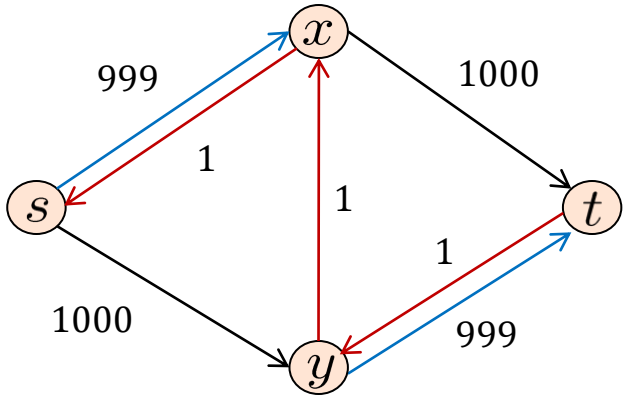
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

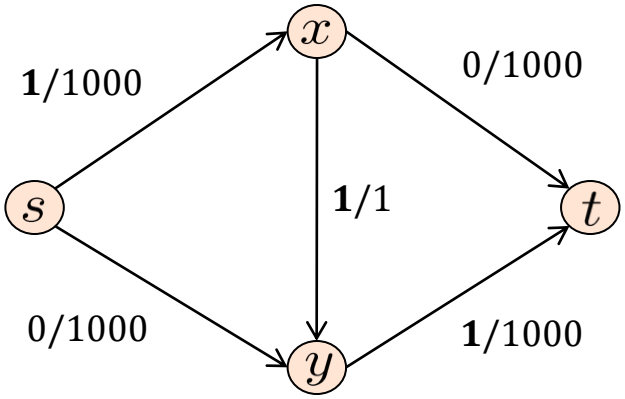


Υπολειπόμενο δίκτυο G_f

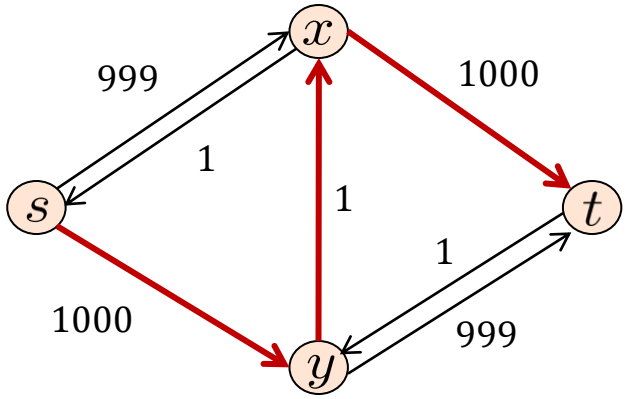
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

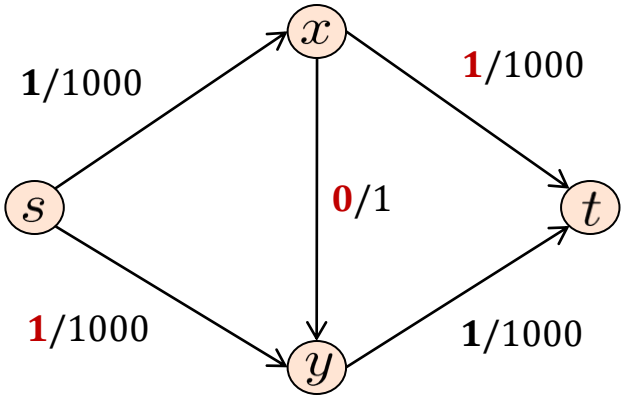


Υπολειπόμενο δίκτυο G_f

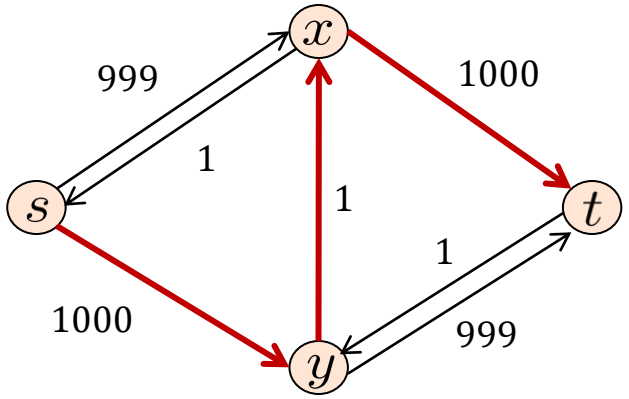
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

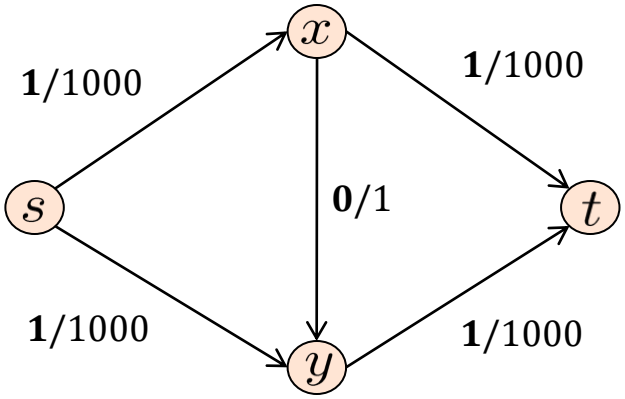


Υπολειπόμενο δίκτυο G_f

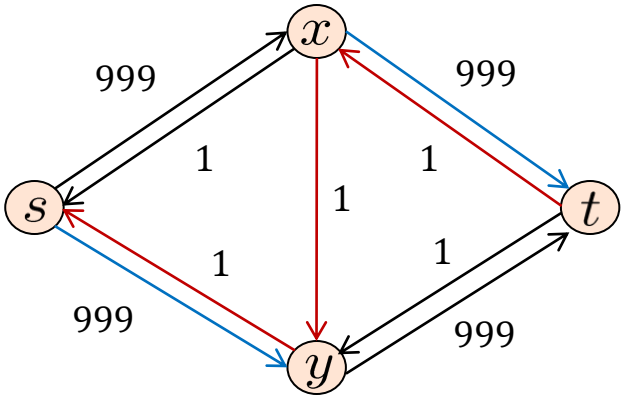
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

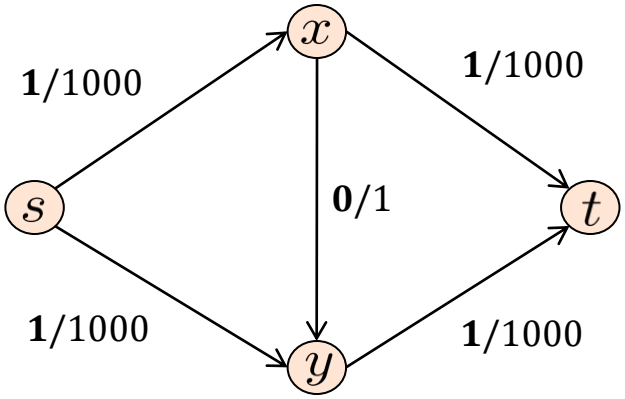


Υπολειπόμενο δίκτυο G_f

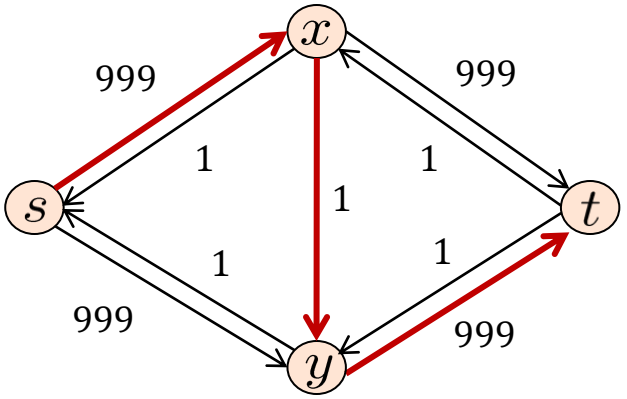
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

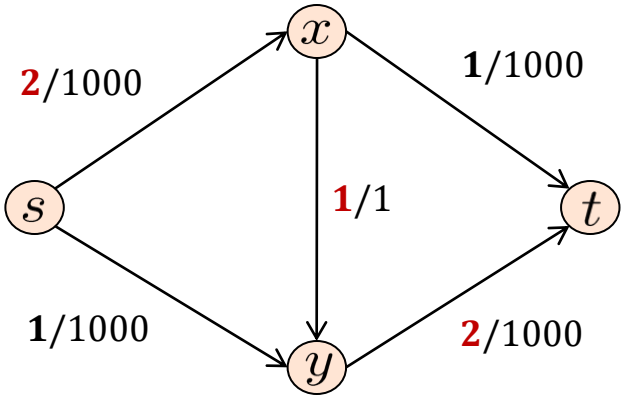


Υπολειπόμενο δίκτυο G_f

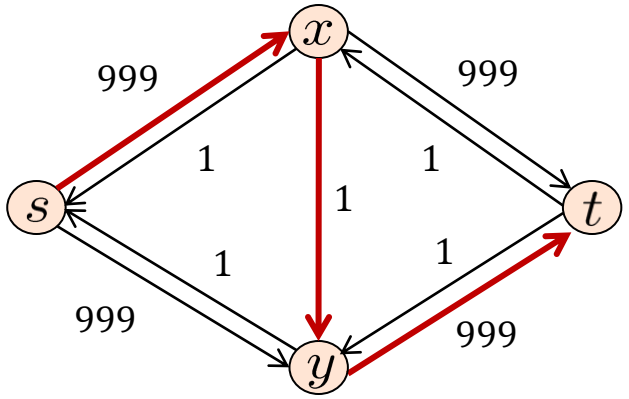
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G

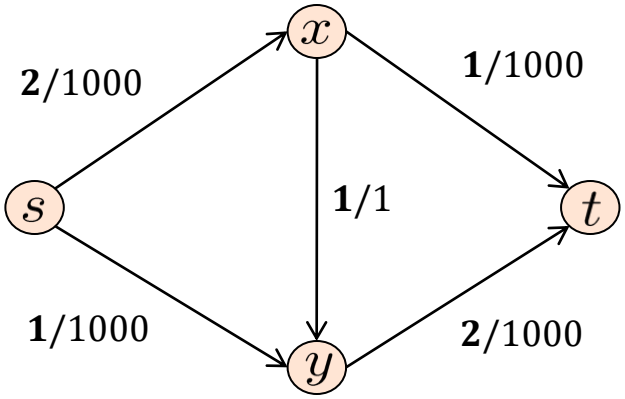


Υπολειπόμενο δίκτυο G_f

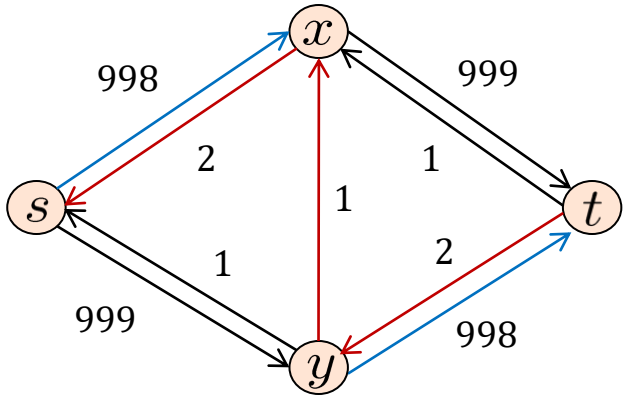
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

Αλγόριθμος των Ford-Fulkerson

Μια κακή περίπτωση



Δίκτυο G



Υπολειπόμενο δίκτυο G_f

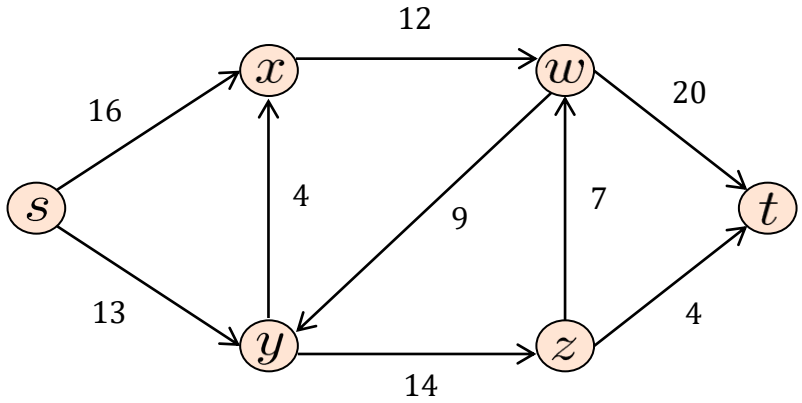
Υπολειπόμενη χωρητικότητα αυξητικής διαδρομής = 1

κ.ο.κ

Ένα Πιστοποιητικό Βελτιστότητας

Στη μέθοδο των Ford-Fulkerson αυξάνουμε τη ροή κατά μήκος αυξητικών διαδρομών έως να βρεθεί μια μέγιστη ροή !!!

- ✓ Πως γνωρίζουμε ότι πράγματι έχουμε βρει μια μέγιστη ροή ?
- ✓ Υπάρχει ροή με $|f| > 23$?



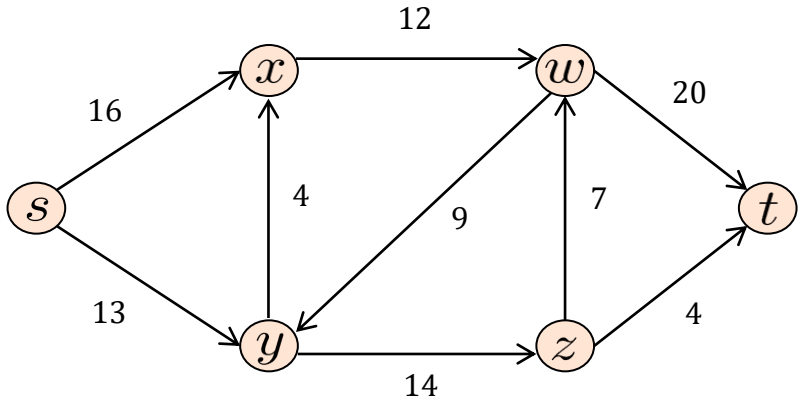
Δίκτυο $G = (V, E)$

Τιμή της **μέγιστης ροής** του δικτύου:

$$|f| = \sum_{v \in V} f(s, v) \quad \mathbf{= 23}$$

Ένα Πιστοποιητικό Βελτιστότητας

Στη μέθοδο των Ford-Fulkerson αυξάνουμε τη ροή κατά μήκος αυξητικών διαδρομών έως να βρεθεί μια μέγιστη ροή !!!



Δίκτυο $G = (V, E)$

- ✓ Την απάντηση δίνει το

Θεώρημα

Μέγιστης Ροής – Ελάχιστης Τομής

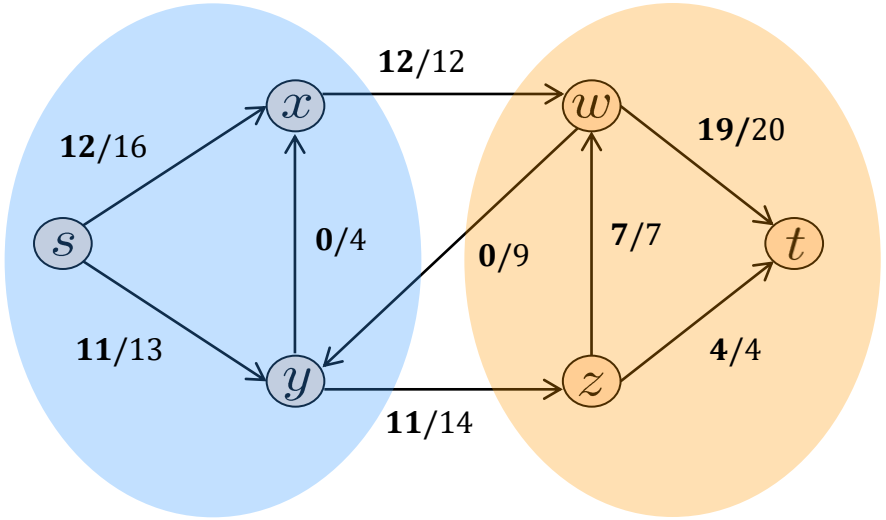
Ένα Πιστοποιητικό Βελτιστότητας

Τομή Δικτύου Ροής (S, T) :

Μια διαμέριση των κόμβων του δικτύου σε σύνολα

$S \subset V$ και $T = V - S$

τέτοια ώστε $s \in S$ και $t \in T$

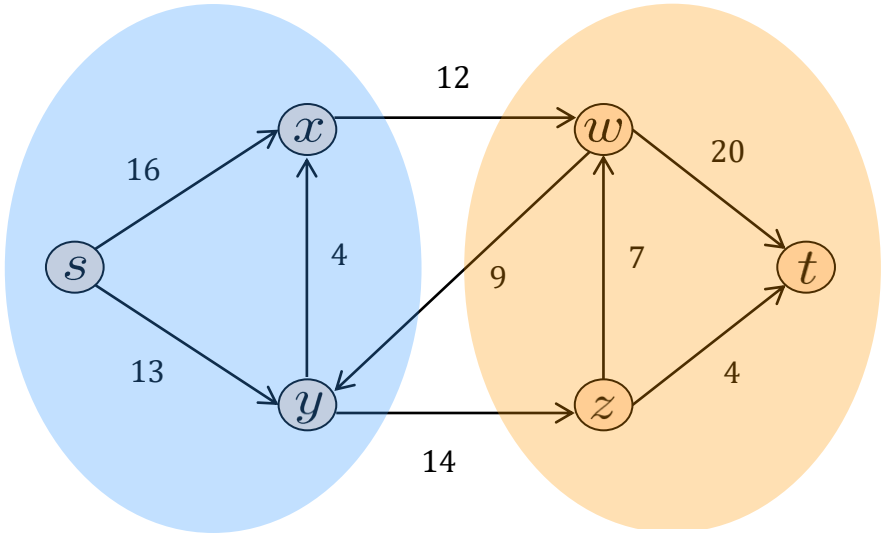


Καθαρή ροή δια μέσου της τομής (S, T): $f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v)$

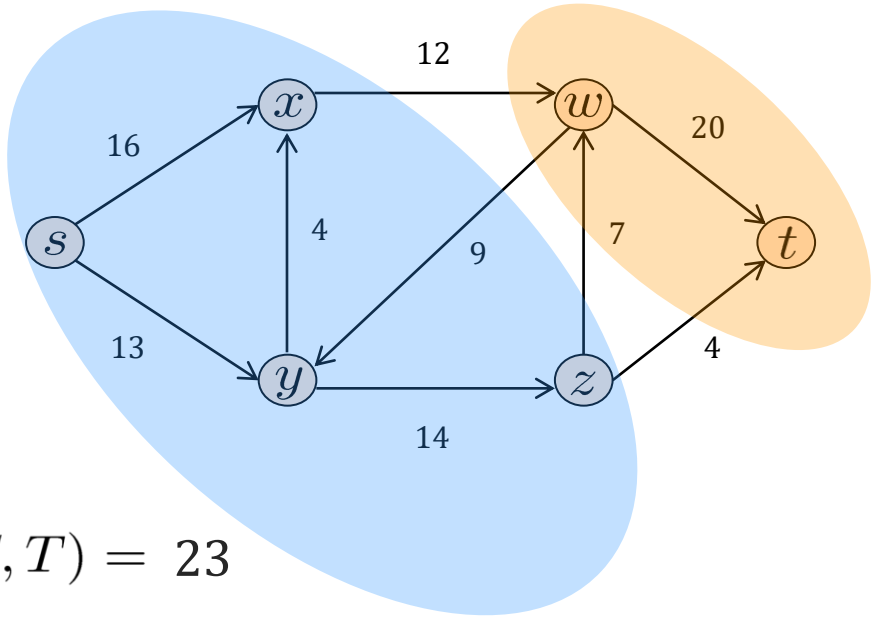
Χωρητικότητα της τομής (S, T) : $c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$

Θεώρημα (Μέγιστης ροής – Ελάχιστης τομής)

Το μέγεθος της μέγιστης ροής σε ένα δίκτυο ισούται με την χωρητικότητα της ελάχιστης τομής (S, T)



$c(S,T) = 26$

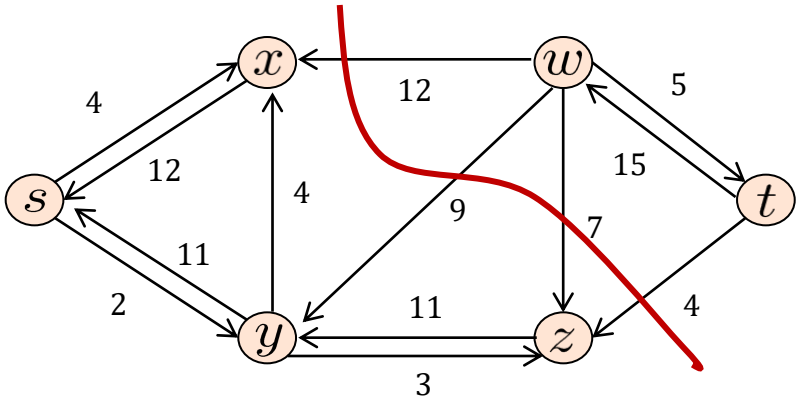


$c(S,T) = 23$

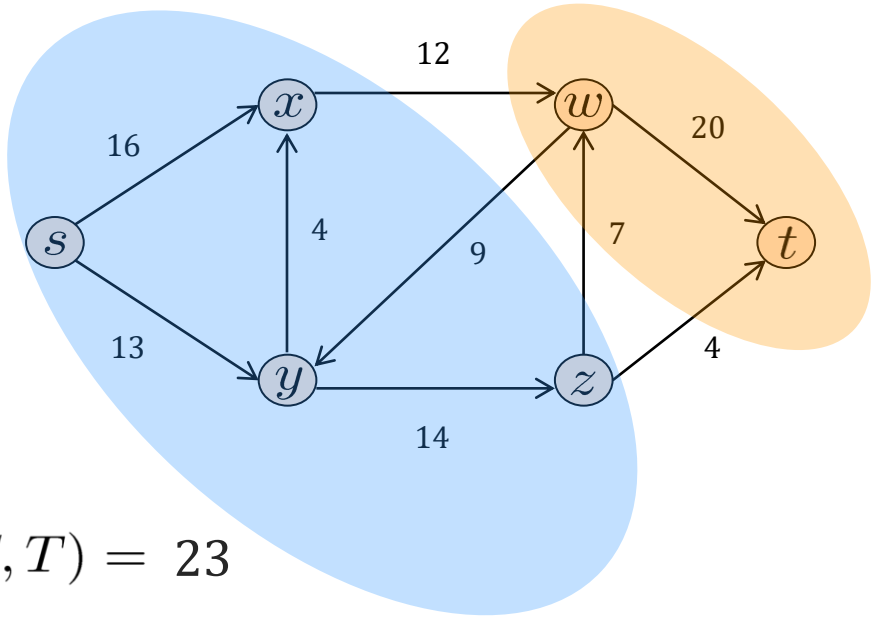
Θεώρημα (Μέγιστης ροής – Ελάχιστης τομής)

Οι ακόλουθες προτάσεις είναι ισοδύναμες:

1. Η f είναι μια μέγιστη ροή στο G
2. Δεν υπάρχει αυξητική διαδρομή στο υπολειπόμενο δίκτυο G_f
3. Υπάρχει τομή (S, T) του G τέτοια ώστε $|f| = c(S, T)$



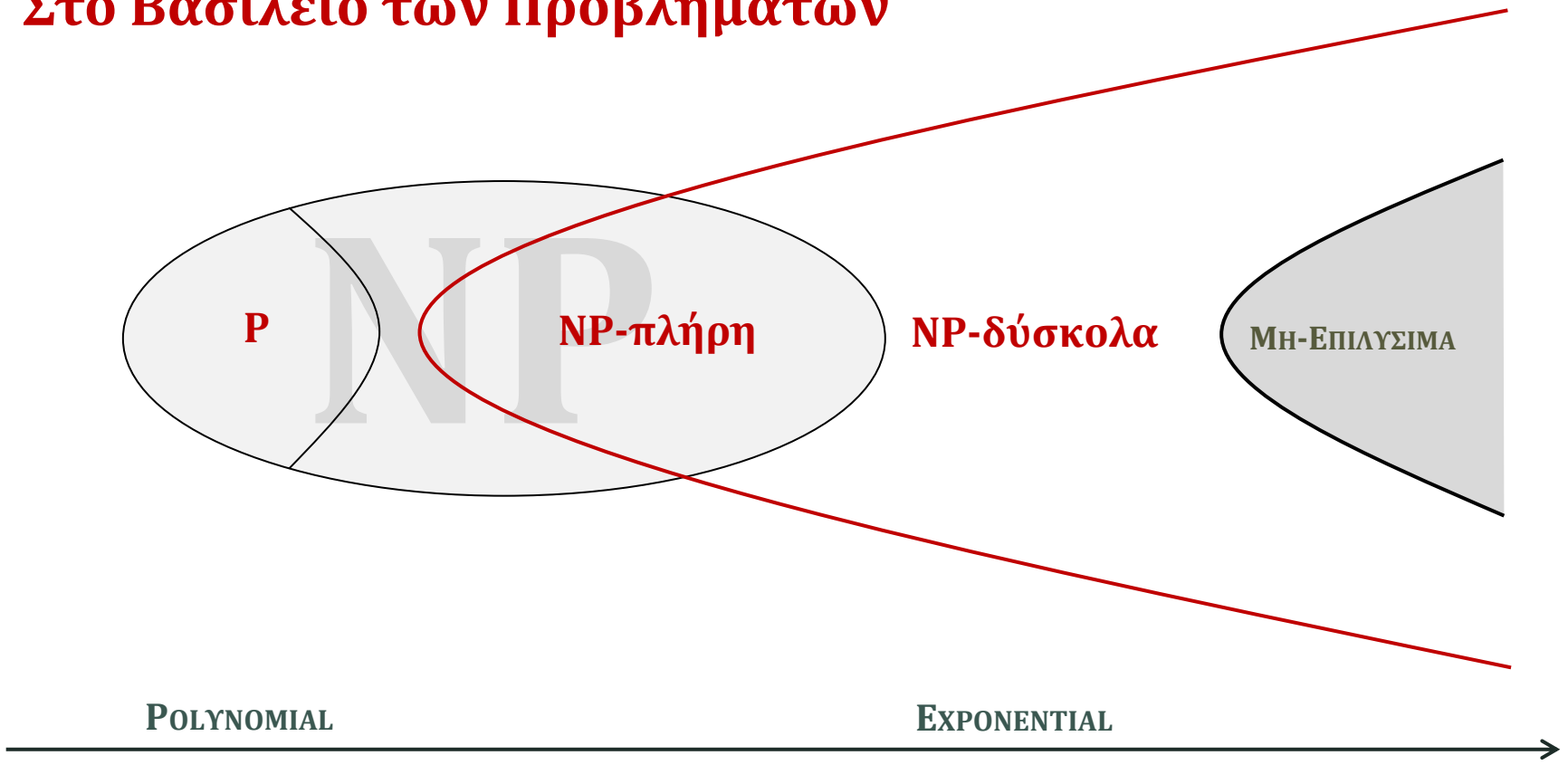
G_f



$c(S, T) = 23$

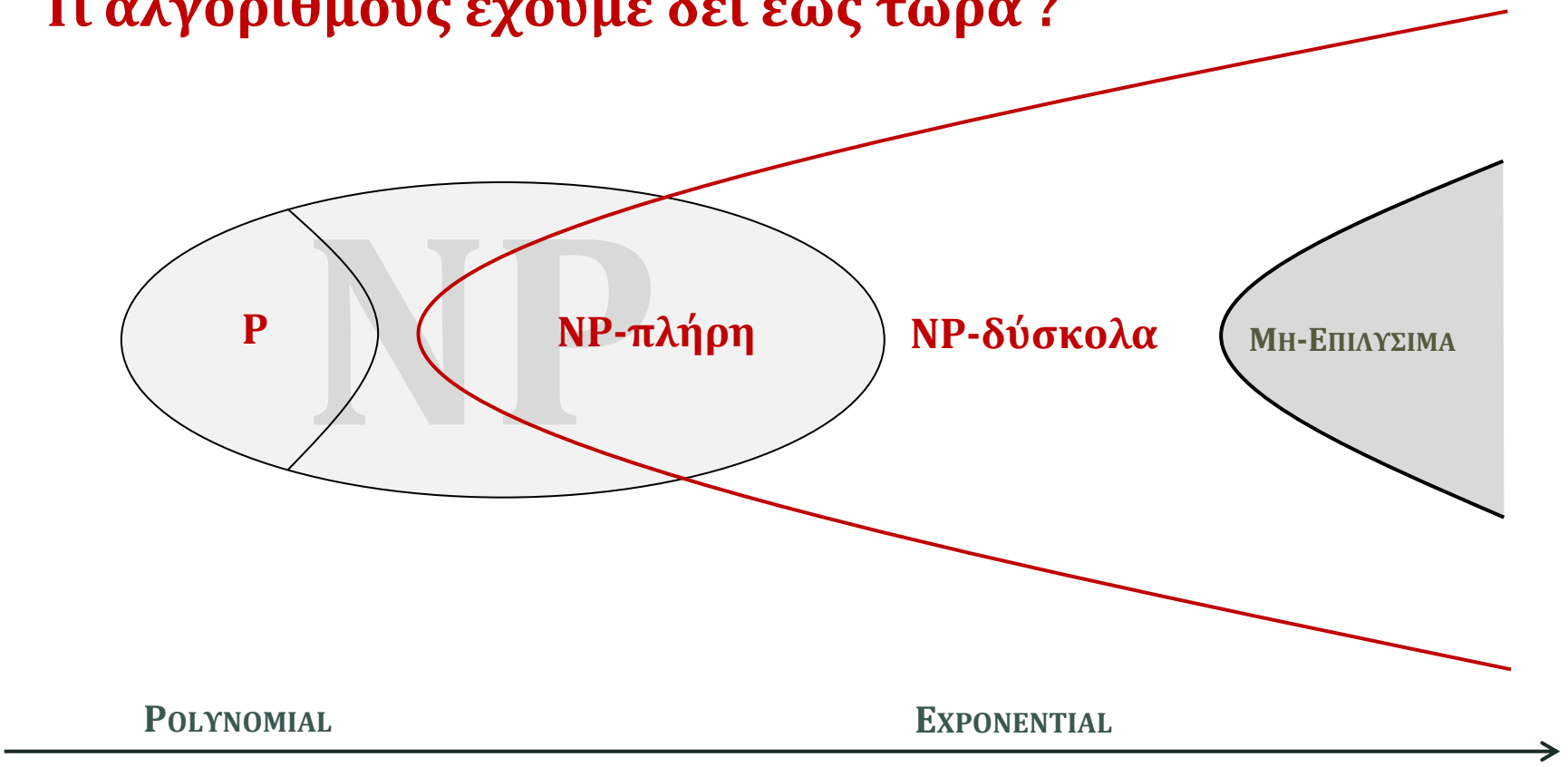
Προβλήματα Εύκολα, Δύσκολα και Μη-επιλύσιμα

- **Στο Βασίλειο των Προβλημάτων**



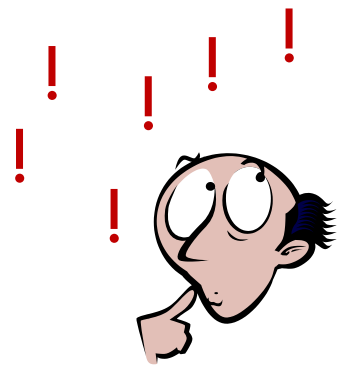
Προβλήματα Εύκολα, Δύσκολα και Μη-επιλύσιμα

- **Τι αλγορίθμους έχουμε δει έως τώρα ?**



Τι αλγορίθμους έχουμε δει έως τώρα ?

- Αναπτύξαμε αλγορίθμους για να βρίσκουμε:
  - ✓ ελάχιστες διαδρομές,
  - ✓ αντιστοιχίσεις σε διμερή γραφήματα,
  - ✓ μέγιστες ροές σε δίκτυα, και



γενικά, για να βρίσκουμε βέλτιστες (optimal) λύσεις !
με χρήση έξυπνων ***αλγοριθμικών τεχνικών*** και ***αναγωγών*** !!

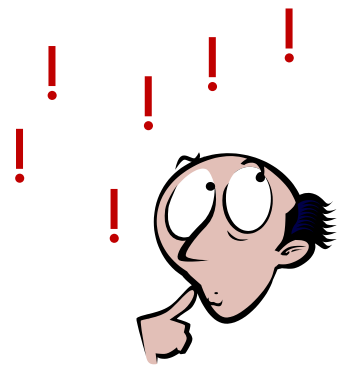
- Όλοι αυτοί οι αλγόριθμοι είναι **αποδοτικοί** (efficient) !!!

ο χρόνος εκτέλεσής τους $T(n)$ είναι πολυωνυμική συνάρτηση του *μεγέθους της εισόδου* n (όπως n, n^2, n^3,…)

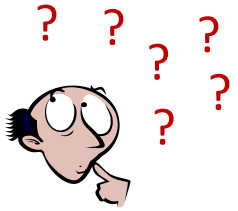
Τι αναζητούμε σε όλα αυτά τα Προβλήματα ?

- Σε όλα **αναζητούμε μια λύση**, για παράδειγμα:
  - ✓ μια διαδρομή,
  - ✓ ένα δένδρο,
  - ✓ μια αντιστοίχιση, κλπ

  ανάμεσα σε ένα **εκθετικό πλήθος** λύσεων !!!



- **Πράγματι:**



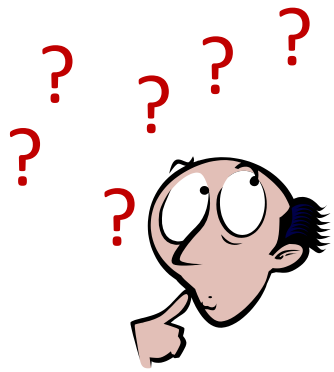
n αγόρια μπορούν να αντιστοιχηθούν με n κορίτσια με **n!** διαφορετικούς τρόπους !!!

ένα γράφημα τάξης n έχει $\mathbf{n^{n-2}}$ συνδετικά δένδρα

ένα τυπικό γράφημα τάξης n έχει **εκθετικό πλήθος** διαδρομών από τον κόμβο s στον t.

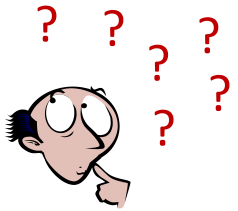
Προβλήματα Αναζήτησης

- **ΟΛΑ** αυτά τα **Προβλήματα Αναζήτησης** μπορούν να λυθούν σε **εκθετικό χρόνο !!!**
  - ✓ Με τον έλεγχο **όλων** των **υποψηφίων λύσεων**, μία προς μία !!!



Όμως, ένας αλγόριθμος με εκθετική πολυπλοκότητα, όπως $T(n) = 2^n$**,** είναι πρακτικά **άχρηστος !!!**

- **Τι Θέλουμε !!!**



Έξυπνες και Πρωτότυπες Ιδέες !!!

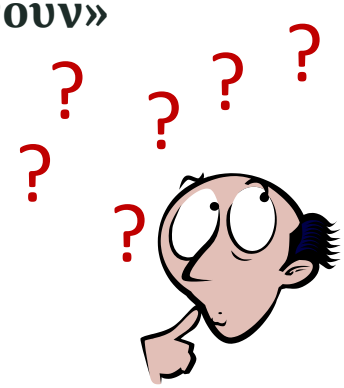
για να παρακάμψουμε την

Εξαντλητική Αναζήτηση !!!

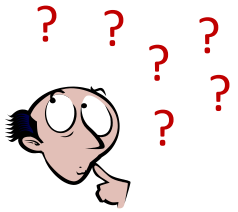
- **Υπάρχουν Έξυπνες και Πρωτότυπες Ιδέες ?**

ΝΑΙ !!! έχουμε δει **αλγοριθμικές τεχνικές** που **«κατατροπώνουν»** τον **εφιάλτη** της **εκθετικότητας** !!!

- ✓ Άπληστη τεχνική
- ✓ Δυναμικός προγραμματισμός
- ✓ Γραμμικός Προγραμματισμός



- **Κάποτε όμως έρχεται και η ώρα της…**

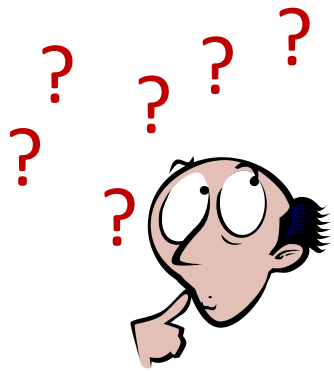


Αποτυχίας !!!

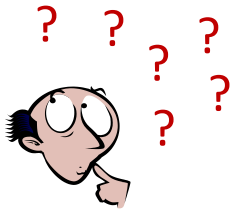
Τι σημαίνει «Αποτυχία» ?

- ✓ Υπάρχουν κάποια **«προβλήματα αναζήτησης»** για τα οποία :

δεν φαίνεται να υπάρχει
αποτελεσματικός τρόπος επίλυσης



Τι ξέρουμε για τα προβλήματα αυτά ?



Οι **ταχύτεροι αλγόριθμοι** που γνωρίζουμε είναι όλοι **εκθετικοί !!!**

Ορίστε μερικά από αυτά !!!...

1 Ικανοποιησιμότητα

Πρόβλημα SAT

Μας δίδεται ένας λογικός τύπος (Boolean formula) σε κανονική συζευκτική μορφή (CNF), όπως για παράδειγμα:

$$(x \vee y \vee z) \wedge (\bar{x} \vee y) \wedge (y \vee \bar{z}) \wedge (z \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$

και μας ζητείται είτε να βρούμε μια *ικανοποιούσα ανάθεση τιμών αληθείας* (satisfying truth assignment) ή να αναφέρουμε ότι δεν υπάρχει καμία !!!

Μια *ικανοποιούσα ανάθεση τιμών αληθείας* είναι μια ανάθεση τιμών **true** ή **false** σε κάθε μεταβλητή έτσι ώστε *ο λογικός τύπος να είναι* **true** **!!!**

2 Το πρόβλημα του Περιοδεύοντος Πωλητή

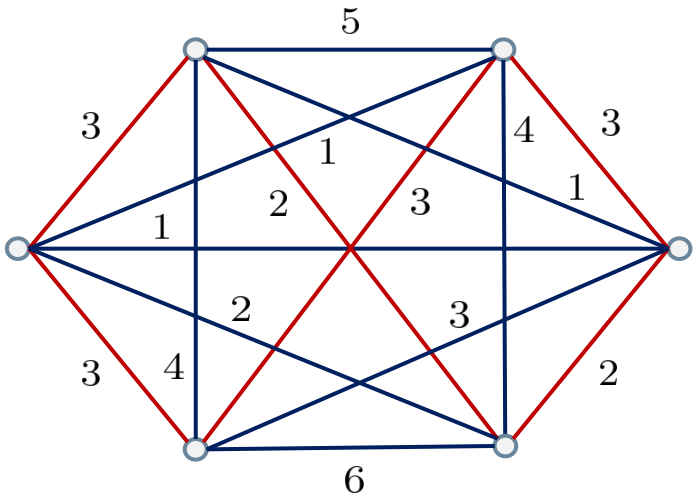
Πρόβλημα TSP

Μας δίδεται ένα πλήρες έμβαρο γράφημα **n** κόμβων, καθώς και ένας *προϋπολογισμός* **b**,

και μας ζητείται μια περιήγηση (tour), δηλαδή ένας κύκλος που διέρχεται από κάθε κόμβο μία μόνο φορά,

με συνολικό κόστος το πολύ **b**

ή να αναφέρουμε ότι δεν υπάρχει τέτοια περιήγηση !!!



Η βέλτιστη περιήγηση του παραδείγματος έχει κόστος 16 !!!

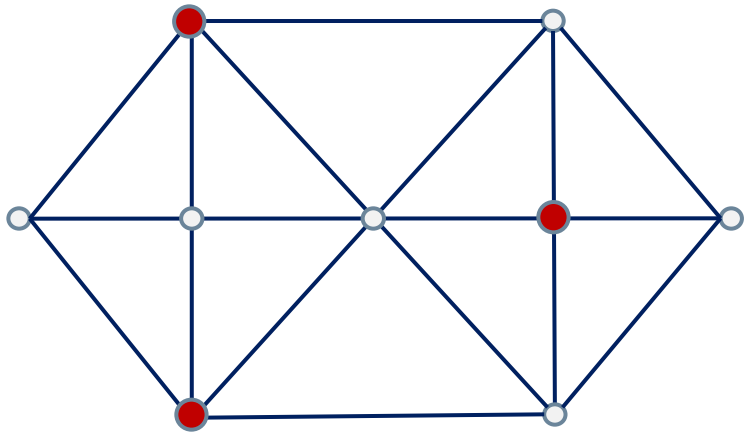
3 Ανεξάρτητα Σύνολα Γραφημάτων

Πρόβλημα ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

Μας δίδεται ένα γράφημα **n** κόμβων και ένας *ακέραιος* **k**,

και μας ζητείται να βρούμε ένα σύνολο **IS** με **k** κόμβους οι οποίοι είναι ανεξάρτητοι (independent), δηλαδή ανά δύο δεν έχουν καμία ακμή μεταξύ τους,

ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!



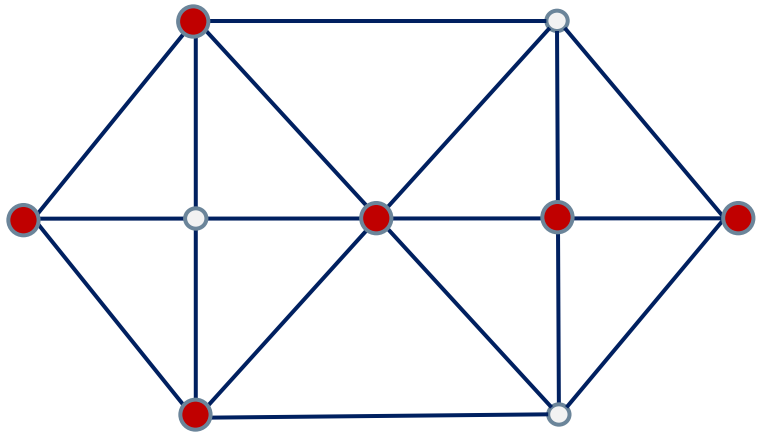
Το πρόβλημα λύνεται αποδοτικά σε δένδρα, αλλά όχι σε γενικά γραφήματα!!!

4 Κάλυψη Γραφήματος με Κόμβους

Πρόβλημα Κομβικη-Καλυψη

Μας δίδεται ένα γράφημα **n** κόμβων και ένας *ακέραιος* **k**,

και μας ζητείται να βρούμε ένα σύνολο **VC** με **k** κόμβους οι οποίοι καλύπτουν κάθε ακμή, δηλαδή κάθε ακμή έχει τουλάχιστον ένα κόμβο της στο σύνολο VC,



ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!

Το πρόβλημα λύνεται αποδοτικά
σε κάποιες κλάσεις γραφημάτων, αλλά όχι σε γενικά γραφήματα!!!

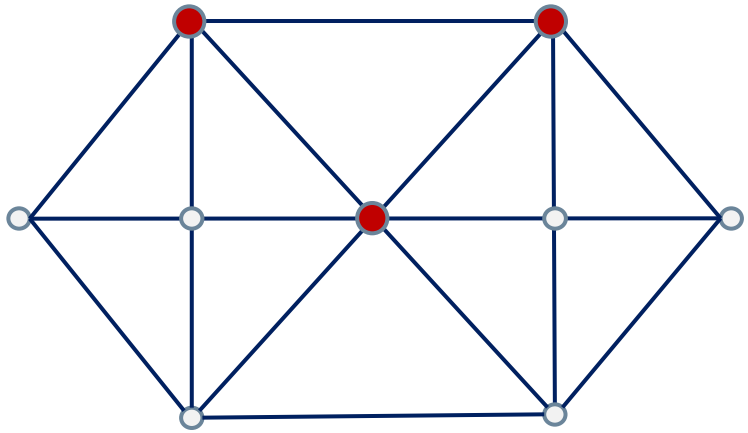
5 Κλίκες Γραφημάτων

Πρόβλημα Κλικα

Μας δίδεται ένα γράφημα **n** κόμβων και ένας *ακέραιος* **k**,

και μας ζητείται να βρούμε ένα σύνολο **CL** με **k** κόμβους οι οποίοι έχουν πλήρη σύνδεση, δηλαδή ανά δύο έχουν ακμή μεταξύ τους,

ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!



Το πρόβλημα λύνεται αποδοτικά
σε κάποιες κλάσεις γραφημάτων, αλλά όχι σε γενικά γραφήματα!!!

6 Μέγιστες Διαδρομές

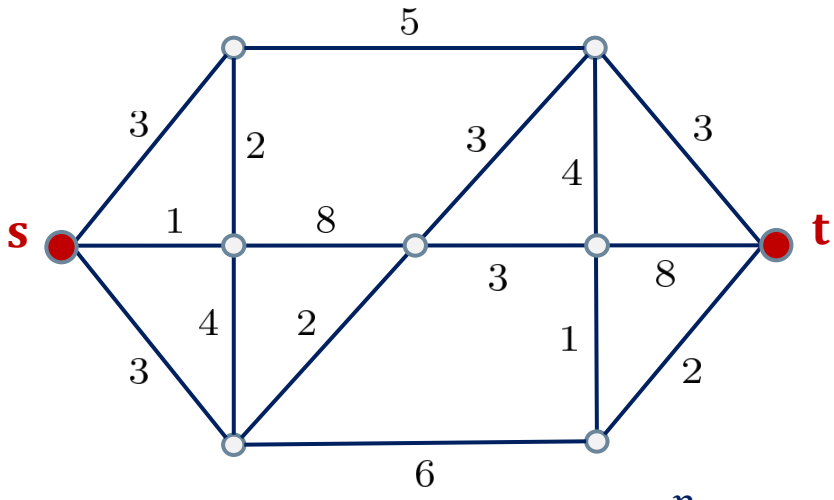
Πρόβλημα Μεγιστη-Διαδρομη

Μας δίδεται ένα έμβαρο γράφημα **n** κόμβων (με μη-αρνητικά βάρη) και δύο διακριτοί κόμβοι **s** και **t** μαζί με έναν *ακέραιο* **k**,

και μας ζητείται να βρούμε μια διαδρομή **Δ** από τον s στον t

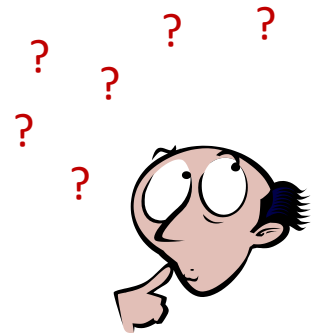
με συνολικό κόστος τουλάχιστον **k**

ή να αναφέρουμε ότι δεν υπάρχει τέτοια διαδρομή !!!



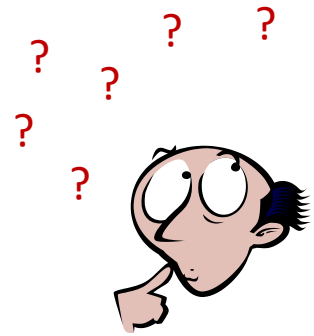
Για να αποφύγουμε τετριμμένες λύσεις απαιτούμε η διαδρομή να είναι *απλή*, δηλαδή να μην περιέχει επαναλαμβανόμενους κόμβους !!!

Δύσκολα & Εύκολα Προβλήματα !!!



Δύσκολα Προβλήματα	Εύκολα Προβλήματα
3SAT	2SAT
TSP	MST
Μεγιστη Διαδρομη	Ελαχιστη Διαδρομη
Σακιδιο	Μοναδιαιο Σακιδιο
Ανεξαρτητο Συνολο	Ανεξαρτητο Συνολο σε Δενδρα
ILP	LP
Διαδρομη Hamilton	Διαδρομη Euler

Δύσκολα & Εύκολα Προβλήματα !!!



Δύσκολα Προβλήματα

3SAT

TSP

Μεγιστη Διαδρομη

Σακιδιο

Ανεξαρτητο Συνολο

ILP

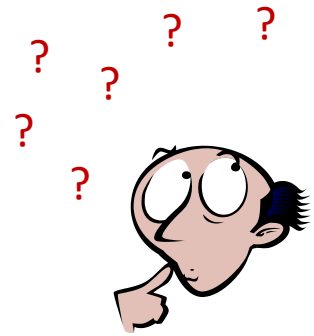
Διαδρομη Hamilton

Εύκολα Προβλήματα

Όλα τα προβλήματα εδώ λύνονται **αποδοτικά** με αλγορίθμους:

- ✓ Αναζήτησης σε γραφήματα
- ✓ Άπληστων τεχνικών
- ✓ Ροών
- ✓ Δυναμικού προγραμματισμού
- ✓ Γραμμικού προγραμματισμού

Δύσκολα & Εύκολα Προβλήματα !!!



Δύσκολα Προβλήματα

Θλιβερή Αντίθεση !!!

Τα προβλήματα εδώ **είναι όλα δύσκολα**, όλα για τον ίδιο λόγο !!!

Στη βάση τους είναι **όλα το ίδιο πρόβλημα !!!**

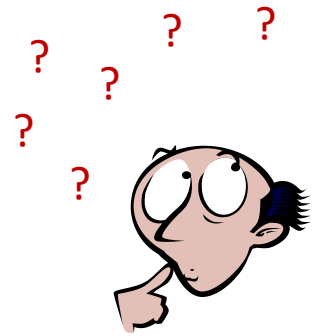
με ***διαφορετικές μεταμφιέσεις*** !!!

Εύκολα Προβλήματα

Όλα τα προβλήματα εδώ λύνονται **αποδοτικά** με αλγορίθμους:

- ✓ Αναζήτησης σε γραφήματα
- ✓ Άπληστων τεχνικών
- ✓ Ροών
- ✓ Δυναμικού προγραμματισμού
- ✓ Γραμμικού προγραμματισμού

Δύσκολα & Εύκολα Προβλήματα !!!



Δύσκολα Προβλήματα

Θλιβερή Αντίθεση !!!

Τα προβλήματα εδώ **είναι όλα δύσκολα**, όλα για τον ίδιο λόγο !!!

Στη βάση τους είναι **όλα το ίδιο πρόβλημα !!!**

NP-πλήρη !!!

Εύκολα Προβλήματα

Όλα τα προβλήματα εδώ λύνονται **αποδοτικά** με αλγορίθμους:

- ✓ Αναζήτησης σε γραφήματα
- ✓ Άπληστων τεχνικών
- ✓ Ροών
- ✓ Δυναμικού προγραμματισμού
- ✓ Γραμμικού προγραμματισμού

P και NP !!!

- **Γνωρίζουμε τι είναι ένα Πρόβλημα Αναζήτησης** !!!

Χαρακτηριστικό:

✓ Μια **προτεινόμενη λύση S** μπορεί να **ελεγχθεί γρήγορα** για την ορθότητά της !!!

P και NP !!!

- **Γνωρίζουμε τι είναι ένα Πρόβλημα Αναζήτησης** !!!

Υπάρχει αποδοτικός (πολυωνυμικός) **αλγόριθμος ελέγχου C**

✓ Ο χρόνος εκτέλεσης του αλγόριθμου **C(I, S)** φράσσεται από ένα πολυώνυμο ως προς **|I|** , όπου I ένα στιγμιότυπο του προβλήματος.

Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***μια Λύση τους Ελέγχεται*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **NP !!!**

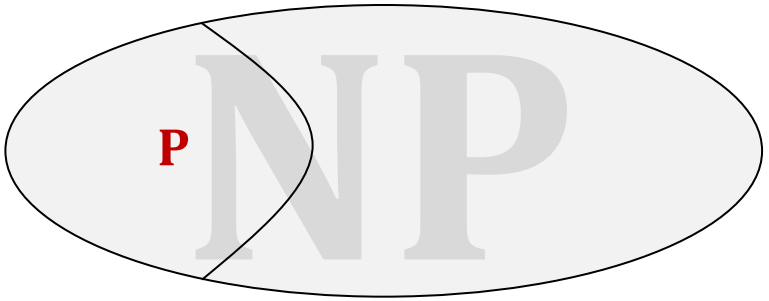
P και NP !!!

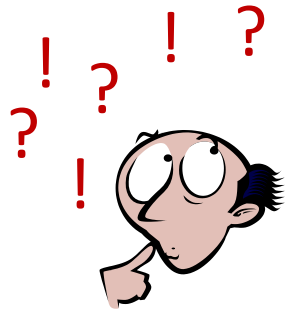
- **Γνωρίζουμε τι είναι ένα Πρόβλημα Αναζήτησης !!!**

Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***μια Λύση τους Ελέγχεται*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **NP !!!**

P και NP !!!

- **ΠΟΛΛΑ πρόβλημα αναζήτησης NP** γνωρίζουμε ότι μπορούν να λυθούν σε **πολυωνυμικό χρόνο** !!!

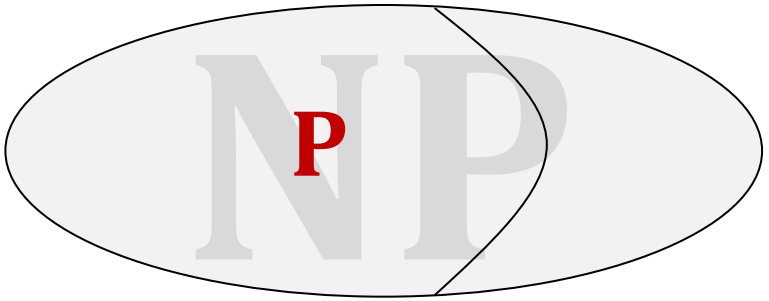


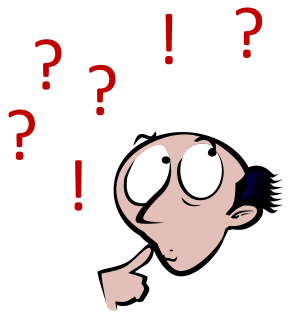


Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***Μπορούν να Λυθούν*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **P !!!**

Το Μεγάλο Ερώτημα !!!

- **Πόσα ΠΟΛΛΑ προβλήματα αναζήτησης NP** υπάρχουν που μπορούν να λυθούν σε **πολυωνυμικό χρόνο ?**

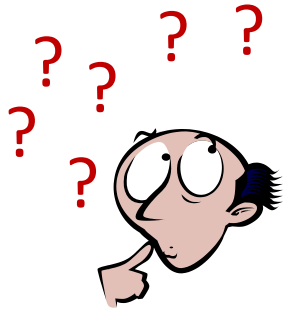




Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***Μπορούν να Λυθούν*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **P !!!**

Το Μεγάλο Ερώτημα !!!

- **Μήπως ΟΛΑ τα προβλήματα αναζήτησης NP** μπορούν να λυθούν σε **πολυωνυμικό χρόνο ?**

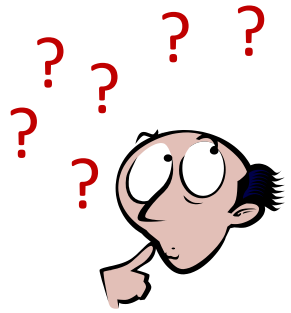


Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***Μπορούν να Λυθούν*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **P !!!**

Το Μεγάλο Ερώτημα !!!

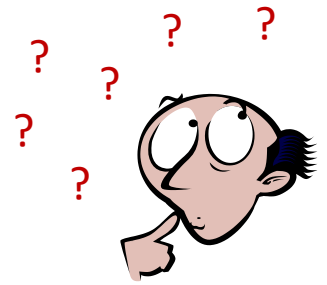
- **Μήπως ΟΛΑ τα προβλήματα αναζήτησης NP** μπορούν να λυθούν σε **πολυωνυμικό χρόνο ?**

P = NP



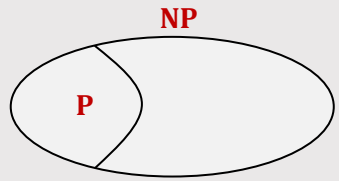
Συμβολίζουμε *την κλάση* ***ΟΛΩΝ*** *των* ***ΠΡΟΒΛΗΜΑΤΩΝ ΑΝΑΖΗΤΗΣΗΣ*** *που* ***Μπορούν να Λυθούν*** *σε* ***ΠΟΛΥΩΝΥΜΙΚΟ ΧΡΟΝΟ*** *με* **P !!!**

Το Μεγάλο Ερώτημα !!!



- **Ισχύει:**

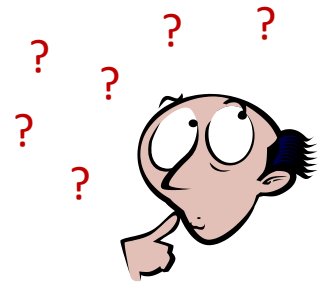
$$\mathbf{P = NP} \quad \text{ή} \quad \mathbf{P \neq NP} \quad ?$$



Οι περισσότεροι ερευνητές πιστεύουν πως **P≠NP** !!!

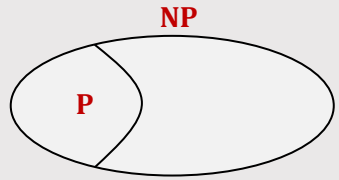
- ✓ Είναι δύσκολο να πιστέψουμε ότι μπορεί πάντοτε να αποφευχθεί η εκθετική αναζήτηση !...
- ✓ Ή, ότι μια τεχνική θα «σπάσει» όλα αυτά τα δύσκολα προβλήματα !
- ✓ Υπάρχει ένας καλός λόγος για να πιστεύουν οι μαθηματικοί ότι $P \neq NP$!

Το Μεγάλο Ερώτημα !!!



- **Ισχύει:**

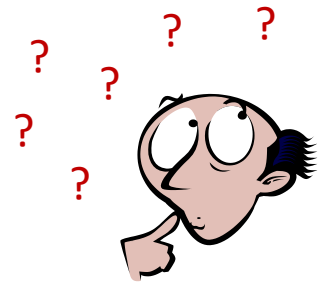
$$\mathbf{P = NP} \quad \text{ή} \quad \mathbf{P \neq NP} \quad ?$$



Ωστόσο !...

Δεν υπάρχει απόδειξη !!!

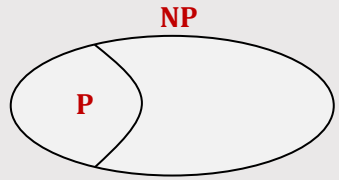
και η απόδειξη φαίνεται, και ίσως είναι, εξαιρετικά δύσκολη !!!



Το Μεγάλο Ερώτημα !!!

- **Ισχύει:**

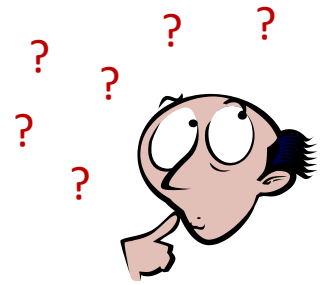
$$\mathbf{P = NP} \quad \text{ή} \quad \mathbf{P \neq NP} \quad ?$$



Το ερώτημα είναι ένα από τα **βαθύτερα** και **σημαντικότερα αναπάντητα ερωτήματα** της επιστήμης μας **!!!**

Million Dollar Question !
(Clay Institute millennium problems)

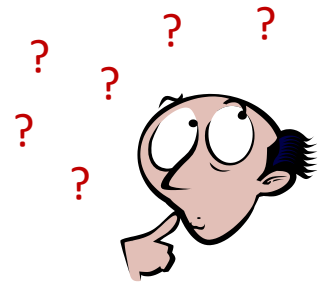
Δύσκολα Προβλήματα !!!



- Εάν ισχύει: **P ≠ NP**

«Ζώνη Αβεβαιότητας» NP-P

Δύσκολα Προβλήματα !!!



- **Εάν ισχύει:** **P ≠ NP**

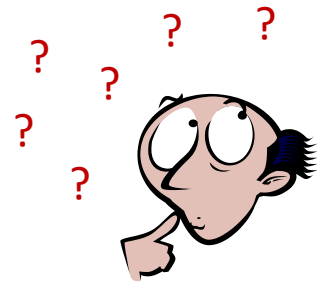
✓ Τι ισχύει για τα **προβλήματα αναζήτησης** του αριστερού πίνακα ?

✓ Υπάρχουν **ΕΝΔΕΙΞΕΙΣ** ότι αυτά είναι ιδιαίτερα προβλήματα και ότι ***δεν έχουν αποδοτικούς αλγορίθμους*** επίλυσης !!!

3SAT
TSP
ΜΕΓΙΣΤΗ ΔΙΑΔΡΟΜΗ
ΣΑΚΙΔΙΟ
ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ
ILP
ΔΙΑΔΡΟΜΗ HAMILTON

2SAT
MST
ΕΛΑΧΙΣΤΗ ΔΙΑΔΡΟΜΗ
ΜΟΝΑΔΙΑΙΟ ΣΑΚΙΔΙΟ
ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ ΣΕ ΔΕΝΔΡΑ
LP
ΔΙΑΔΡΟΜΗ EULER

- **Τέτοιες ενδείξεις παρέχουν οι Αναγωγές**



Δύσκολα Προβλήματα !!!

- Θα δείξουμε, **μέσω αναγωγών**, ότι :

όλα τα **προβλήματα του πίνακα**, κατά μία έννοια (υπολογιστικά),

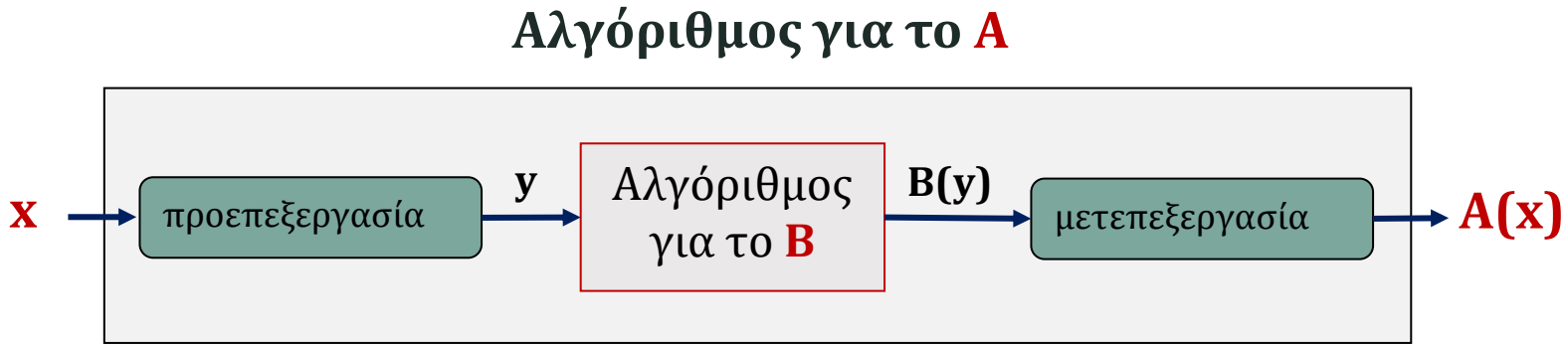
- ✓ είναι ακριβώς το ίδιο πρόβλημα !
- ✓ είναι τα δυσκολότερα στην κλάση NP !
- ✓ εάν ένα λυθεί πολυωνυμικά, τότε κάθε πρόβλημα στην NP λύνεται πολυωνυμικά !

3SAT
TSP
Μεγιστη Διαδρομη
Σακιδιο
Ανεξαρτητο Συνολο
ILP
Διαδρομη Hamilton

- Έτσι, εάν πιστεύουμε ότι **P≠NP** τότε όλα τα προβλήματα του πίνακα είναι **δύσκολα !**

Πάλι Αναγωγές !!!

- **Ας θυμηθούμε !!!...** οι αναγωγές **εκφράζουν** ένα **πρόβλημα αναζήτησης** σε κάποιο άλλο !!!
- **A ≤ B**



ΤΕΛΕΙΑ-ΑΝΤΙΣΤΟΙΧΙΣΗ ≤ ΜΕΓΙΣΤΗ-ΡΟΗ

ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ≤ LP

Πάλι Αναγωγές !!!

- **Ας θυμηθούμε !!!...** έως τώρα, **ο σκοπός της αναγωγής** ήταν **σαφής** και **έντιμος** !!!
- **A ≤ B**

Άνω Φράγματα

γνωρίζουμε έναν **αποδοτικό αλγόριθμο** για το **B**,
και τον ***χρησιμοποιούμε***
για να **λύσουμε αποδοτικά** το πρόβλημα **A** !!!

ΤΕΛΕΙΑ-ΑΝΤΙΣΤΟΙΧΙΣΗ ≤ ΜΕΓΙΣΤΗ-ΡΟΗ

ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ≤ LP

ΜΕΓΙΣΤΗ-ΑΥΞΟΥΣΑ-ΥΠΑΚΟΛΟΥΘΙΑ ≤ ΜΕΓΙΣΤΗ-ΔΙΑΔΡΟΜΗ-DAG

Πάλι Αναγωγές !!!

- **Στη συνέχεια…** θα ***χρησιμοποιήσουμε*** τις **αναγωγές** για ένα *κάπως* ***διεστραμμένο*** *στόχο*:
- **A ≤ B**

Κάτω Φράγματα

γνωρίζουμε ότι το πρόβλημα **A είναι δύσκολο**,
και ***χρησιμοποιούμε*** την αναγωγή
για **να αποδείξουμε ότι** και το **B** είναι **δύσκολο** !!!

A ≤ B

Ροή Δυσκολίας

Πάλι Αναγωγές !!!

- **Στη συνέχεια…** θα ***χρησιμοποιήσουμε*** τις **αναγωγές** για ένα *κάπως* ***διεστραμμένο*** *στόχο*:
- **A ≤ B**

Κάτω Φράγματα

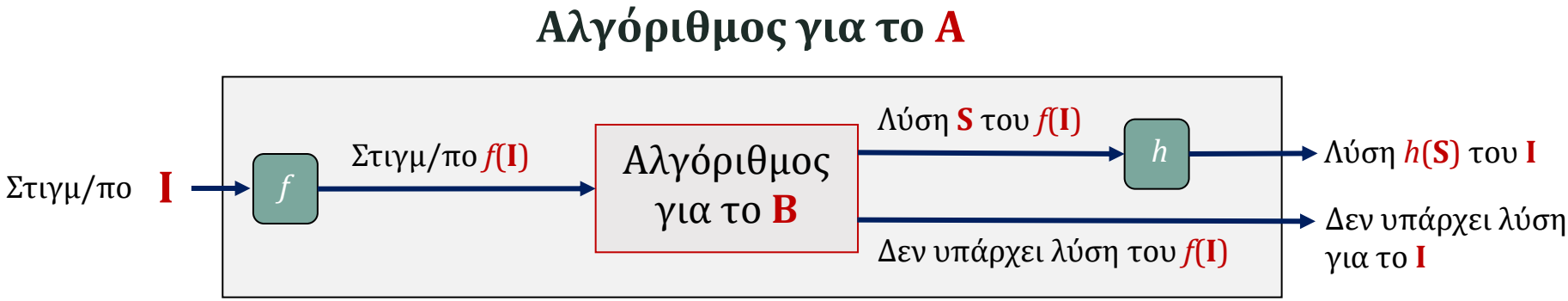
γνωρίζουμε ότι το πρόβλημα **A είναι δύσκολο**,
και ***χρησιμοποιούμε*** την αναγωγή
για **να αποδείξουμε ότι** και το **B** είναι **δύσκολο** !!!

Εάν **A ≤ B** και **B ≤ C** τότε **A ≤ C**

Ροή Δυσκολίας

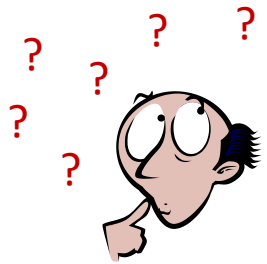
Πάλι Αναγωγές !!!

- **Ας εξειδικεύσουμε…** τον ορισμό της **αναγωγής** για **προβλήματα αναζήτησης !!!**
- **A ≤ B**

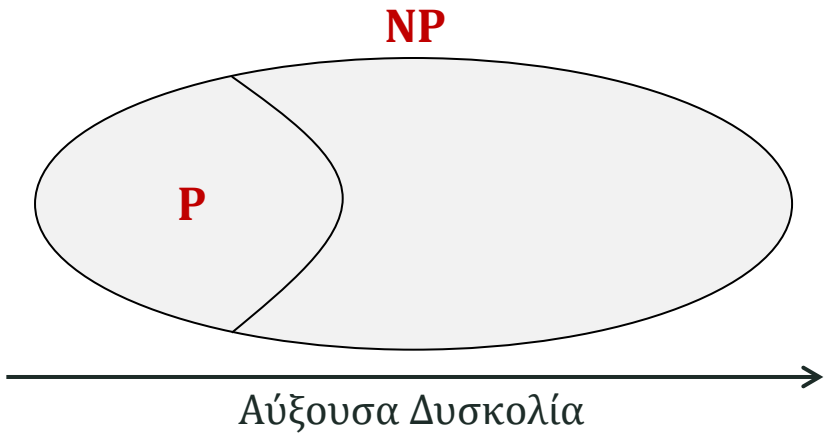


Μια αναγωγή **A ≤ B** είναι δύο πολυωνυμικοί αλγόριθμοι ***f*** και ***h***
f : ***μετασχηματίζει*** κάθε στιγμιότυπο I του A σε ένα στιγμιότυπο *f*(I) του B
h : ***απεικονίζει*** μια λύση S του στιγμιότυπου *f*(I) σε μια λύση *h*(S) του I

NP-πλήρη !!!

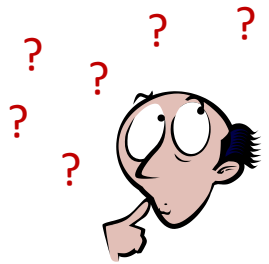


- **Και τώρα…** ορίζουμε την κλάση των **δυσκολότερων προβλημάτων αναζήτησης !**
- **A ≤ B**

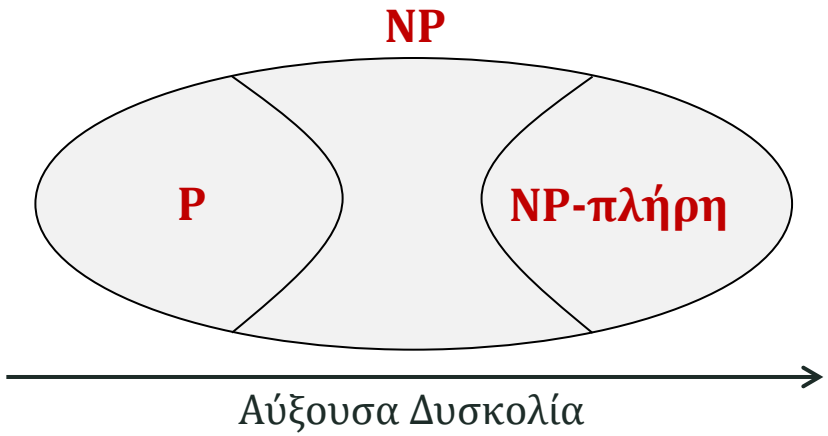


Ένα πρόβλημα αναζήτησης είναι **NP-πλήρες** (NP-complete) *εάν* ***ΟΛΑ ΤΑ ΑΛΛΑ ΠΡΟΒΛΗΜΑΤΑ ΑΝΑΖΗΤΗΣΗΣ ΣΤΟ*** **NP** ***ΑΝΑΓΟΝΤΑΙ*** *σε αυτό* **!!!**

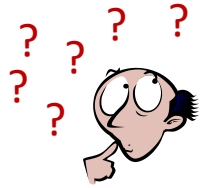
NP-πλήρη !!!



- **Και τώρα…** ορίζουμε την κλάση των **δυσκολότερων προβλημάτων αναζήτησης !**
- **Εάν P ≠ NP**



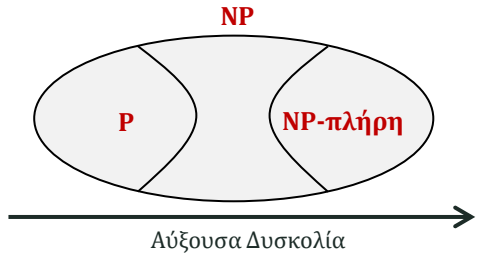
Ένα πρόβλημα αναζήτησης είναι **NP-πλήρες** (NP-complete) *εάν* ***ΟΛΑ ΤΑ ΑΛΛΑ ΠΡΟΒΛΗΜΑΤΑ ΑΝΑΖΗΤΗΣΗΣ ΣΤΟ*** **NP** ***ΑΝΑΓΟΝΤΑΙ*** *σε αυτό* **!!!**



NP-πλήρη !!!

- **Αυτή…** είναι μια **πολύ ισχυρή απαίτηση !!!**

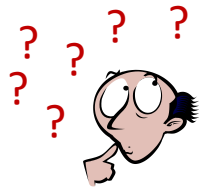
✓ Για να είναι ένα πρόβλημα **NP-πλήρες** πρέπει να είναι



Χρήσιμο για την επίλυση ***κάθε προβλήματος NP στον Κόσμο !!!***

✓ Είναι εντυπωσιακό, αλλά !!!...

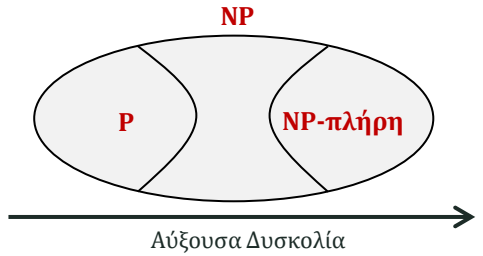
Υπάρχουν *τέτοια προβλήματα* (3SAT, TSP, ΜΕΓΙΣΤΗ ΔΙΑΔΡΟΜΗ, …)



NP-πλήρη !!!

- **Ισχύει:**

✓ Όλα τα **NP-πλήρη** πρόβλημα **ανάγονται** το ένα στο άλλο **!!!**

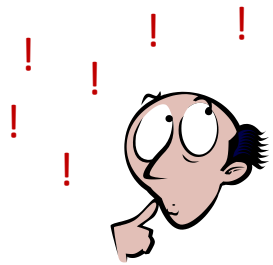


SAT ≤ ··· ≤ Q ≤ ··· ≤ R

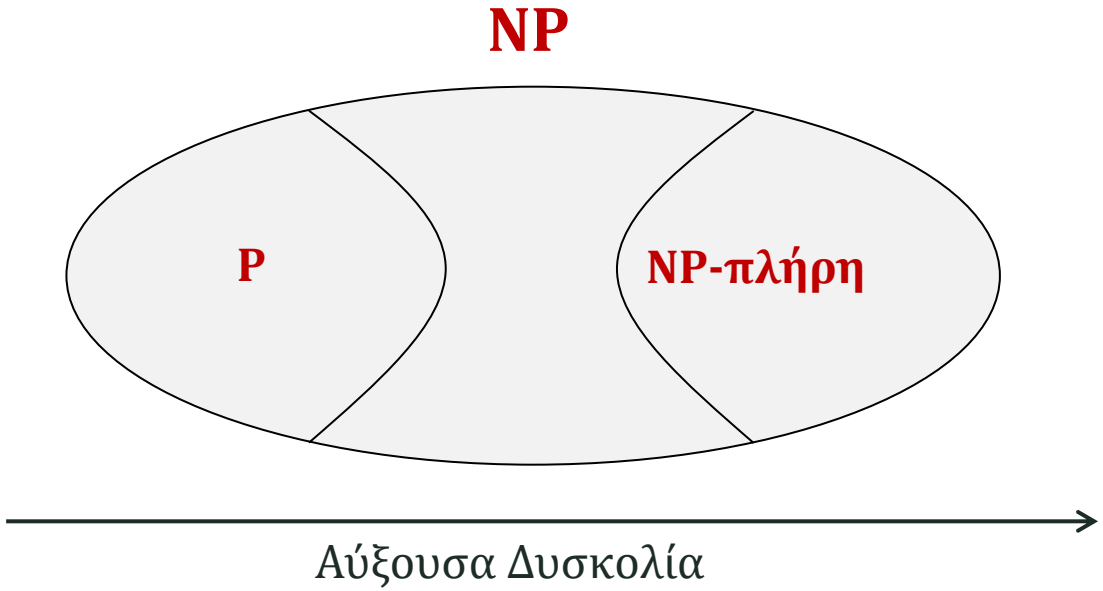
✓ Και, όλα τα **NP** προβλήματα **ανάγονται** στα **NP-πλήρη !!!**

ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-ΣΤΟ-NP ≤ SAT

P, NP και NP-πλήρη !!!

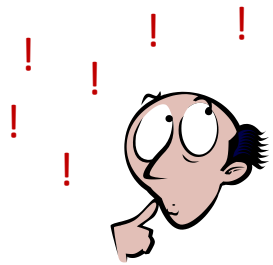


- Εάν **P ≠ NP**

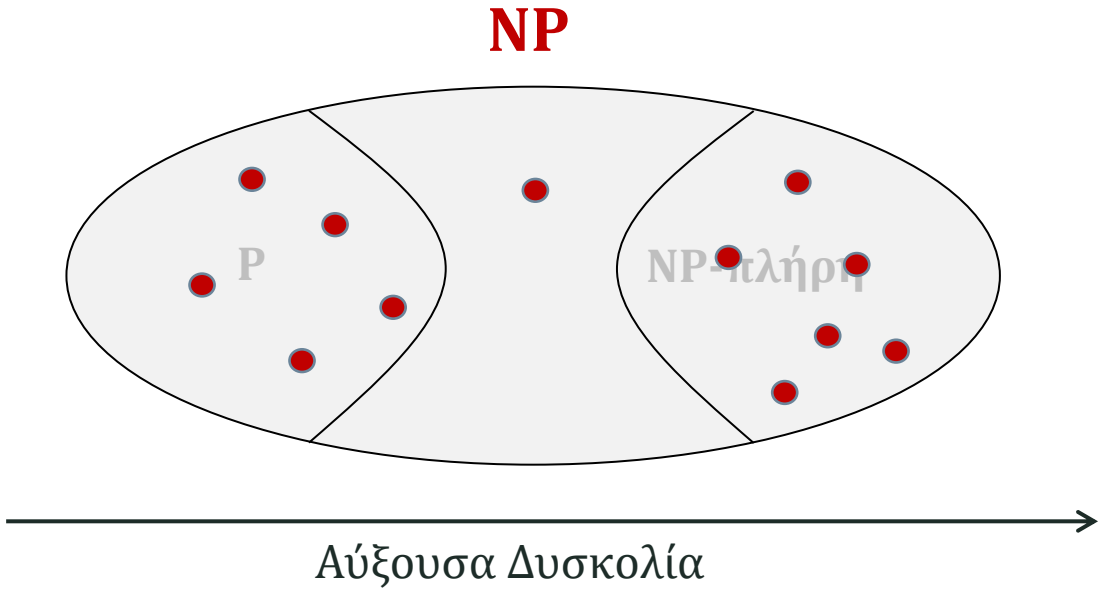


Μια Σχηματική Παρουσίαση !!!

P, NP και NP-πλήρη !!!

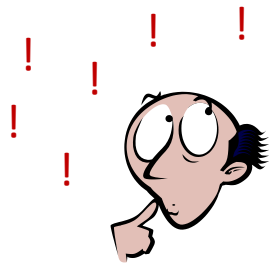


- Εάν **P ≠ NP**

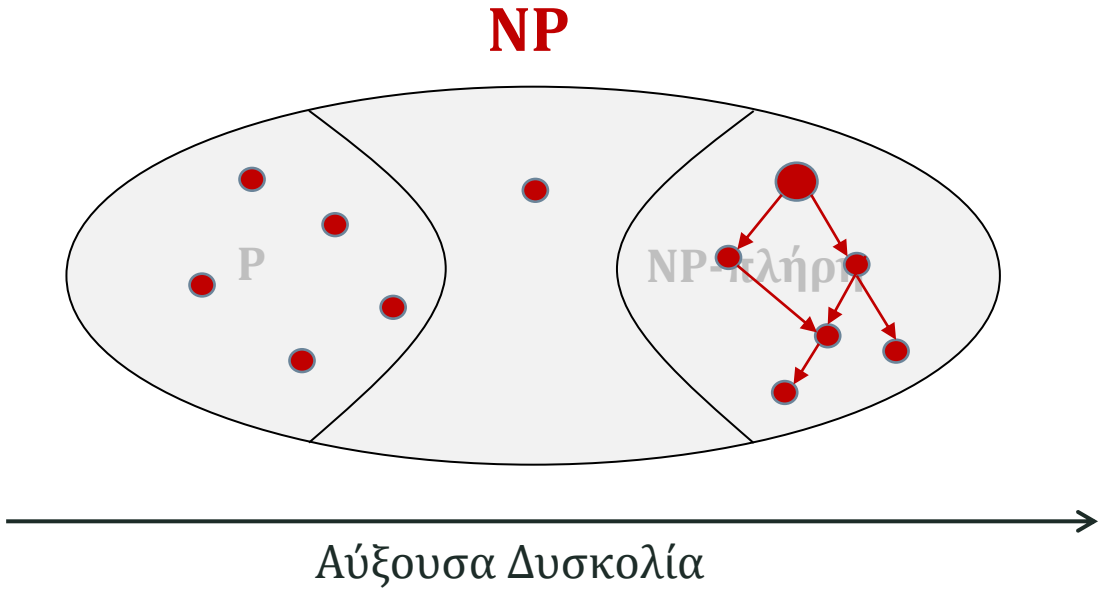


ΟΛΑ ΤΑ ΠΡΟΒΛΗΜΑΤΑ **NP**

P, NP και NP-πλήρη !!!

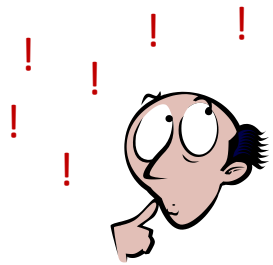


- Εάν **P ≠ NP**

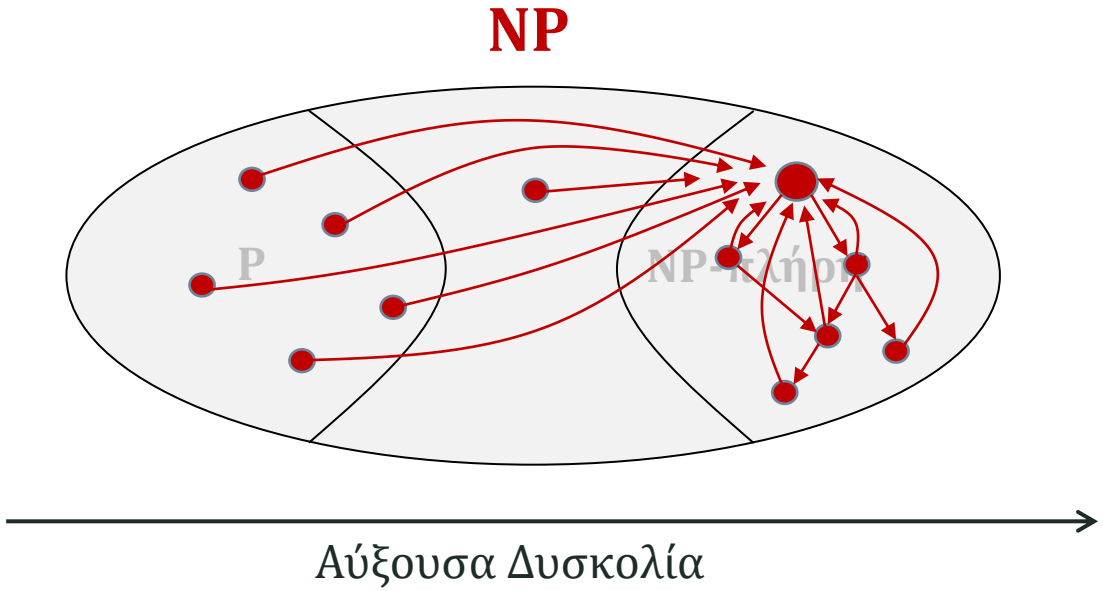


SAT ≤ ⋯ ≤ Q ≤ ⋯ ≤ R

P, NP και NP-πλήρη !!!

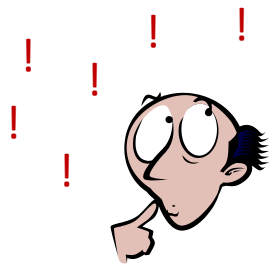


- Εάν **P ≠ NP**

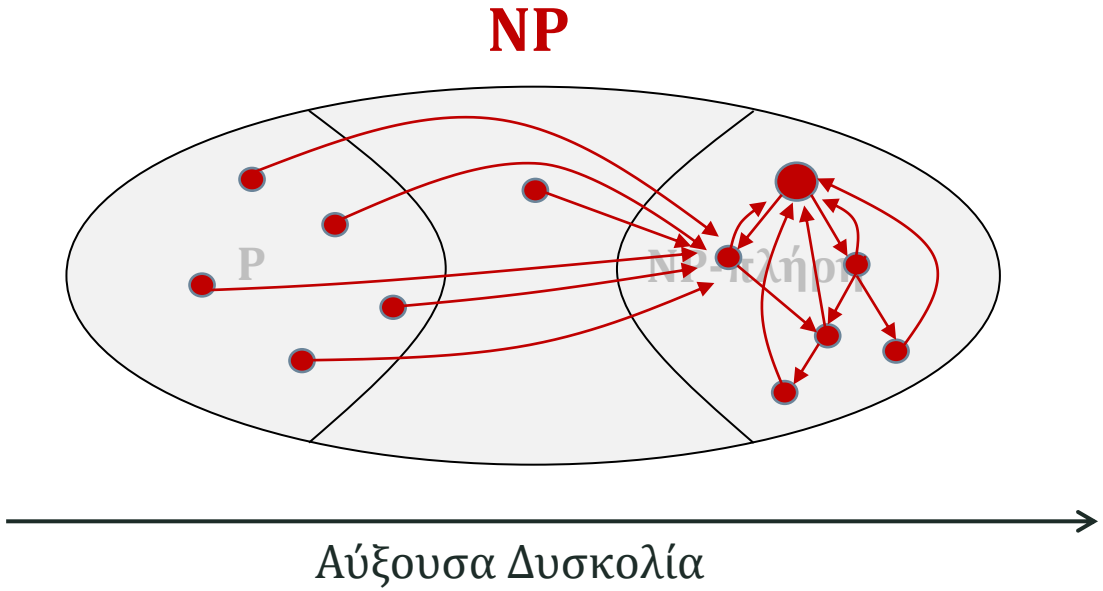


ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-ΣΤΟ-NP ≤ SAT ≤ ⋯ ≤ Q ≤ ⋯ ≤ R

P, NP και NP-πλήρη !!!

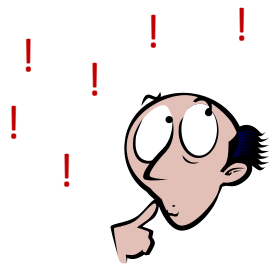


- Εάν **P ≠ NP**

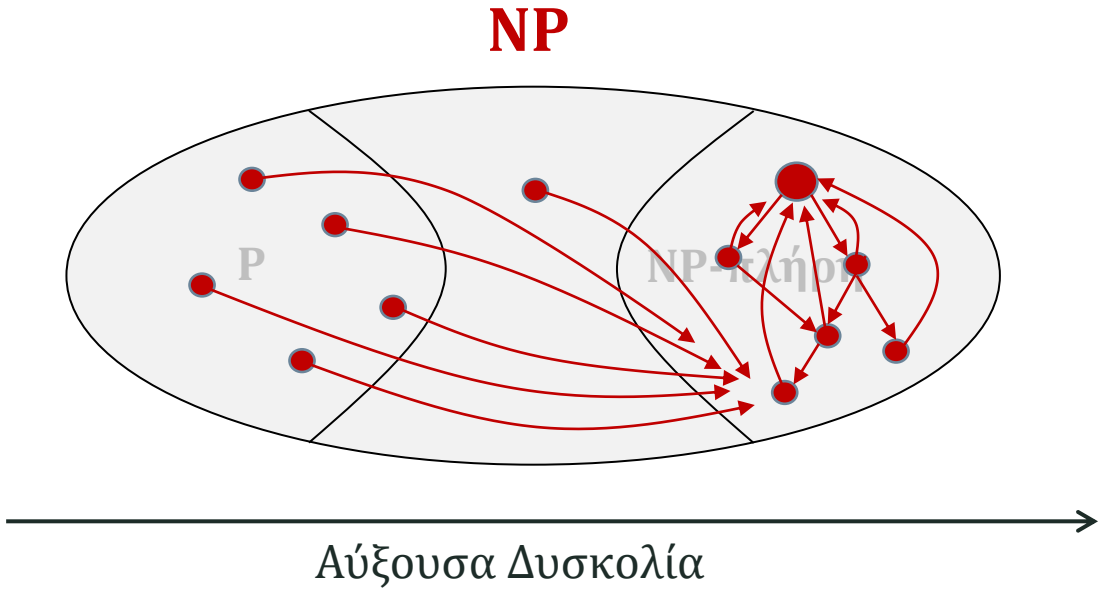


ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-ΣΤΟ-NP ≤ SAT ≤ ⋯ ≤ Q ≤ ⋯ ≤ R

P, NP και NP-πλήρη !!!



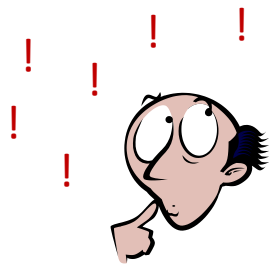
- Εάν **P ≠ NP**



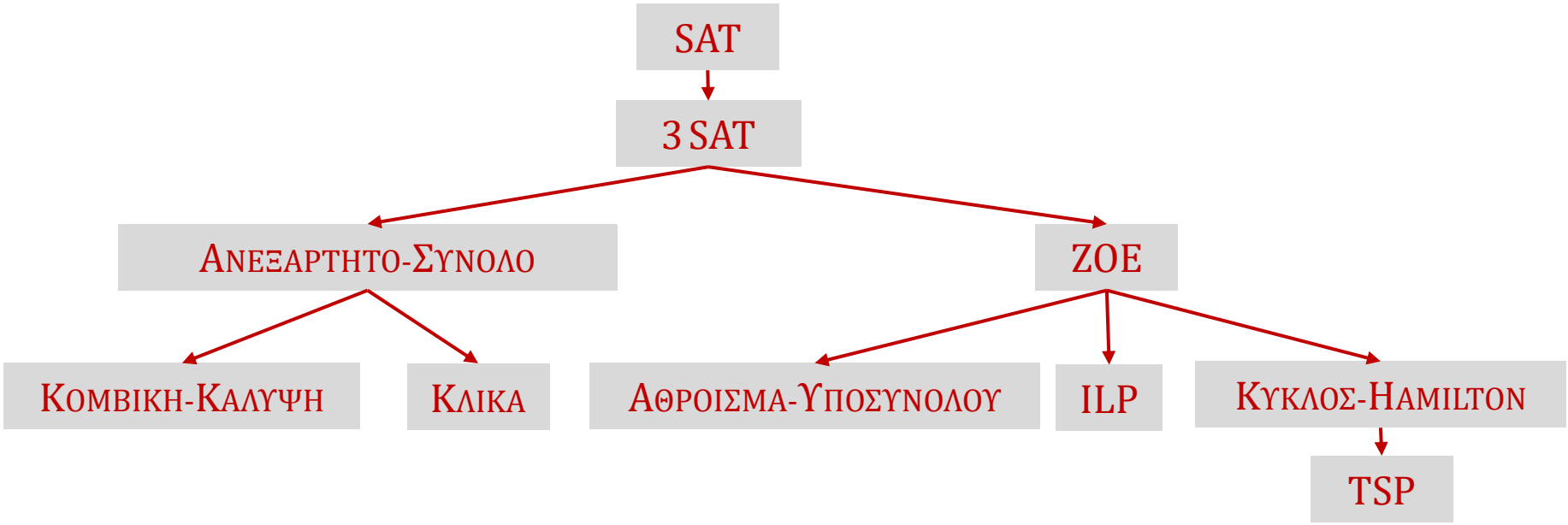
ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-ΣΤΟ-NP ≤ SAT ≤ ⋯ ≤ Q ≤ ⋯ ≤ R

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



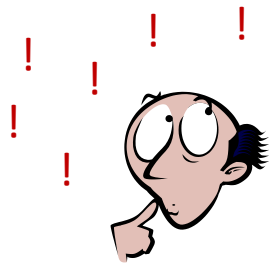
Οποιοδηποτε-Προβλημα-**NP**



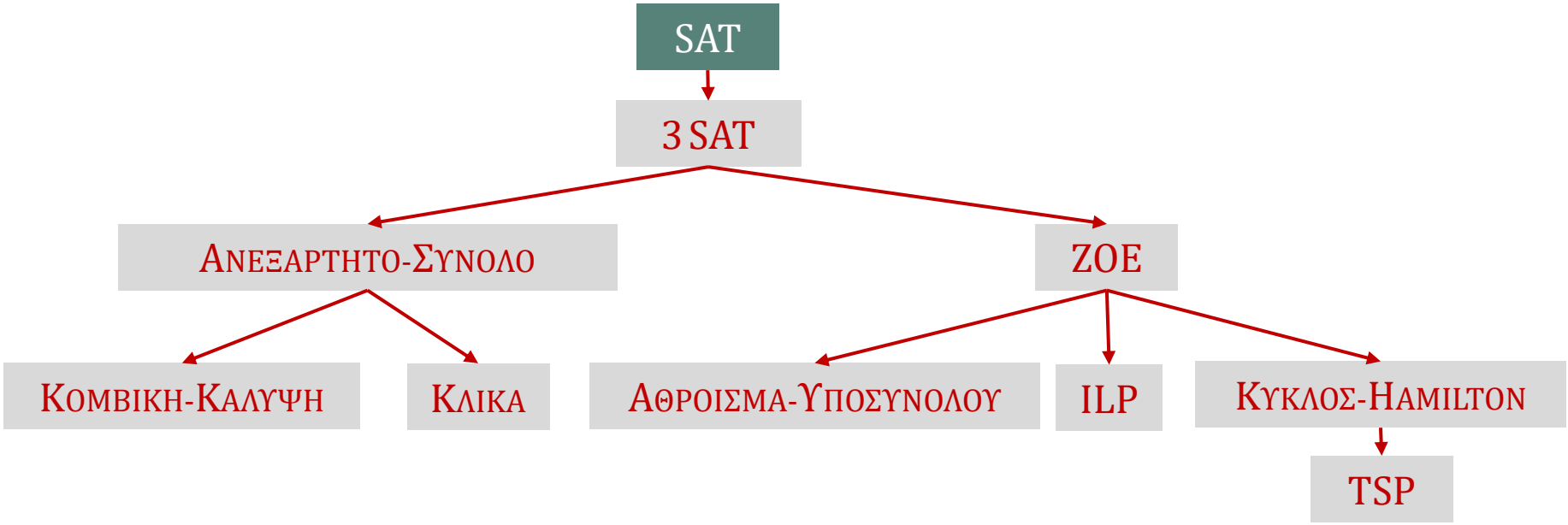
Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



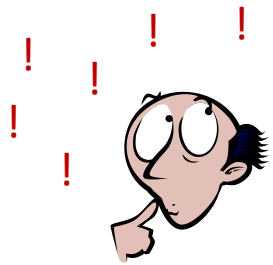
ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-**NP**



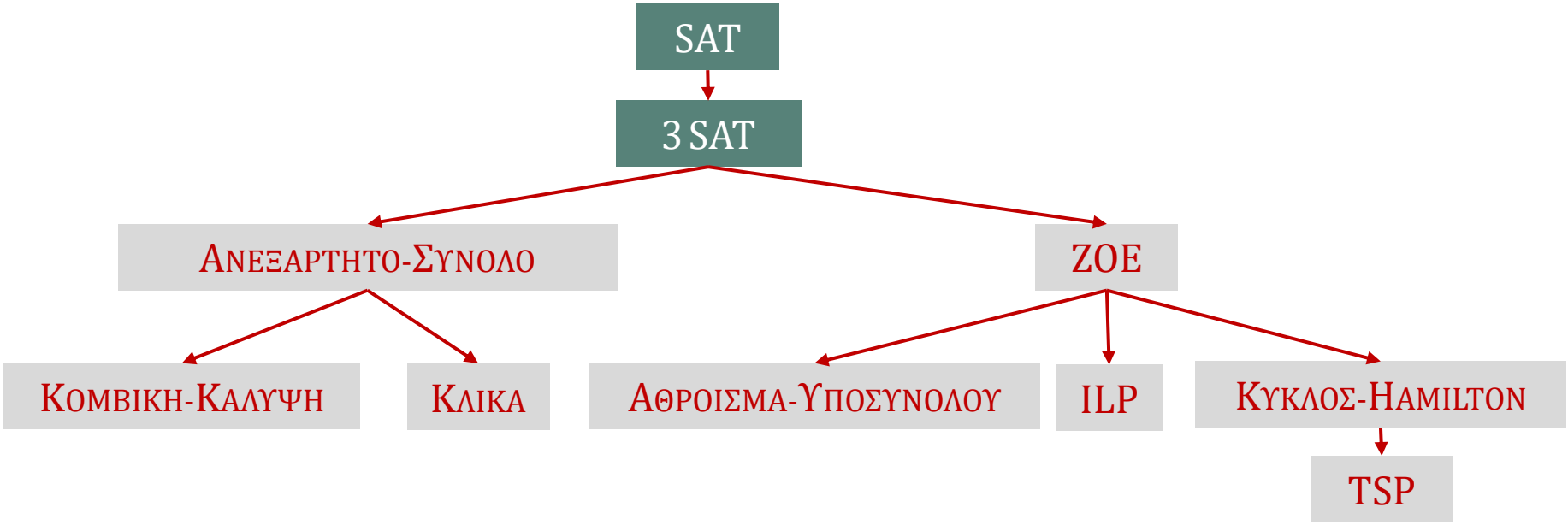
Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



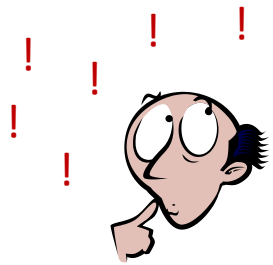
Οποιοδηποτε-Προβλημα-**NP**



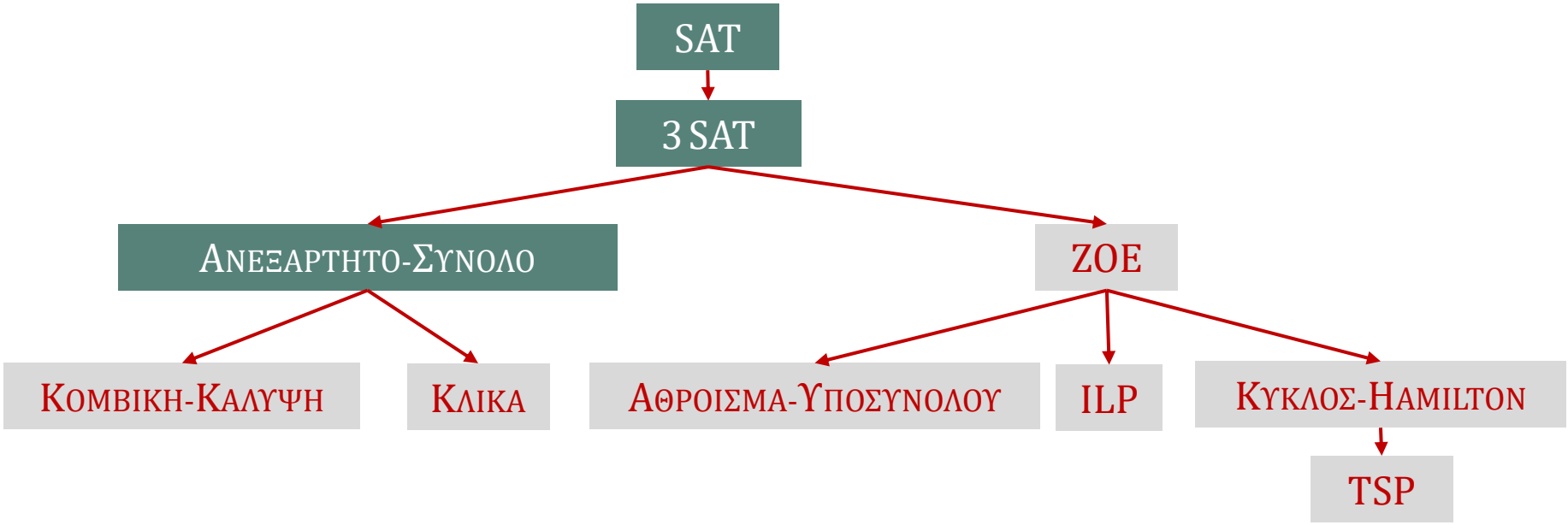
Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



Οποιοδηποτε-Προβλημα-**NP**

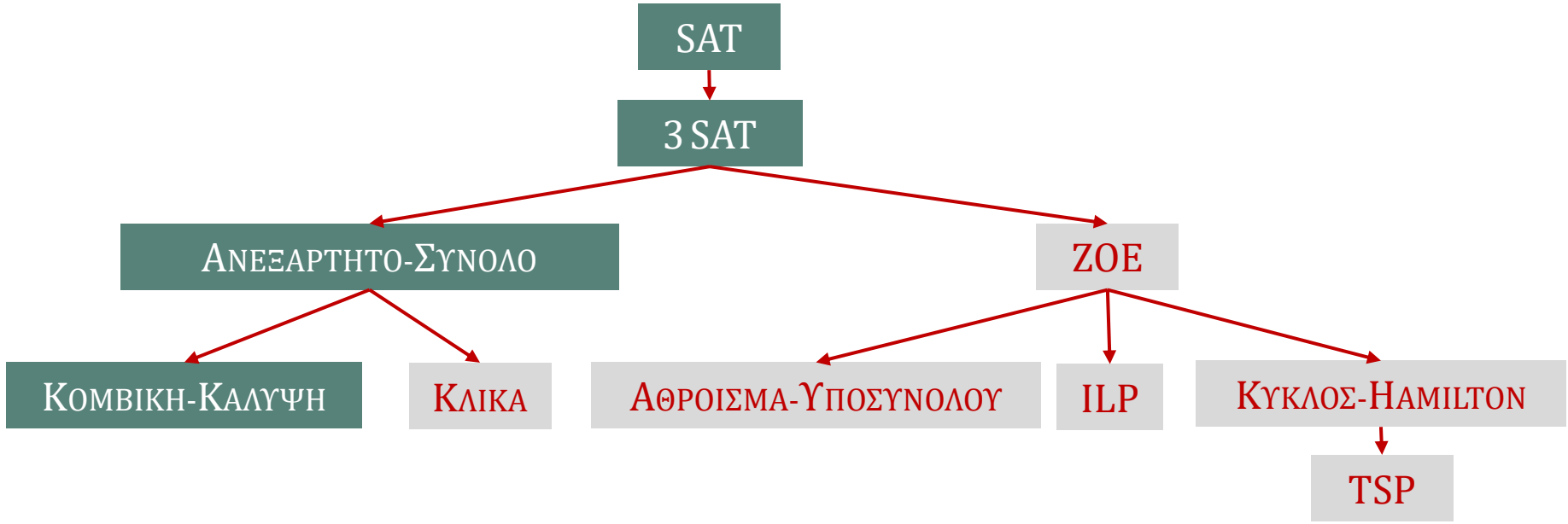


Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$

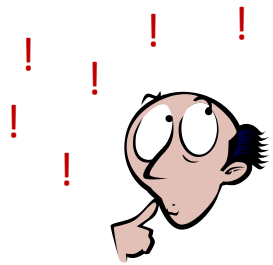
Οποιοδηποτε-Προβλημα-**NP**



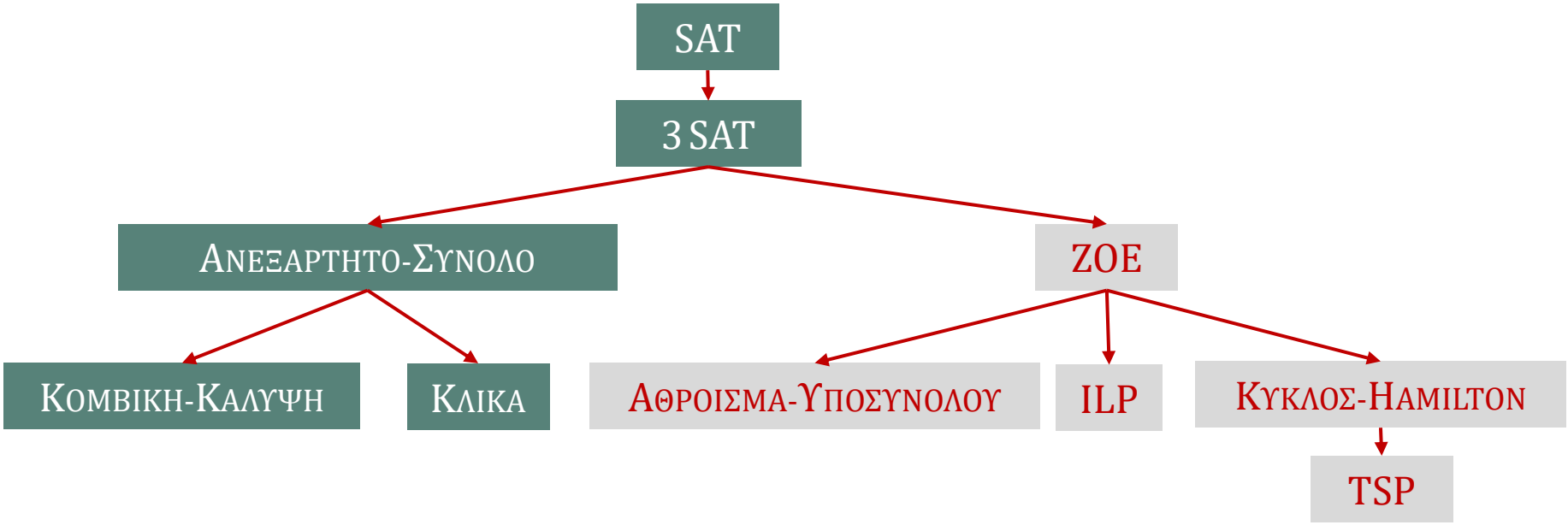
Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



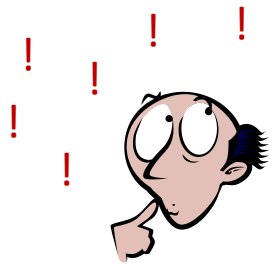
Οποιοδηποτε-Προβλημα-**NP**



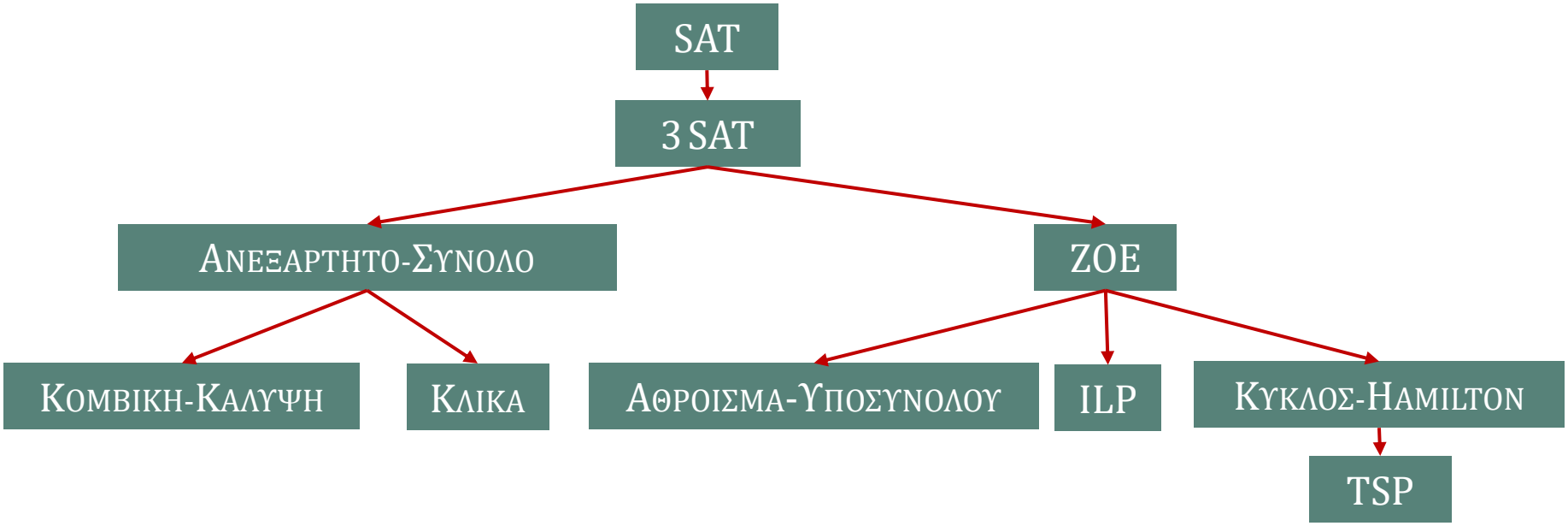
Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



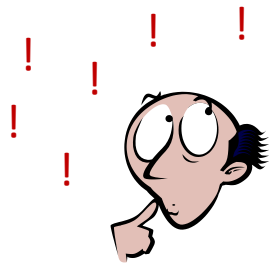
Οποιοδηποτε-Προβλημα-**NP**

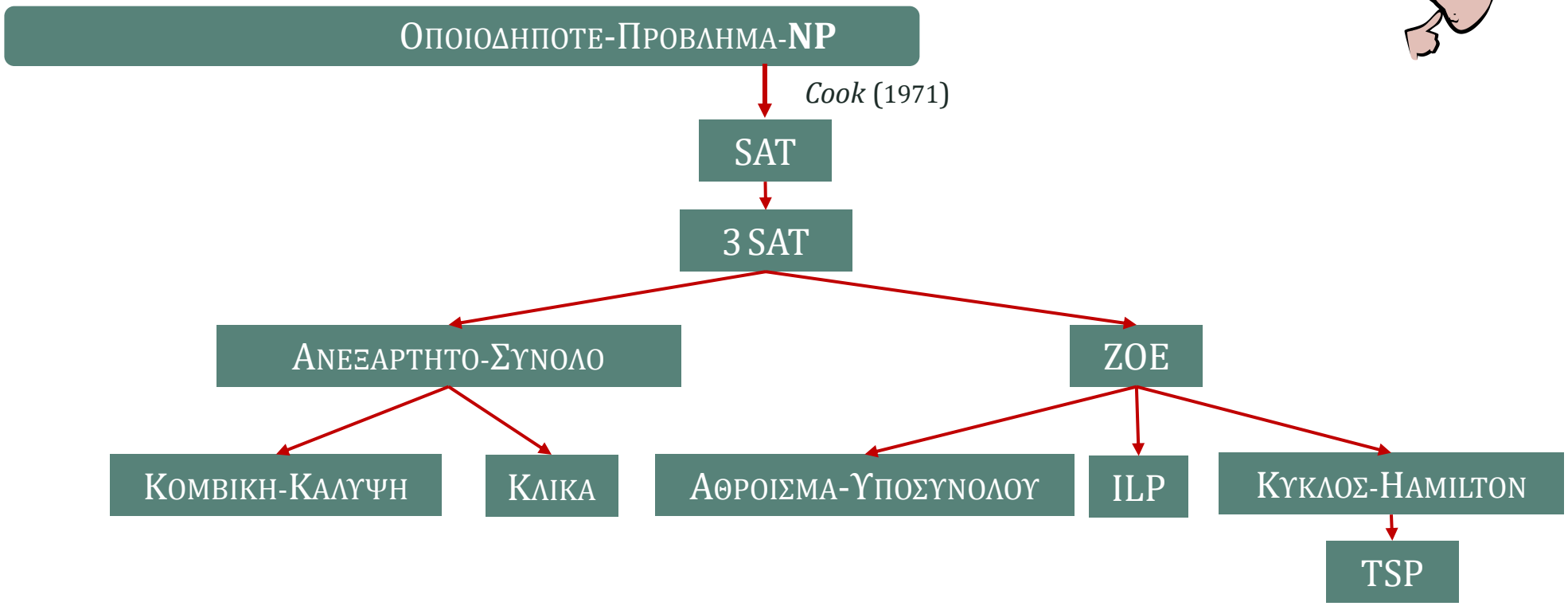


Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$

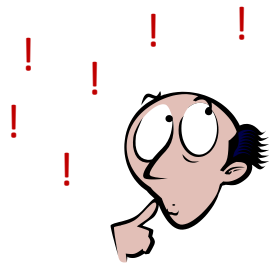


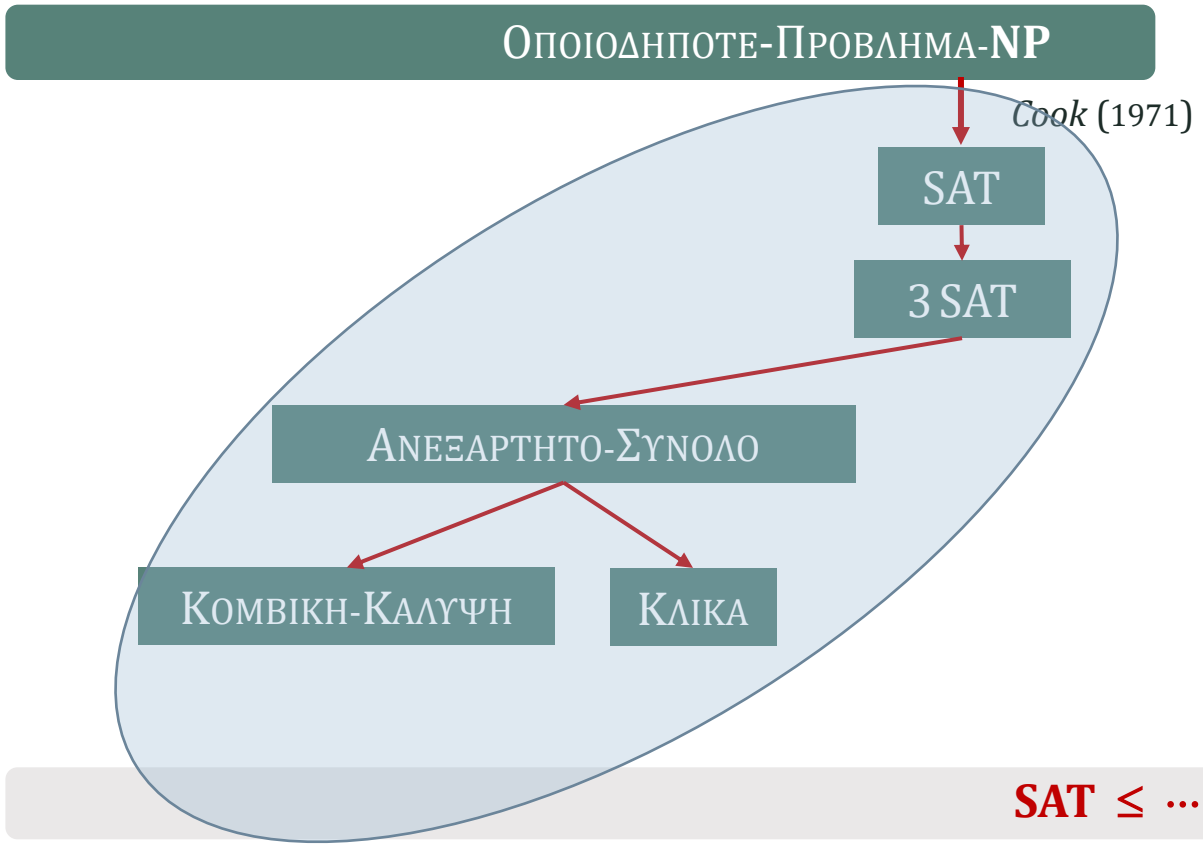


Θα δείξουμε μέσω Αναγωγών !!!

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



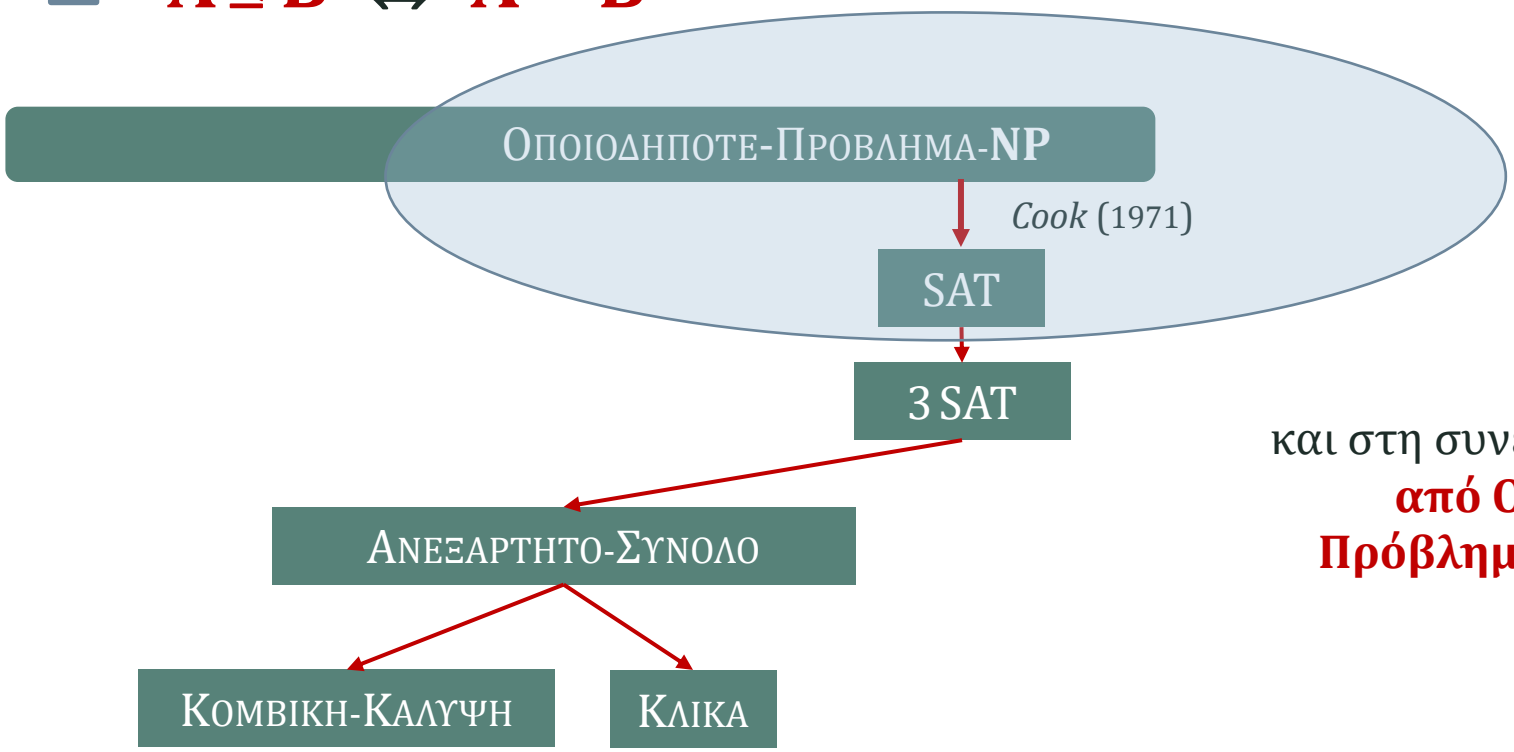


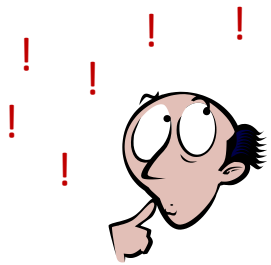
Αρχικά θα δείξουμε αυτές τις **4 Αναγωγές** !!!

$$SAT \leq \cdots \leq Q \leq \cdots \leq R$$

Αναγωγές Προβλημάτων NP

- $A \leq B \Leftrightarrow A \rightarrow B$



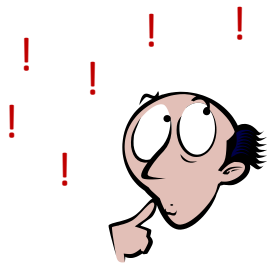


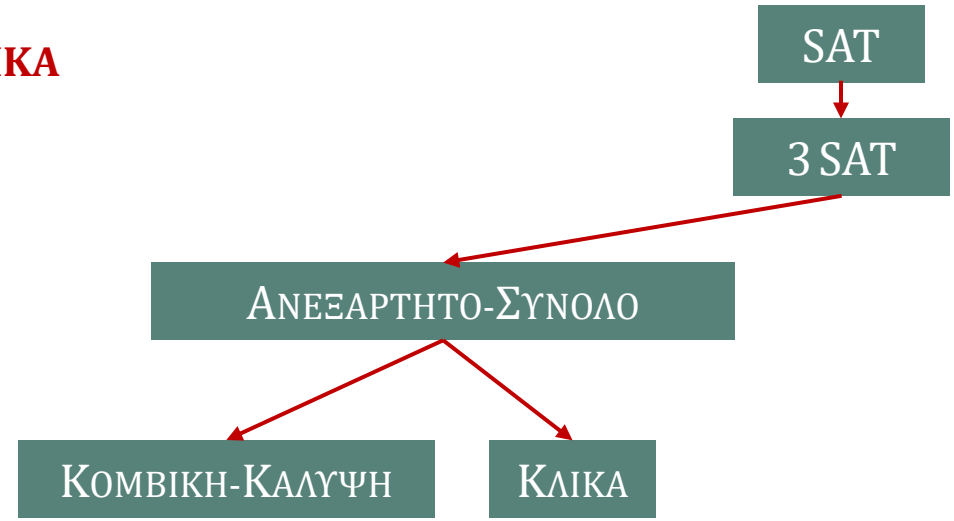
και στη συνέχεια την αναγωγή:
από Οποιοδήποτε Πρόβλημα NP στο SAT !!!

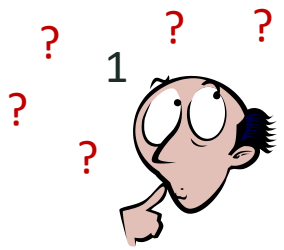
ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-ΣΤΟ-NP ≤ SAT ≤ ⋯ ≤ Q ≤ ⋯ ≤ R

Αναγωγές Προβλημάτων NP

1. SAT ⟶ 3SAT
2. 3SAT ⟶ ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ
3. ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ
4. ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΛΙΚΑ







1 SAT → 3SAT

- **Το πρόβλημα SAT**

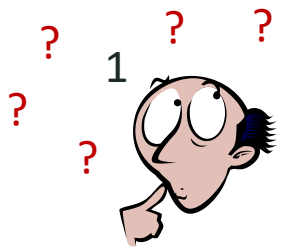
  Μας δίδεται ένας λογικός τύπος (Boolean formula) σε συζευκτική μορφή:

  $$(x \vee y \vee z \vee \bar{w}) \wedge (\bar{x} \vee y) \wedge (\bar{z} \vee x \vee \bar{w} \vee \bar{p} \vee y) \wedge (x \vee \bar{y} \vee z)$$

  και μας ζητείται είτε να βρούμε μια *ικανοποιούσα ανάθεση τιμών αληθείας* (satisfying truth assignment) ή να αναφέρουμε ότι δεν υπάρχει καμία !!!

- **3SAT**

  $$(x \vee y \vee z) \wedge (\bar{w} \vee y) \wedge (\bar{y} \vee z) \wedge (p \vee \bar{x}) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$$



1 SAT → 3SAT

Τέχνασμα για την Αναγωγή του SAT στο 3SAT:

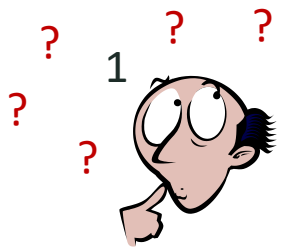
- Με δεδομένο ένα στιγμιότυπο **I** του SAT, χρησιμοποιούμε ακριβώς το ίδιο στιγμιότυπο για το 3SAT, με την διαφορά ότι ένας όρος με **k > 3** στοιχεία

$$(a_1 \vee a_2 \vee \ldots \vee a_k)$$

αντικαθίσταται από ένα σύνολο όρων

$$(a_1 \vee a_2 \vee y_1) \wedge (\overline{y}_1 \vee a_3 \vee y_2) \wedge (\overline{y}_2 \vee a_4 \vee y_3) \wedge \ldots \wedge (\overline{y}_{k-3} \vee a_{k-1} \vee a_k)$$

όπου τα $y_1, y_2, \ldots, y_{k-3}$ είναι νέες μεταβλητές.



1 SAT → 3SAT

Τέχνασμα για την Αναγωγή του SAT στο 3SAT:

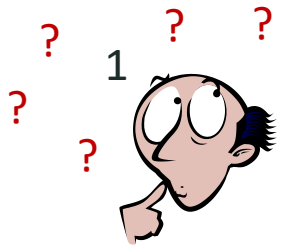
- Έστω **I′** το στιγμιότυπο του 3SAT που προκύπτει... προφανώς σε χρόνο T(|I|) !!!
- Στιγμιότυπο **I′** είναι ισοδύναμο με **I** ως προς την **ικανοποιησιμότητα** !!!

διότι για οποιαδήποτε ανάθεση τιμών στα $\mathbf{a_i}$ $(1 \le i \le k)$, ισχύει:

$(a_1 \vee a_2 \vee \ldots \vee a_k)$ ικανοποιείται $\Leftrightarrow$

υπάρχει μια ανάθεση τιμών των $y_1, y_2, \ldots, y_{k-3}$ για την οποία οι όροι

$(a_1 \vee a_2 \vee y_1) \wedge (\overline{y}_1 \vee a_3 \vee y_2) \wedge \ldots \wedge (\overline{y}_{k-3} \vee a_{k-1} \vee a_k)$ ικανοποιούνται όλοι



1 SAT → 3SAT

Τέχνασμα για την Αναγωγή του SAT στο 3SAT:

- Για να το δείτε… υποθέστε ότι **ικανοποιούνται όλοι οι όροι στα δεξιά**.

Τότε, τουλάχιστον ένα από τα $\mathbf{a_1, a_2, \ldots, a_k}$ πρέπει να είναι **true** !!!

$(a_1 \vee a_2 \vee \ldots \vee a_k)$

ικανοποιείται

$\Leftrightarrow$

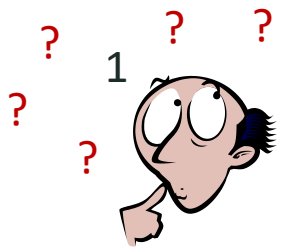
διαφορετικά το $\mathbf{y_1}$ θα πρέπει να είναι true, το οποίο αναγκάζει το $\mathbf{y_2}$ να είναι true, … κάνοντας false τον τελευταίο όρο !!!

Άρα, ικανοποιείται επίσης και ο αριστερός όρος $(a_1 \vee a_2 \vee \ldots \vee a_k)$

υπάρχει μια ανάθεση τιμών των $y_1, y_2, \ldots, y_{k-3}$ για την οποία οι όροι

$$(a_1 \vee a_2 \vee y_1) \wedge (\overline{y}_1 \vee a_3 \vee y_2) \wedge \ldots \wedge (\overline{y}_{k-3} \vee a_{k-1} \vee a_k)$$

έστω ότι ικανοποιούνται όλοι



1 SAT → 3SAT

Τέχνασμα για την Αναγωγή του SAT στο 3SAT:

- Αντιστρόφως… υποθέστε ότι **ικανοποιείται ο όρος** $(a_1 \vee a_2 \vee \ldots \vee a_k)$

Τότε, κάποια a_i πρέπει να είναι **true** !!!

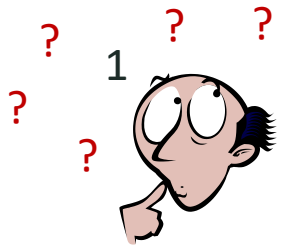
Θέτουμε τιμές **true** στις $y_1, y_2, \ldots, y_{i-2}$ και στις $y_{i-1}, y_i, \ldots, y_{k-3}$ τιμές **false** !!!

$(a_1 \vee a_2 \vee \ldots \vee a_k)$ ικανοποιείται $\Leftrightarrow$ Αυτό διασφαλίζει ότι **ικανοποιούνται** όλοι οι όροι δεξιά !!!

υπάρχει μια ανάθεση τιμών των $y_1, y_2, \ldots, y_{k-3}$ για την οποία οι όροι

$$(a_1 \vee a_2 \vee y_1) \wedge (\bar{y}_1 \vee a_3 \vee y_2) \wedge \ldots \wedge (\bar{y}_{k-3} \vee a_{k-1} \vee a_k)$$

ικανοποιούνται όλοι



2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

- **3SAT**

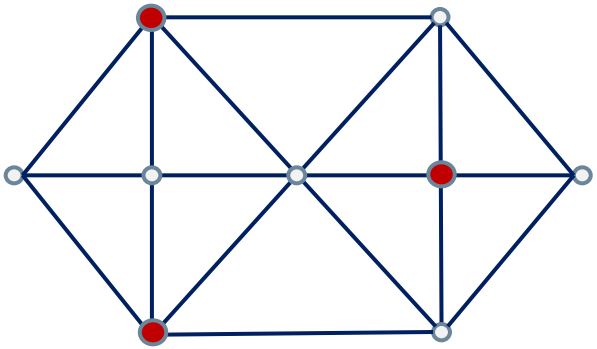
$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$

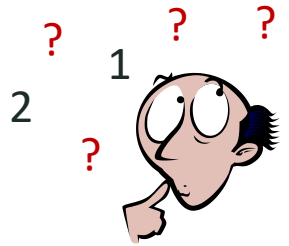
- **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**

  Μας δίδεται γράφημα **n** κόμβων και *ακέραιος* **k**

  και μας ζητείται να βρούμε ένα σύνολο **IS** με **k** κόμβους οι οποίοι είναι ανεξάρτητοι, δηλ. ανά δύο δεν έχουν ακμή μεταξύ τους,

  ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!



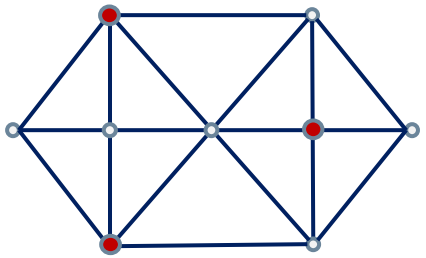


2 3SAT → Ανεξαρτητο-Συνολο

- **Δύο πολύ διαφορετικά προβλήματα !!!**

$(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y})$

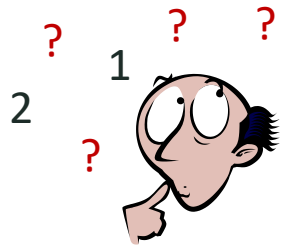
- Συσχετισμός **Λογικής Boole** με **Γραφήματα** ?



- **Ας σκεφτούμε…**

  Σε μια *ικανοποιούσα ανάθεση τιμών αληθείας* του 3SAT θα πρέπει **ένα στοιχείο** από **κάθε όρο** να είναι **true** !!!

- **Συνεπείς επιλογές…** εάν **επιλέξουμε** το **x** να είναι **true** σε **κάποιο όρο** δεν μπορούμε να **επιλέξουμε** το $\overline{x}$ να είναι **true** σε **κάποιο άλλο** !!!

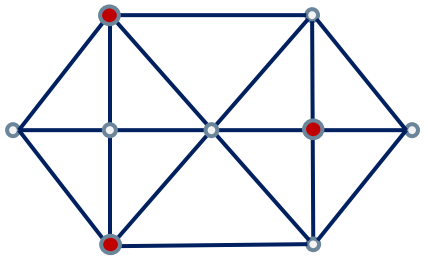


2 3SAT → Ανεξαρτητο-Συνολο

- **Δύο πολύ διαφορετικά προβλήματα !!!**

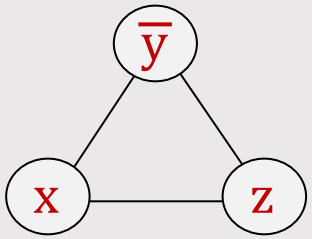
$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$

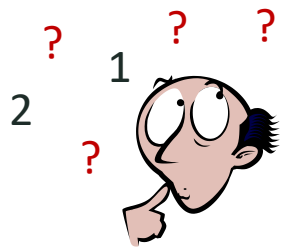
- Συσχετισμός **Λογικής Boole** με **Γραφήματα** ?



ΑΝΑΠΑΡΑΣΤΑΣΗ

$(x \vee \bar{y} \vee z)$ →

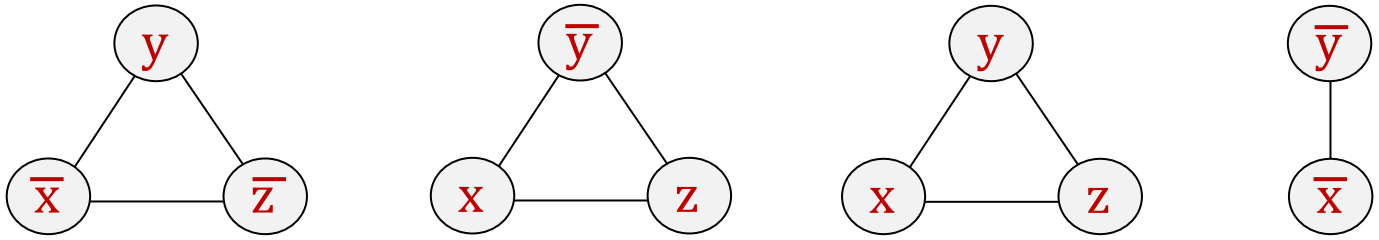




2 3SAT → Ανεξαρτητο-Συνολο

- **Παράδειγμα**

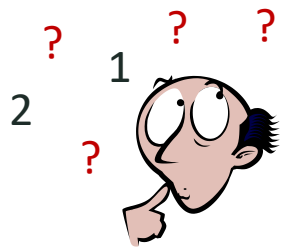
$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



- Στο **γράφημα G** που προκύπτει, ένα **IS** περιέχει *το πολύ ένα στοιχείο από κάθε* ***τρίγωνο–όρο*** !!!

- Για να επιβάλουμε **ΜΙΑ επιλογή από κάθε τρίγωνο-όρο**, θέτουμε:

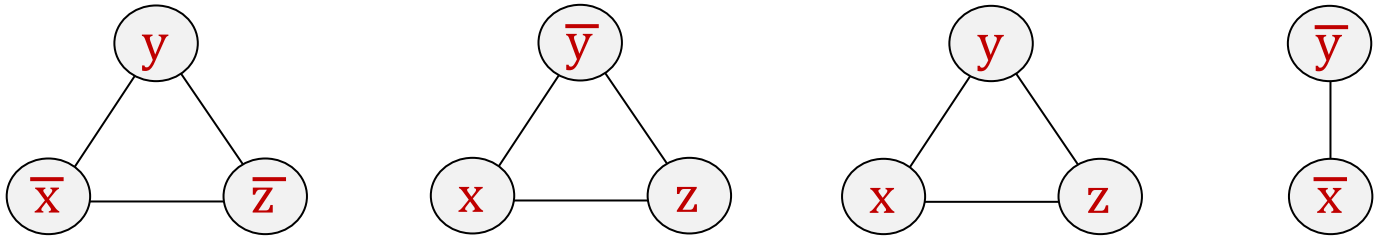
k = πλήθος όρων



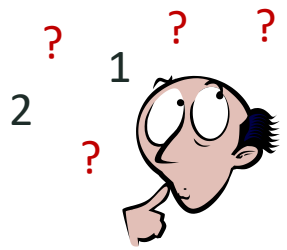
2 3SAT → Ανεξαρτητο-Συνολο

- **Παράδειγμα**

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



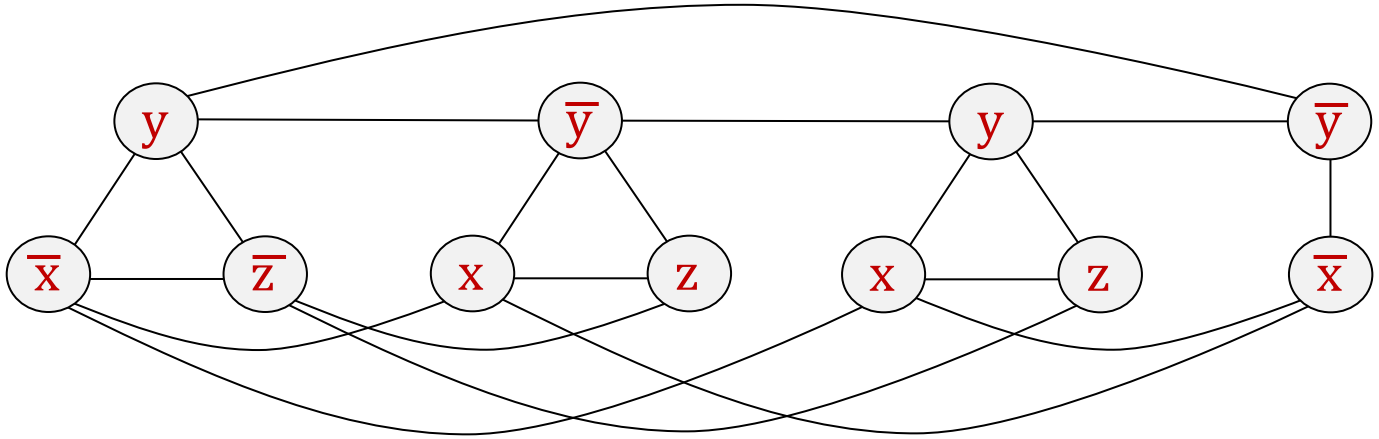
- **Λείπει κάτι ?...** **ΝΑΙ**, ένας τρόπος που θα μας **εμποδίζει να επιλέγουμε *αντίθετα στοιχεία*** - και το **x** και το $\bar{x}$!!!
- **Αυτό είναι πολύ εύκολο !!!**



2 3SAT → Ανεξαρτητο-Συνολο

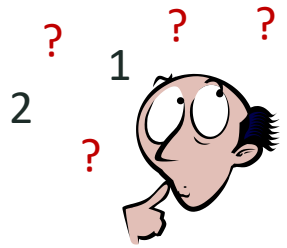
- **Παράδειγμα**

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



- **Αυτό είναι πολύ εύκολο !!!**

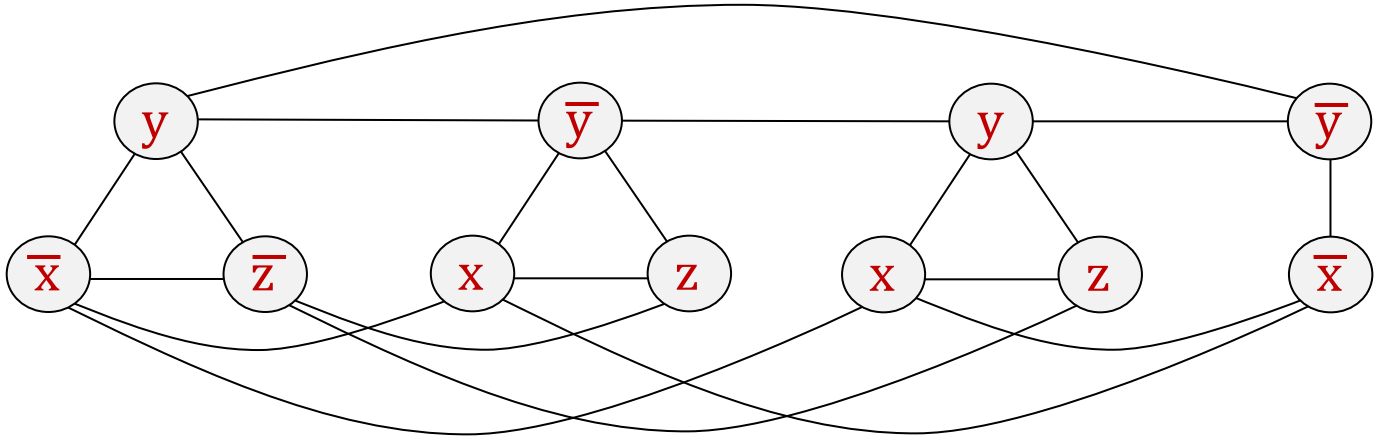
Ορίστε και ο Τρόπος !!!



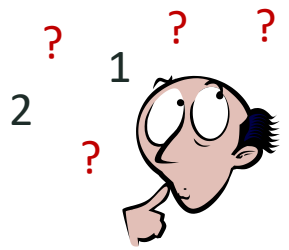
2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

- **Παράδειγμα**

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



- Με δεδομένο ένα στιγμιότυπο **I** του προβλήματος **3SAT** δημιουργήσαμε ένα στιγμιότυπο **(G, k)** του προβλήματος **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**

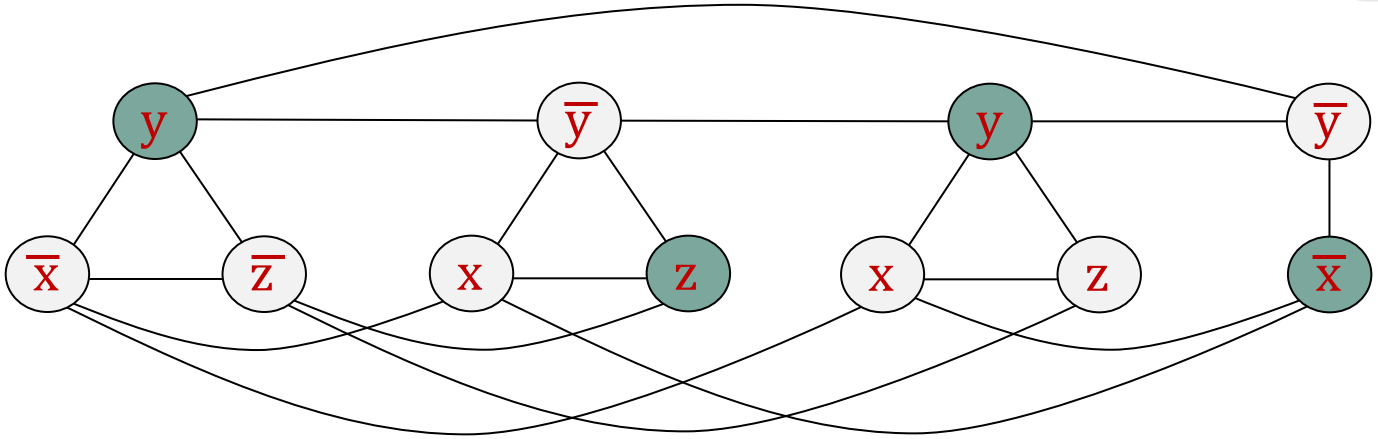


2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

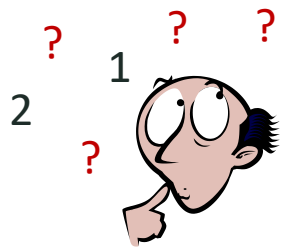
- **Παράδειγμα**

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$

x = false
y = true
z = true



- Με δεδομένο ένα στιγμιότυπο **I** του προβλήματος **3SAT** δημιουργήσαμε ένα στιγμιότυπο **(G, k)** του προβλήματος **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**

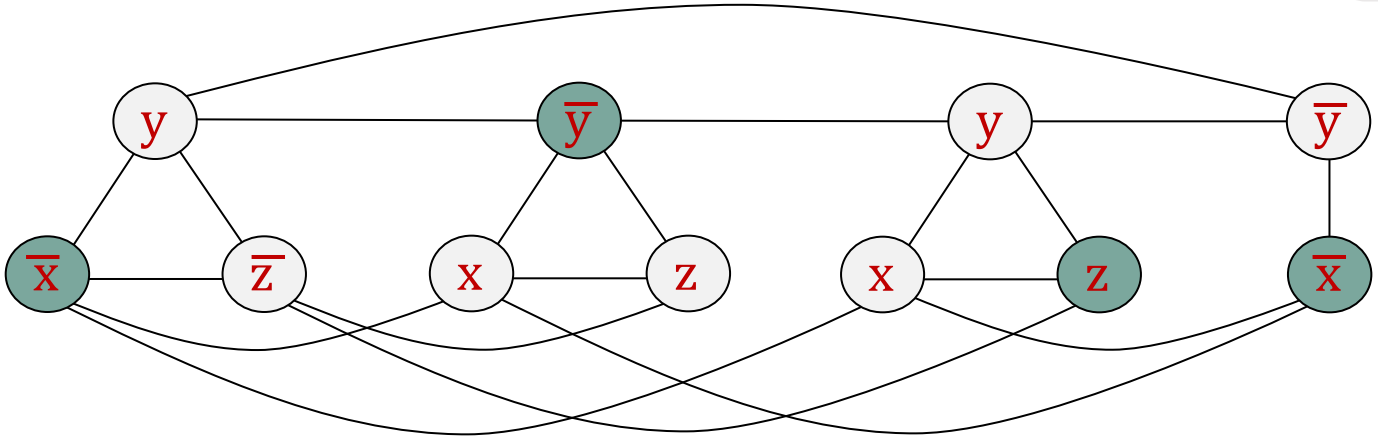


2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

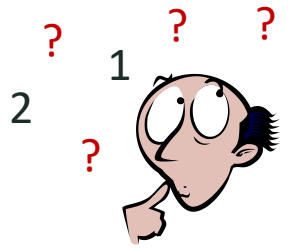
- **Παράδειγμα**

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$

x = false
y = false
z = true



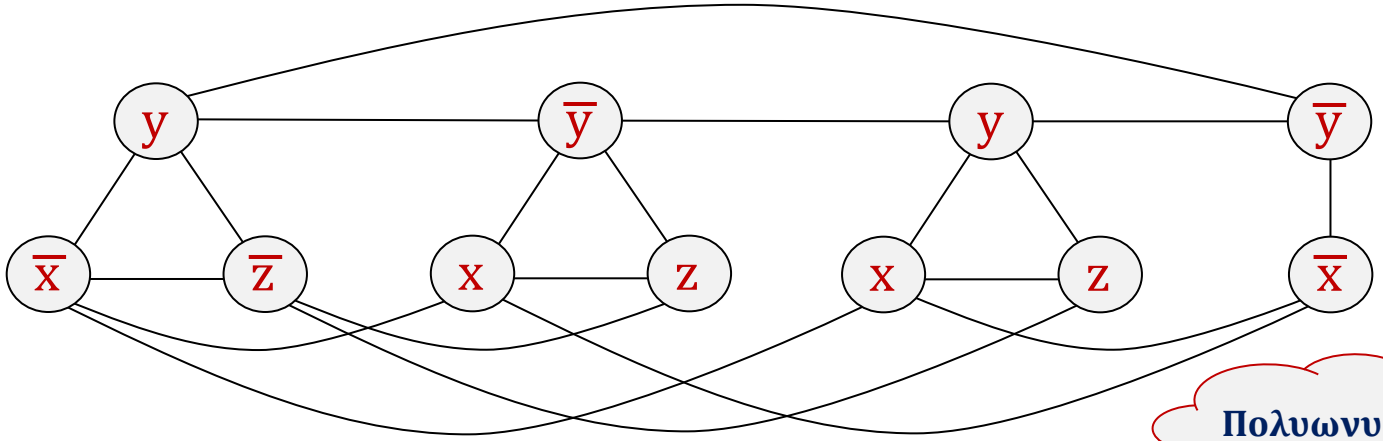
- Με δεδομένο ένα στιγμιότυπο **I** του προβλήματος **3SAT** δημιουργήσαμε ένα στιγμιότυπο **(G, k)** του προβλήματος **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**



2 3SAT → Ανεξαρτητο-Συνολο

- **Παράδειγμα**

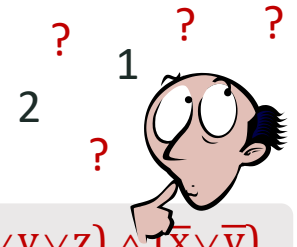
$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



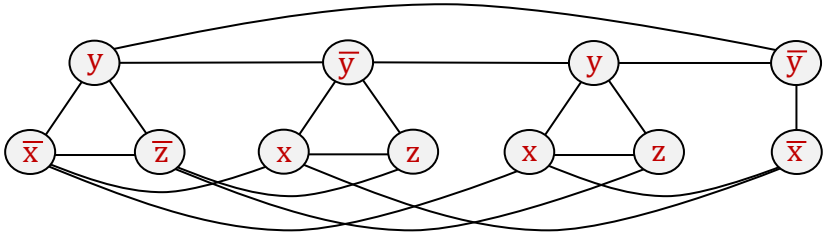
Πολυωνυμικό Χρόνο

- Με δεδομένο ένα στιγμιότυπο **I** του προβλήματος **3SAT** δημιουργήσαμε ένα στιγμιότυπο **(G, k)** του προβλήματος **Ανεξαρτητο-Συνολο**

2 3SAT → Ανεξαρτητο-Συνολο



$(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y})$

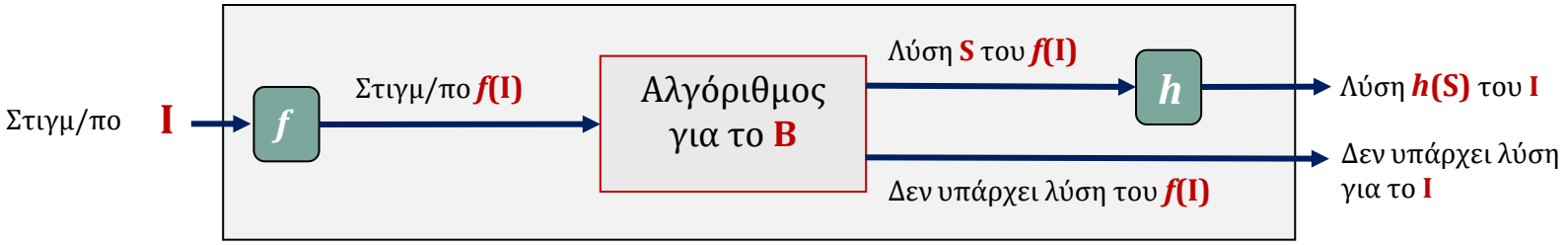


- **Θυμηθείτε !!!**

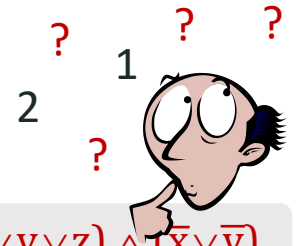
  Στις αναγωγές **δεν χρειαζόμαστε μόνο** τρόπο απεικόνισης στιγμιότυπων **I** του **A** σε στιγμιότυπα ***f*(I)** του **B** !!!

  Αλλά και έναν τρόπο **ανακατασκευής** μιας λύσης ***h*(I)** του **I** από οποιαδήποτε λύση **S** του ***f*(I),** δηλ. τη συνάρτηση ***h***

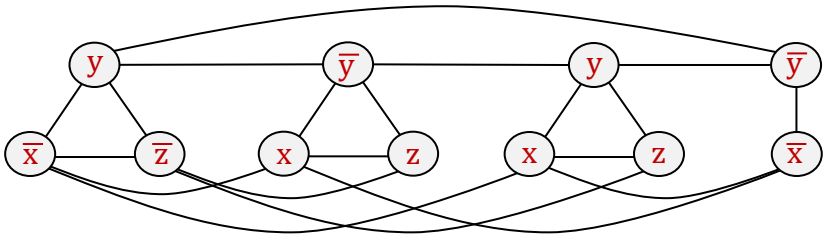
Αλγόριθμος για το A



2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ



$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$

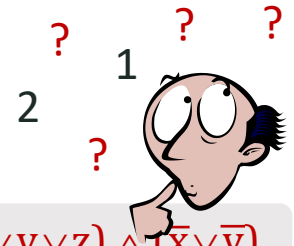


- **Όπως πάντα !!!**

  Υπάρχουν δύο πράγματα για να **αποδείξουμε** !!!

(1) Με δεδομένο ένα ***ανεξάρτητο σύνολο*** **S** με **k** κόμβους στο γράφημα **G**, μπορούμε πάντα να ανακτήσουμε αποδοτικά μια ***ικανοποιούσα ανάθεση τιμών αληθείας*** στο **I.**

(2) Αν το γράφημα **G** δεν έχει ***ανεξάρτητο σύνολο*** **S** μεγέθους **k,** τότε ο λογικός (Boolean) τύπος **I** ***δεν είναι ικανοποιήσιμος***.

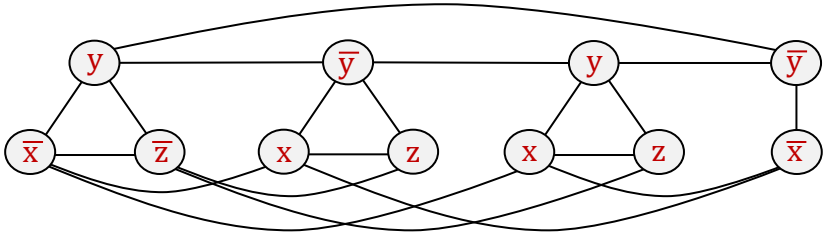


2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

$(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y})$

- **Απόδειξη (1)**

Για οποιαδήποτε μεταβλητή **x**, το **S δεν περιέχει κόμβους** με επιγραφές **x** και $\overline{\mathbf{x}}$!!!

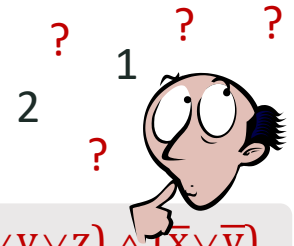


Αναθέτουμε: **x = true** εάν το **S** περιέχει κόμβο με επιγραφή **x**
x = false εάν το **S** περιέχει κόμβο με επιγραφή $\overline{\mathbf{x}}$

Επειδή, **|S| = k** πρέπει να έχει **ένα κόμβο ανά όρο** !!!

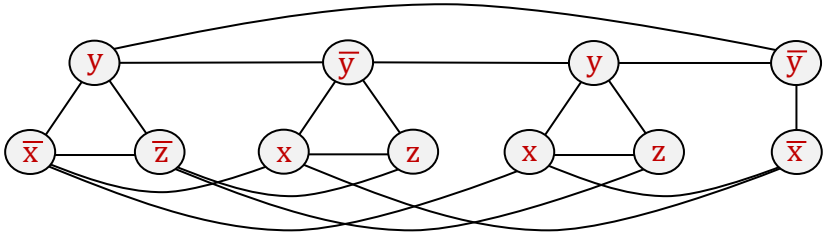
Αυτή η ανάθεση ικανοποιεί όλους τους όρους !!!

(1) Με δεδομένο ένα ***ανεξάρτητο σύνολο S*** με **k** κόμβους στο **G**, μπορούμε πάντα να ανακτήσουμε αποδοτικά μια ***ικανοποιούσα ανάθεση τιμών αληθείας*** στο **I.**



2 3SAT → ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

$(\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y})$

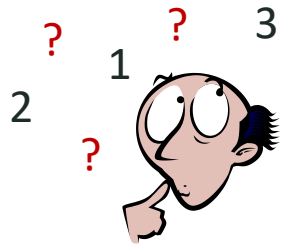


- **Απόδειξη (2)**

Αρκεί να αποδείξουμε ότι:
εάν ο λ.τ. **I** έχει μια *ικανοποιούσα ανάθεση τιμών αληθείας*,
τότε το **G** έχει ένα **IS** μεγέθους **k** !!!

Αυτό είναι εύκολο !!!... Για **κάθε όρο** του **3SAT**, ***επιλέγουμε ένα στοιχείο του οποίου η τιμή στην ικανοποιούσα ανάθεση είναι true*** και ***προσθέτουμε τον αντίστοιχο κόμβο στο S***... τότε το S είναι ανεξάρτητο... το βλέπετε ?

(2) Αν το γράφημα G δεν έχει ***ανεξάρτητο σύνολο S*** μεγέθους **k,** τότε ο λογικός (Boolean) τύπος **I** ***δεν είναι ικανοποιήσιμος***.



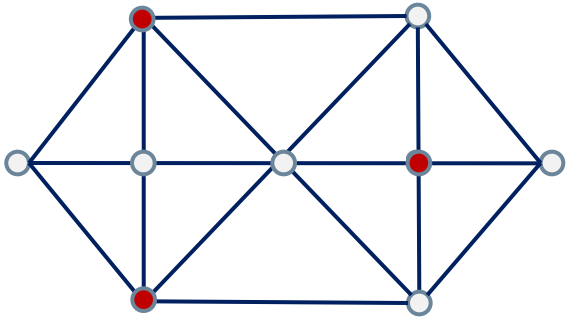
3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ → ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

- **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**

  Μας δίδεται γράφημα **n** κόμβων και *ακέραιος* **k**

  και μας ζητείται να βρούμε ένα σύνολο **IS** με **k** κόμβους οι οποίοι είναι ανεξάρτητοι

  ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!

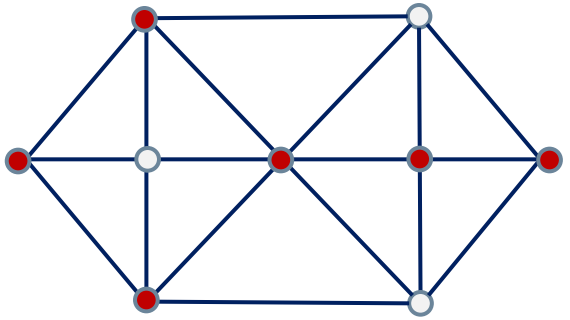


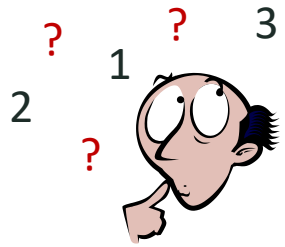
- **ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ**

  Μας δίδεται γράφημα **n** κόμβων και *ακέραιος* **k**

  και μας ζητείται να βρούμε ένα σύνολο **VC** με **k** κόμβους οι οποίοι καλύπτουν κάθε ακμή

  ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!



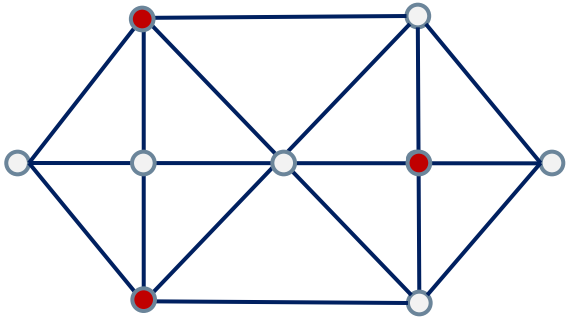


3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ → ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

- **Ισχύει ότι…**

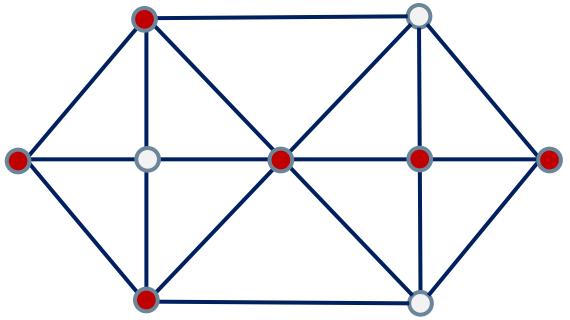
  Κάποιες αναγωγές βασίζονται στην **επινοητικότητα** του **συσχετισμού *δύο πολύ διαφορετικών προβλημάτων*** !!!

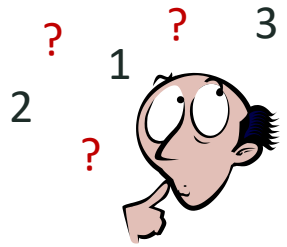
  Ενώ, άλλες απλώς **καταγράφουν** μια λεπτή ***μεταμφίεση*** του ***ενός προβλήματος*** στο ***άλλο*** !!!



- **Τι ισχύει εδώ ?**

  Τι μπορούμε να πούμε για τη **συσχέτιση** των δύο προβλημάτων **?**





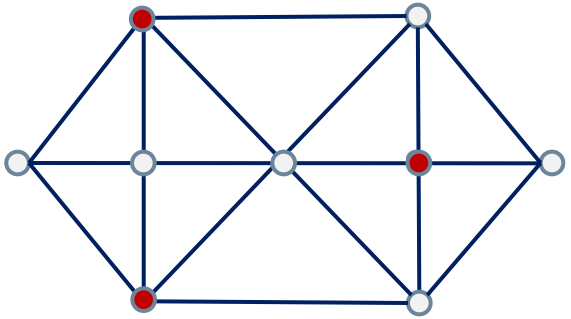
3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

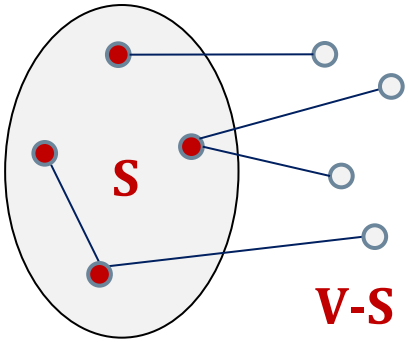
- **Παρατήρηση**

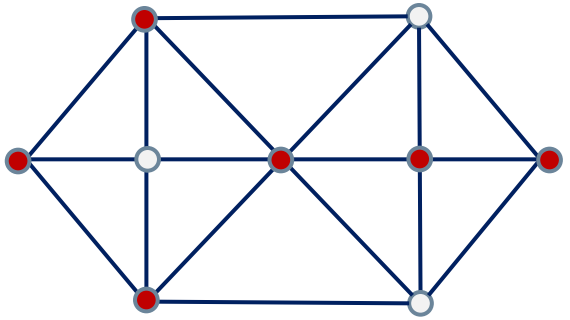
  Ένα σύνολο κόμβων **S** αποτελεί **κομβική κάλυψη** ενός γραφήματος **G = (V, E)**

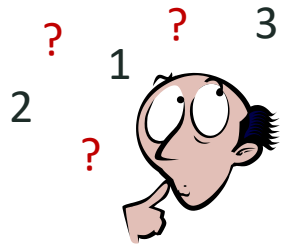
  εάν και μόνο εάν

  οι υπόλοιποι κόμβοι στο σύνολο **V-S** είναι **ανεξάρτητο σύνολο** του **G !!!**







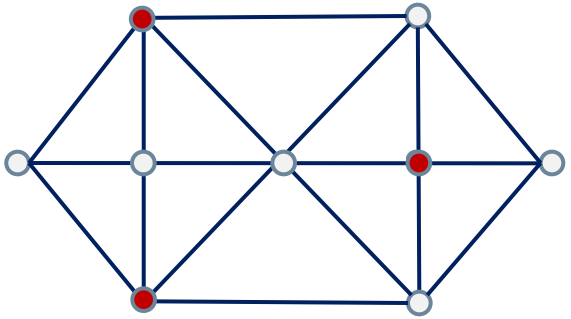


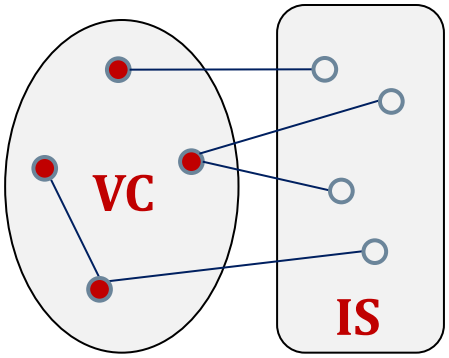
3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ → ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

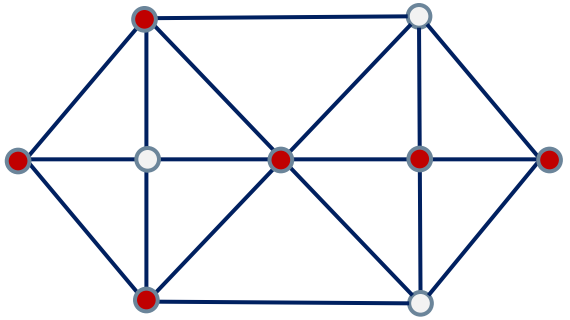
- **Αναγωγή**

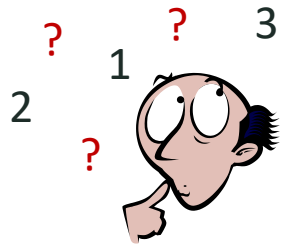
  Για να **λύσουμε** ένα στιγμιότυπο **(G, k)** του προβλήματος ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ

  απλώς **αναζητούμε** μια Κομβική Κάλυψη **VC** του **G** με **|V|-k** κόμβους !!!







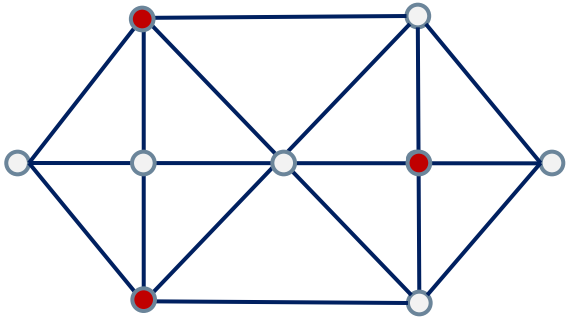


3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

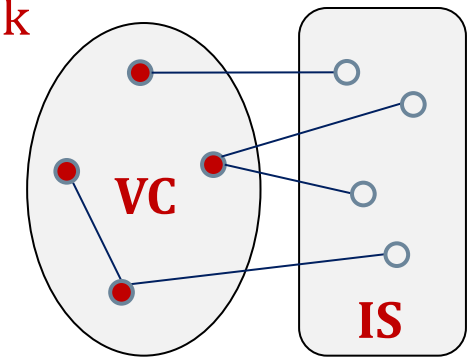
- **Αναγωγή**

  Εάν **υπάρχει** τέτοια **κομβική κάλυψη VC**

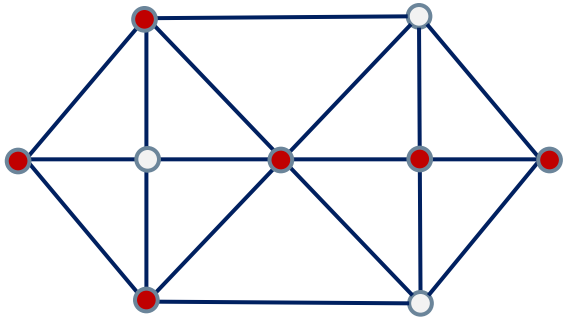
  τότε, *παίρνουμε όλους τους κόμβους* που **δεν ανήκουν** σε αυτή !!!

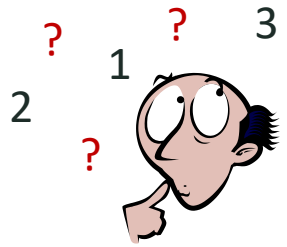


| VC | = |V|-k



| IS | = k



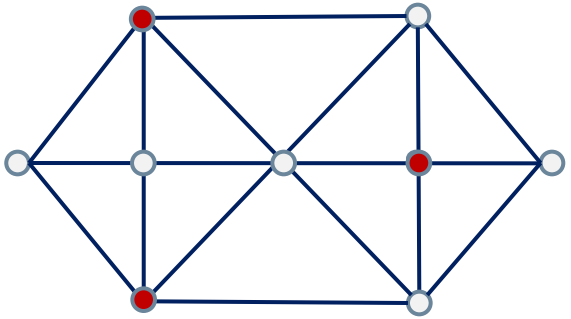


3 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΟΜΒΙΚΗ-ΚΑΛΥΨΗ

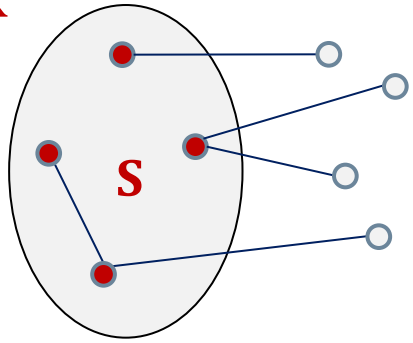
- **Αναγωγή**

  Εάν **ΔΕΝ υπάρχει** τέτοια **κομβική κάλυψη VC**

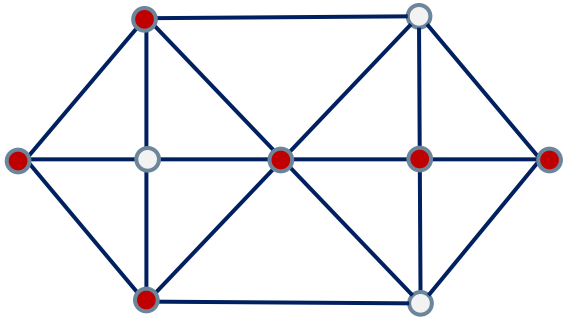
  τότε, το **G** δεν έχει **ανεξάρτητο σύνολο** μεγέθους **k** !!!

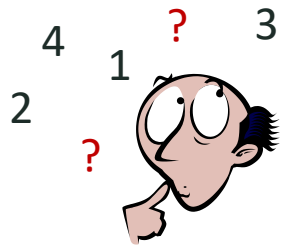


| VC | ≠ |V|-k



| IS | ≠ k





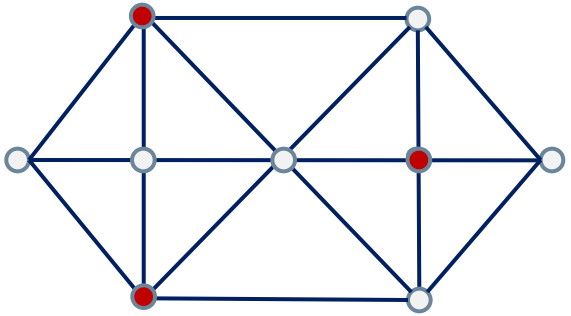
4 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ → ΚΛΙΚΑ

- **ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ**

  Μας δίδεται γράφημα **n** κόμβων και *ακέραιος* **k**

  και μας ζητείται να βρούμε ένα σύνολο **IS** με **k** κόμβους οι οποίοι είναι ανεξάρτητοι,

  ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!

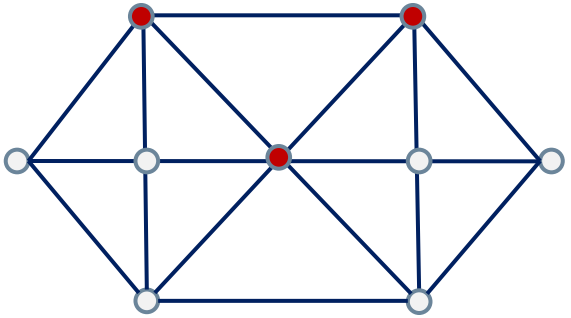


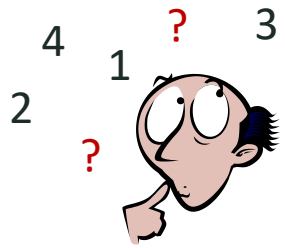
- **ΚΛΙΚΑ**

  Μας δίδεται γράφημα **n** κόμβων και *ακέραιος* **k**

  και μας ζητείται να βρούμε ένα σύνολο **CL** με **k** κόμβους οι οποίοι έχουν πλήρη σύνδεση

  ή να αναφέρουμε ότι δεν υπάρχει τέτοιο σύνολο κόμβων !!!





4 ΑΝΕΞΑΡΤΗΤΟ-ΣΥΝΟΛΟ ⟶ ΚΛΙΚΑ

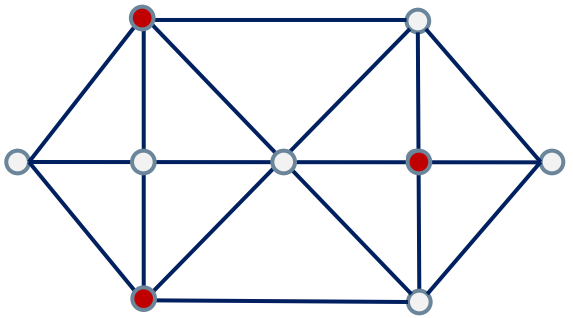
- **Παρατήρηση**

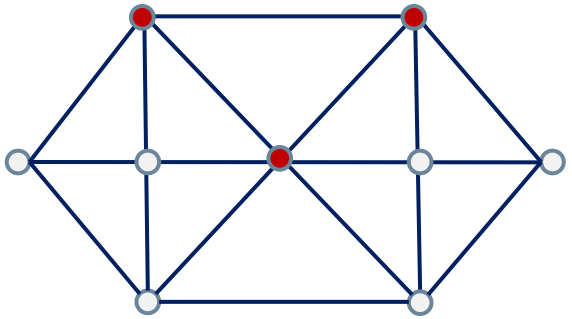
Πολύ εύκολη αναγωγή του ενός προβλήματος στο άλλο !!!

Ένα σύνολο κόμβων **S** αποτελεί **ανεξάρτητο σύνολο** ενός γραφήματος **G = (V, E)**

εάν και μόνο εάν

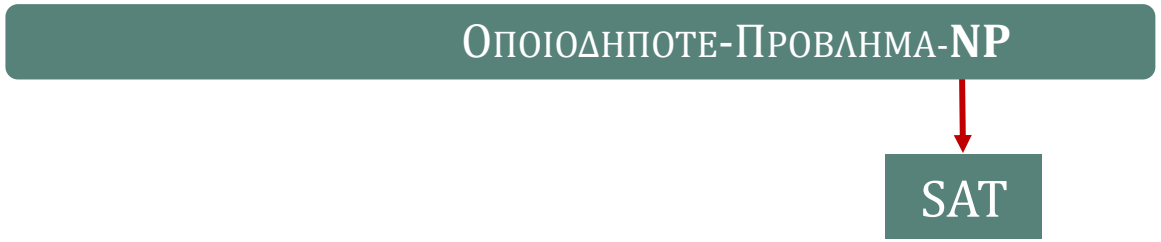
το **S** αποτελεί μια **κλίκα** του *συμπληρώματος (complement)* $\overline{\mathbf{G}} = (\mathbf{V}, \overline{\mathbf{E}})$ του γραφήματος **G !!!**





Αναγωγές Προβλημάτων NP

Οποιοδηποτε-Προβλημα-NP → SAT

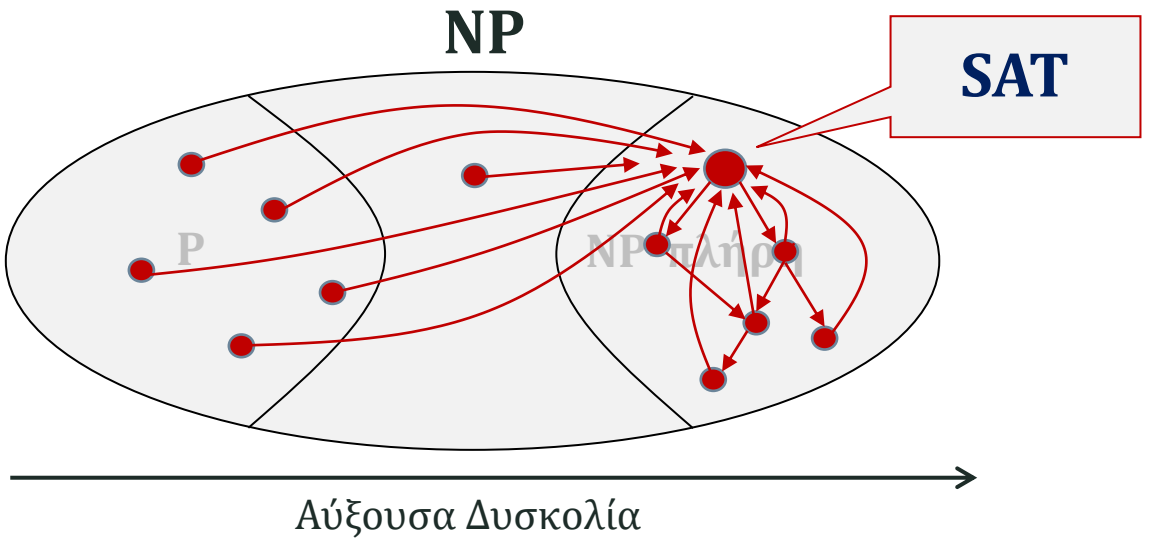


Οποιοδηποτε-Προβλημα-NP → SAT

- **Προβληματα NP**

  Όπως:

  - ✓ Ελάχιστες Διαδρομές
  - ✓ Ελάχιστα Συνδετικά Δένδρα
  - ✓ Αντιστοιχίσεις
  - ✓ Μέγιστες Ροές σε Δίκτυα
  - ✓ 3SAT
  - ✓ Ανεξαρτητο-Συνολο
  - ✓ Κομβικη-Καλυψη
  - ✓ Κλικα



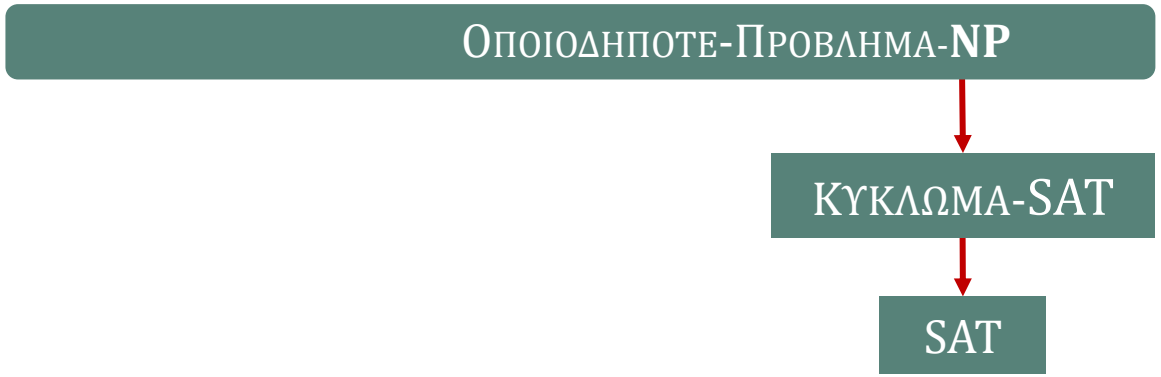
- **SAT**

$$(x \vee y \vee \bar{z} \vee w) \wedge (\bar{x} \vee y) \wedge (z \vee \bar{x} \vee w \vee \bar{p} \vee \bar{y}) \wedge (x \vee \bar{y} \vee z)$$

Αναγωγές Προβλημάτων NP

Οποιοδηποτε-Προβλημα-NP → SAT

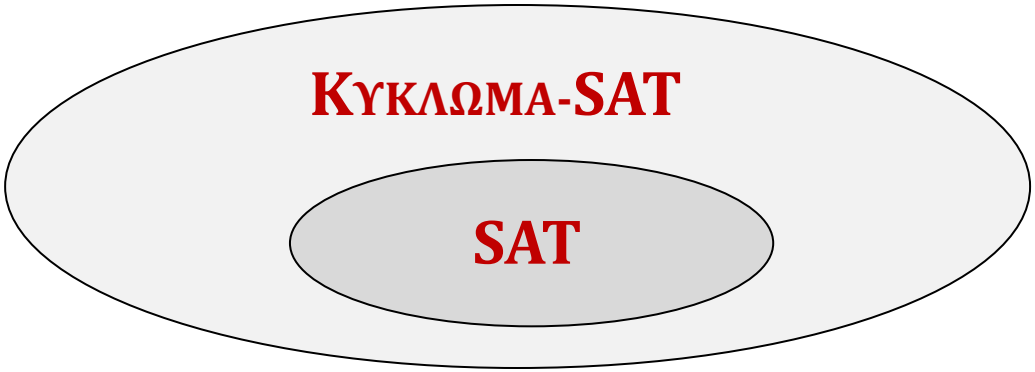
1. Οποιοδηποτε-Προβλημα-NP → Κυκλωμα-SAT
2. Κυκλωμα-SAT → SAT



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP → SAT

- **ΠΡΟΒΛΗΜΑ C-SAT**

  Θα αποδείξουμε ότι *όλα τα προβλήματα στο NP μπορούν να αναχθούν* σε μία γενίκευση του SAT που ονομάζουμε **ΚΥΚΛΩΜΑ-SAT** ή **C-SAT** (Circuit-SAT) !!!



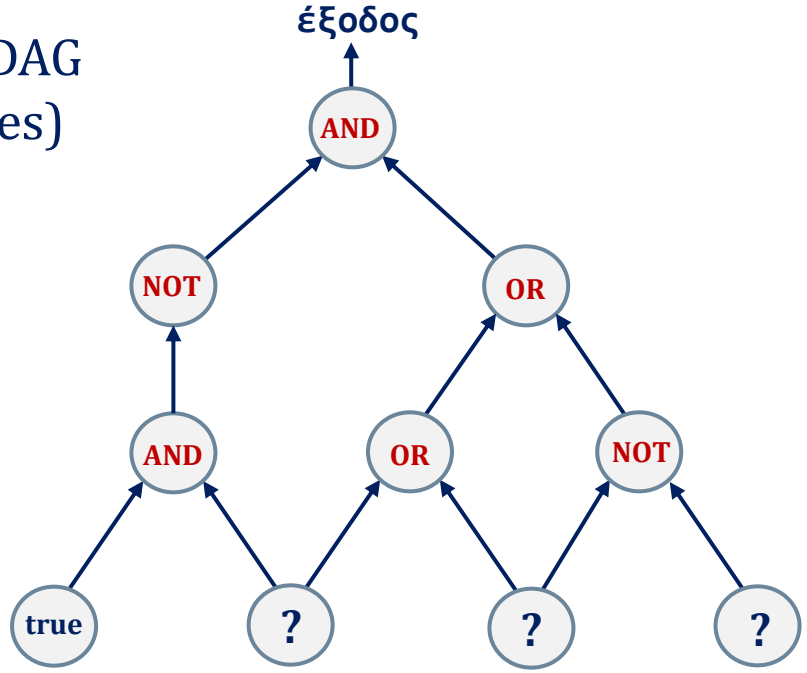
ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP → SAT

ΠΡΟΒΛΗΜΑ C-SAT

Ένα *λογικό (Boolean) κύκλωμα* -- ένα DAG του οποίου οι κόμβοι είναι **πύλες** (gates) 5 διαφορετικών τύπων :

- ✓ Πύλες **AND** και **OR** με βαθμό εισόδου 2
- ✓ Πύλες **NOT** με βαθμό εισόδου 1
- ✓ Πύλες ***γνωστής εισόδου*** με βαθμό 0
- ✓ Πύλες ***άγνωστης εισόδου*** με βαθμό 0

Ένας τερματικός κόμβος του DAG είναι η **πύλη εξόδου** (output gate)



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

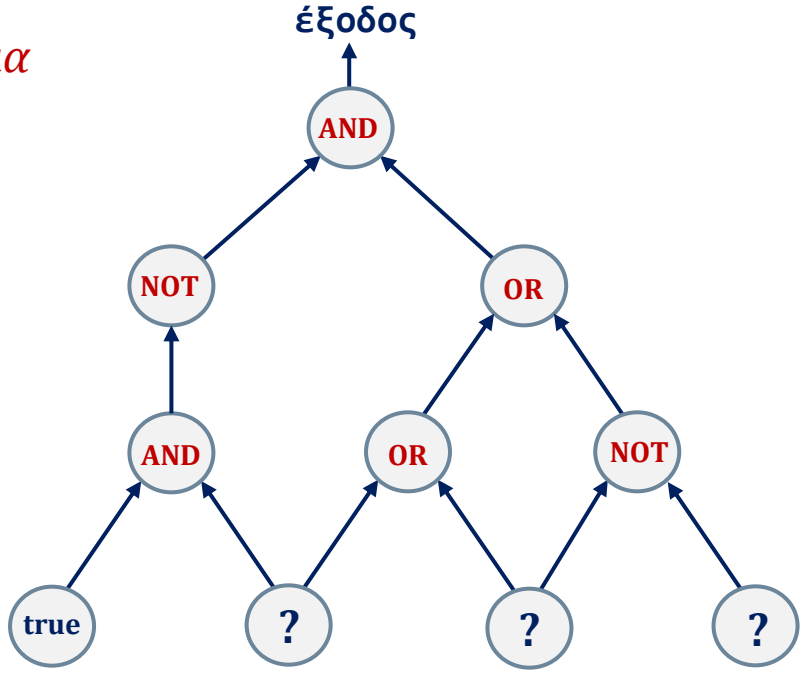
- **ΠΡΟΒΛΗΜΑ C-SAT**

  Με δεδομένο ένα *λογικό (Boolean) κύκλωμα*

  βρείτε μια ***ανάθεση τιμών αληθείας*** για τις άγνωστες εισόδους :

  η **πύλη εξόδου** να αποτιμάται **true**,

  ή να αναφέρουμε ότι δεν υπάρχει καμία τέτοια ανάθεση τιμών !!!



! Οποιοδηποτε-Προβλημα-NP → SAT

- **Πρόβλημα C-SAT**

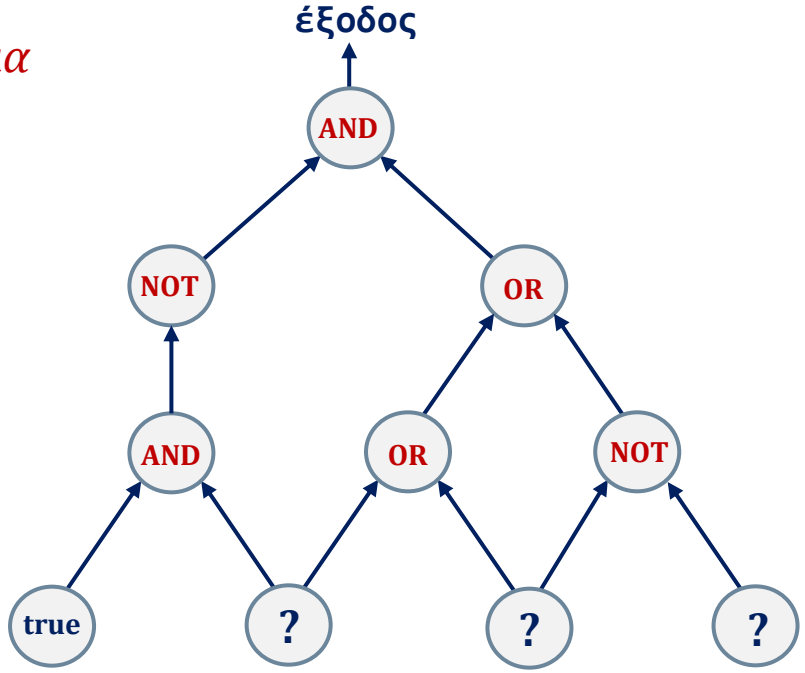
  Με δεδομένο ένα *λογικό (Boolean) κύκλωμα*

  βρείτε μια ***ανάθεση τιμών αληθείας*** για τις άγνωστες εισόδους :

  η **πύλη εξόδου** να αποτιμάται **true**,

  ή να αναφέρουμε ότι δεν υπάρχει καμία τέτοια ανάθεση τιμών !!!

  `(false, true, true)` ⇒ `true`
  `(true, false, true)` ⇒ `false`



! Οποιοδηποτε-Προβλημα-NP → SAT

- **Πρόβλημα C-SAT**

  Με δεδομένο ένα *λογικό (Boolean) κύκλωμα*

  βρείτε μια ***ανάθεση τιμών αληθείας*** για τις άγνωστες εισόδους :

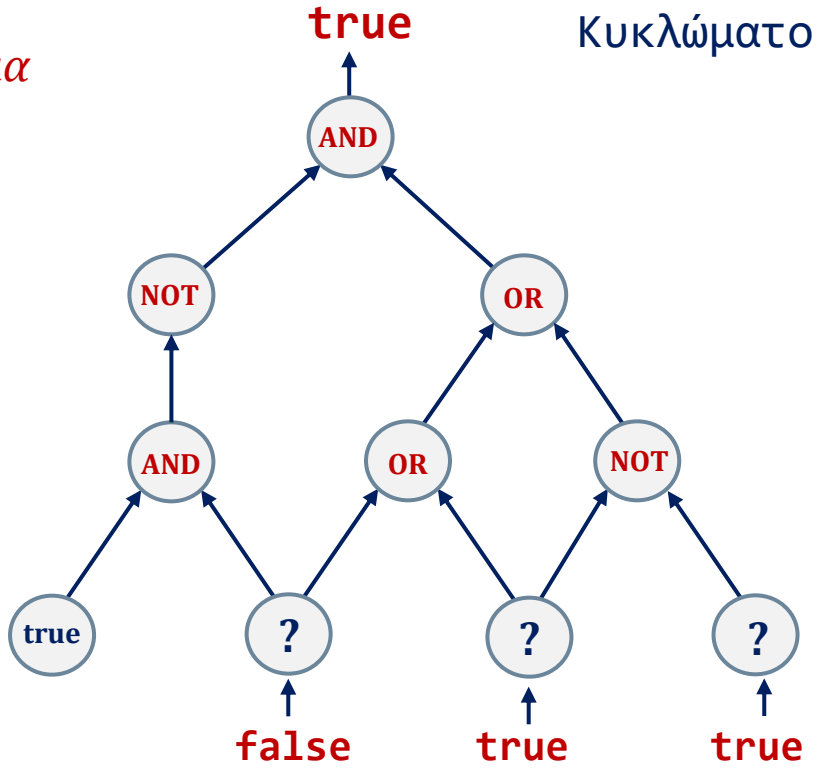
  η **πύλη εξόδου** να αποτιμάται **true**,

  ή να αναφέρουμε ότι δεν υπάρχει καμία τέτοια ανάθεση τιμών !!!

  `(false, true, true)` ⇒ `true`

  `(true, false, true)` ⇒ `false`

Αποτίμηση Κυκλώματος



! Οποιοδηποτε-Προβλημα-NP → SAT

- **Πρόβλημα C-SAT**

  Με δεδομένο ένα *λογικό (Boolean) κύκλωμα*

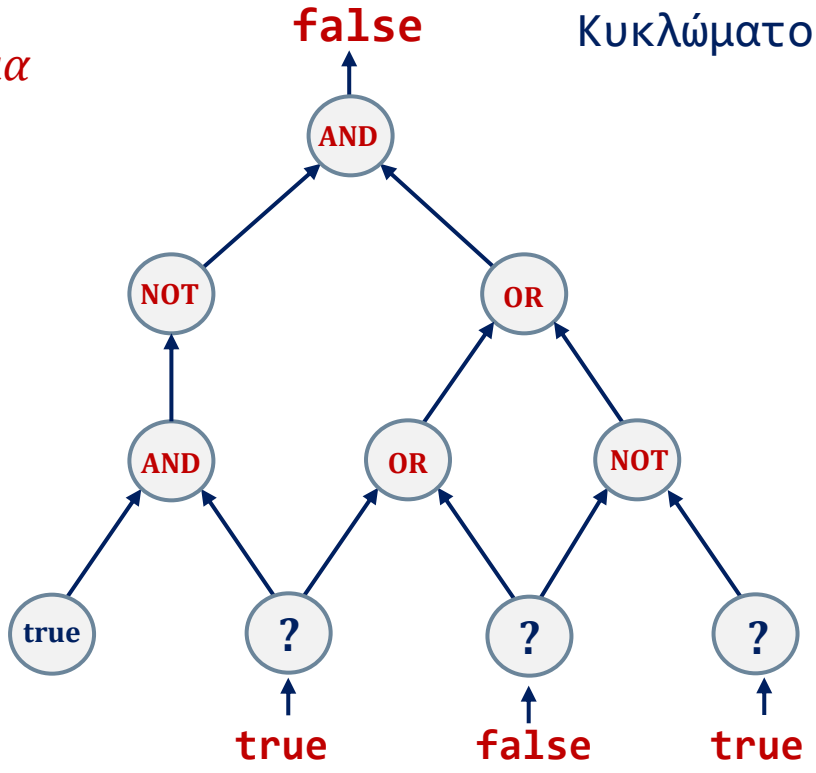
  βρείτε μια ***ανάθεση τιμών αληθείας*** για τις άγνωστες εισόδους :

  η **πύλη εξόδου** να αποτιμάται **true**,

  ή να αναφέρουμε ότι δεν υπάρχει καμία τέτοια ανάθεση τιμών !!!

  (false, true, true) ⇒ true
  (true, false, true) ⇒ false

Αποτίμηση Κυκλώματος



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

- **ΠΡΟΒΛΗΜΑ C-SAT**

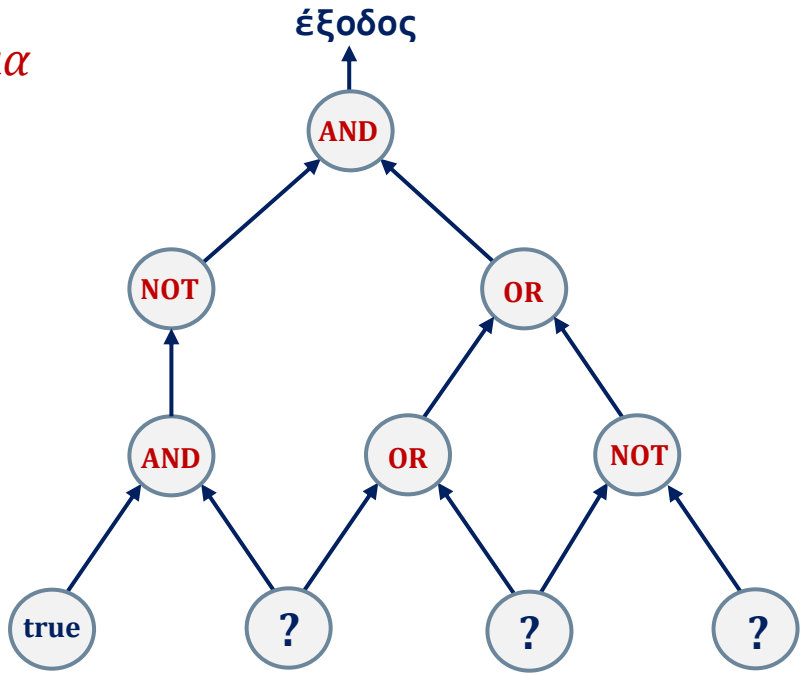
  Με δεδομένο ένα *λογικό (Boolean) κύκλωμα*

  βρείτε μια ***ανάθεση τιμών αληθείας*** για τις άγνωστες εισόδους :

  η **πύλη εξόδου** να αποτιμάται **true**,

  ή να αναφέρουμε ότι δεν υπάρχει καμία τέτοια ανάθεση τιμών !!!

C-SAT ως Πρόβλημα Αναζήτησης



Αναγωγές Προβλημάτων NP

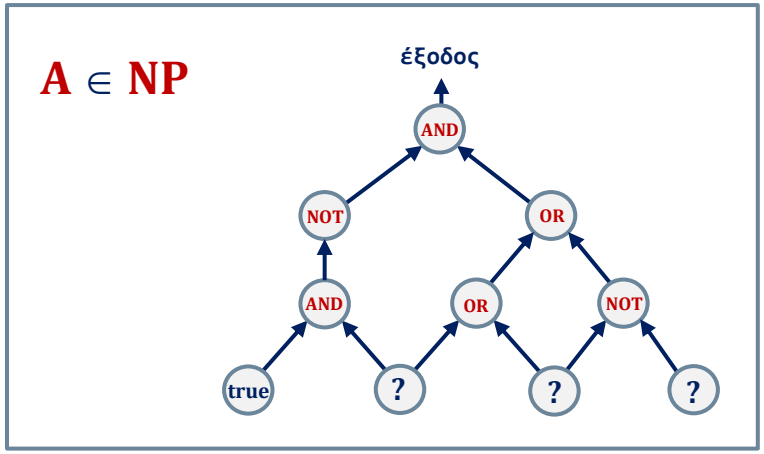
! Οποιοδηποτε-Προβλημα-NP ⟶ SAT

1 A ⟶ Κυκλωμα-SAT

Έστω **A** ένα πρόβλημα στο **NP** !!!

Θέλουμε μια αναγωγή

A ⟶ Κυκλωμα-SAT



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

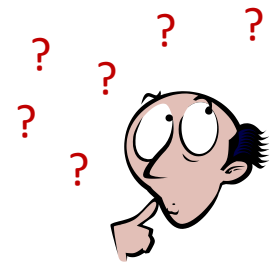
1 A ⟶ ΚΥΚΛΩΜΑ-SAT

Έστω **A** ένα πρόβλημα στο **NP** !!!

Θέλουμε μια αναγωγή

A ⟶ ΚΥΚΛΩΜΑ-SAT

A ∈ NP



Αυτό φαίνεται δύσκολο !!!...

καθώς *δεν γνωρίζουμε σχεδόν τίποτε* για το πρόβλημα A !!!

Αναγωγές Προβλημάτων NP

! Οποιοδηποτε-Προβλημα-NP ⟶ SAT

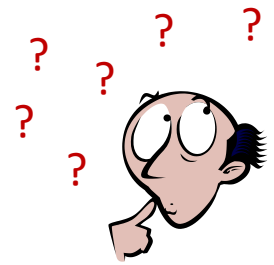
① A ⟶ Κυκλωμα-SAT

Έστω **A** ένα πρόβλημα στο **NP** !!!

Το **ΜΟΝΟ** που γνωρίζουμε για το **A**

Πρόβλημα Αναζήτησης

A ∈ **NP**



Χαρακτηριστικό: Οποιαδήποτε **προτεινόμενη λύση** μπορεί να **ελεγχθεί γρήγορα** για την ορθότητά της !!!

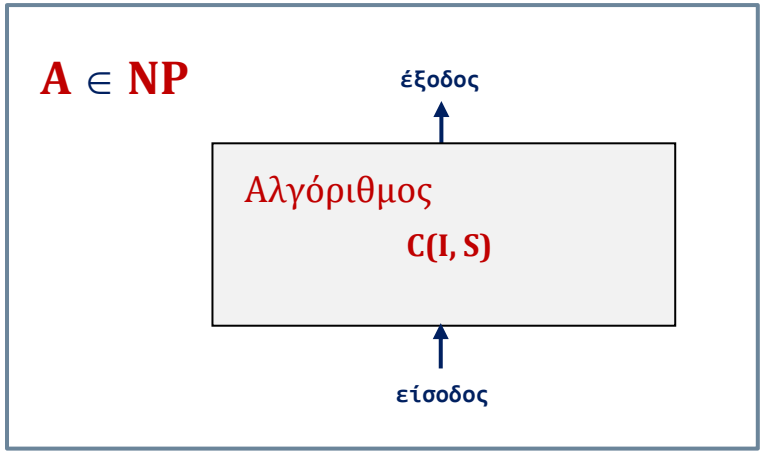
! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

① A ⟶ ΚΥΚΛΩΜΑ-SAT

Έστω **A** ένα πρόβλημα στο **NP** !!!

Το **ΜΟΝΟ** που γνωρίζουμε για το **A**

Πρόβλημα Αναζήτησης



Χαρακτηριστικό: Πολυωνυμικός αλγόριθμος ελέγχου **C(I, S)** !!!

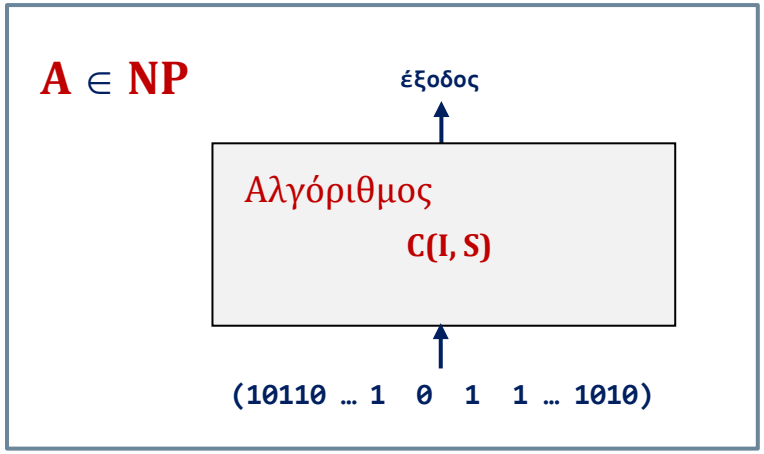
! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

① A ⟶ ΚΥΚΛΩΜΑ-SAT

Έστω **A** ένα πρόβλημα στο **NP** !!!

Το **ΜΟΝΟ** που γνωρίζουμε για το **A**

Πρόβλημα Αναζήτησης



Χαρακτηριστικό: Πολυωνυμικός αλγόριθμος ελέγχου **C(I, S)** !!!

✓ Υποθέτουμε : `I = (1001110010101001)`

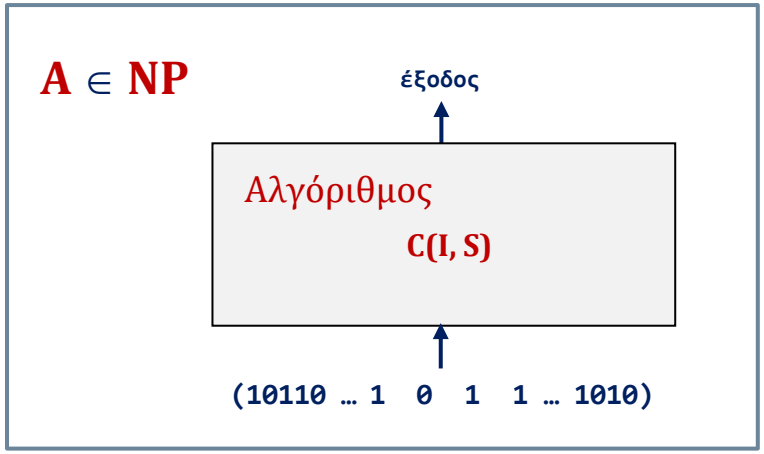
`S = (1101010011101101)` και $|S|=O(|I|)$

! Οποιοδηποτε-Προβλημα-NP ⟶ SAT

① A ⟶ Κυκλωμα-SAT

Γνωρίζουμε ότι:

Οποιοσδήποτε πολυωνυμικός αλγόριθμος μπορεί να *μετατραπεί* σε κύκλωμα !!!



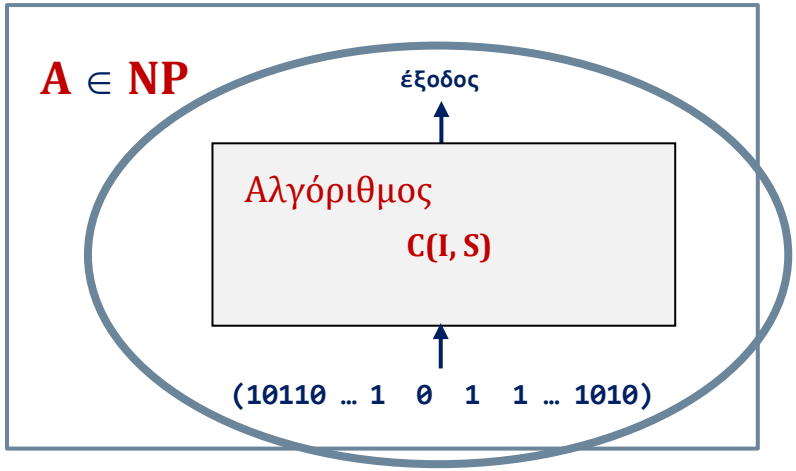
Οι **πύλες εισόδου του κυκλώματος** κωδικοποιούν την είσοδο του αλγόριθμου `I = (1001110010101101 10)`

! Οποιοδηποτε-Προβλημα-NP ⟶ SAT

① A ⟶ Κυκλωμα-SAT

Γνωρίζουμε ότι:

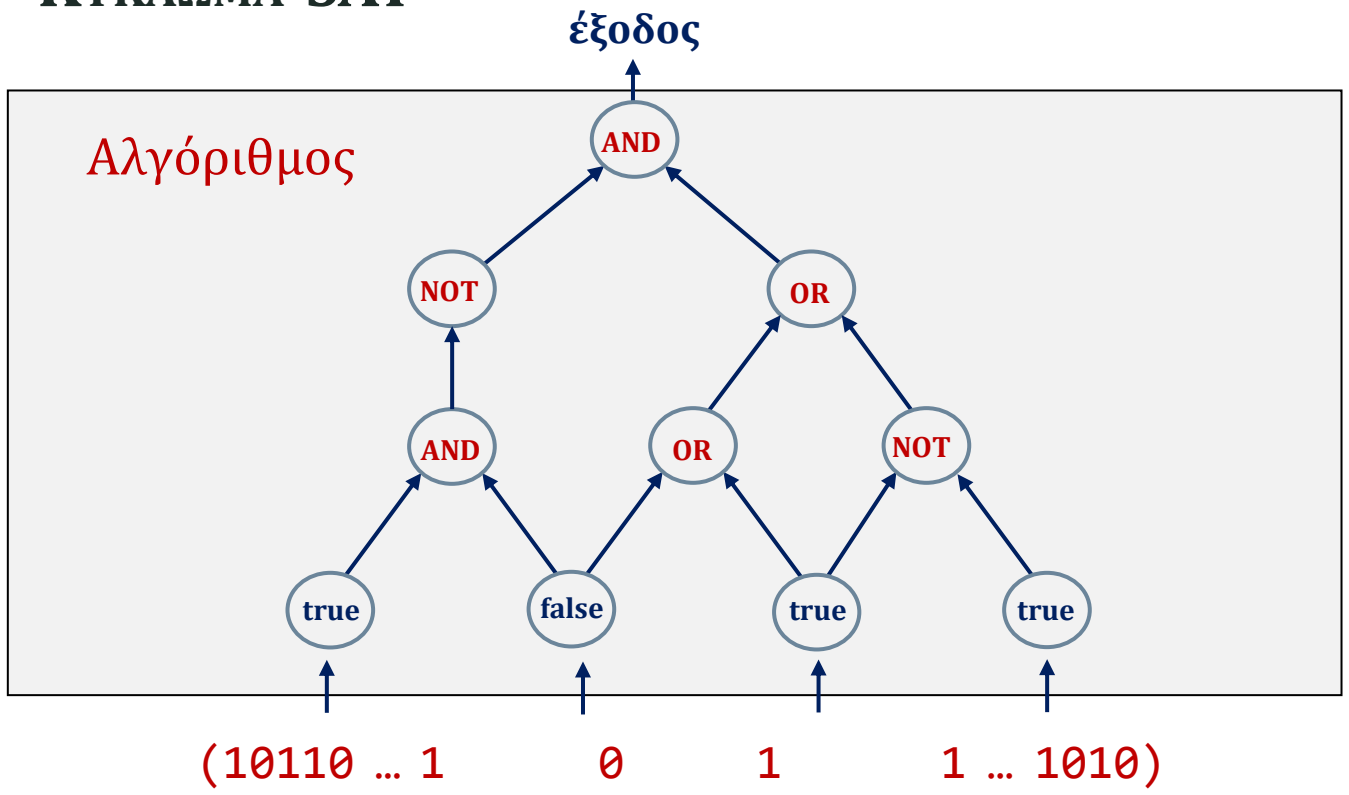
Οποιοσδήποτε πολυωνυμικός αλγόριθμος μπορεί να *μετατραπεί* σε κύκλωμα !!!



Οι **πύλες εισόδου του κυκλώματος** κωδικοποιούν την είσοδο του αλγόριθμου `I = (1001110010101101l0)`

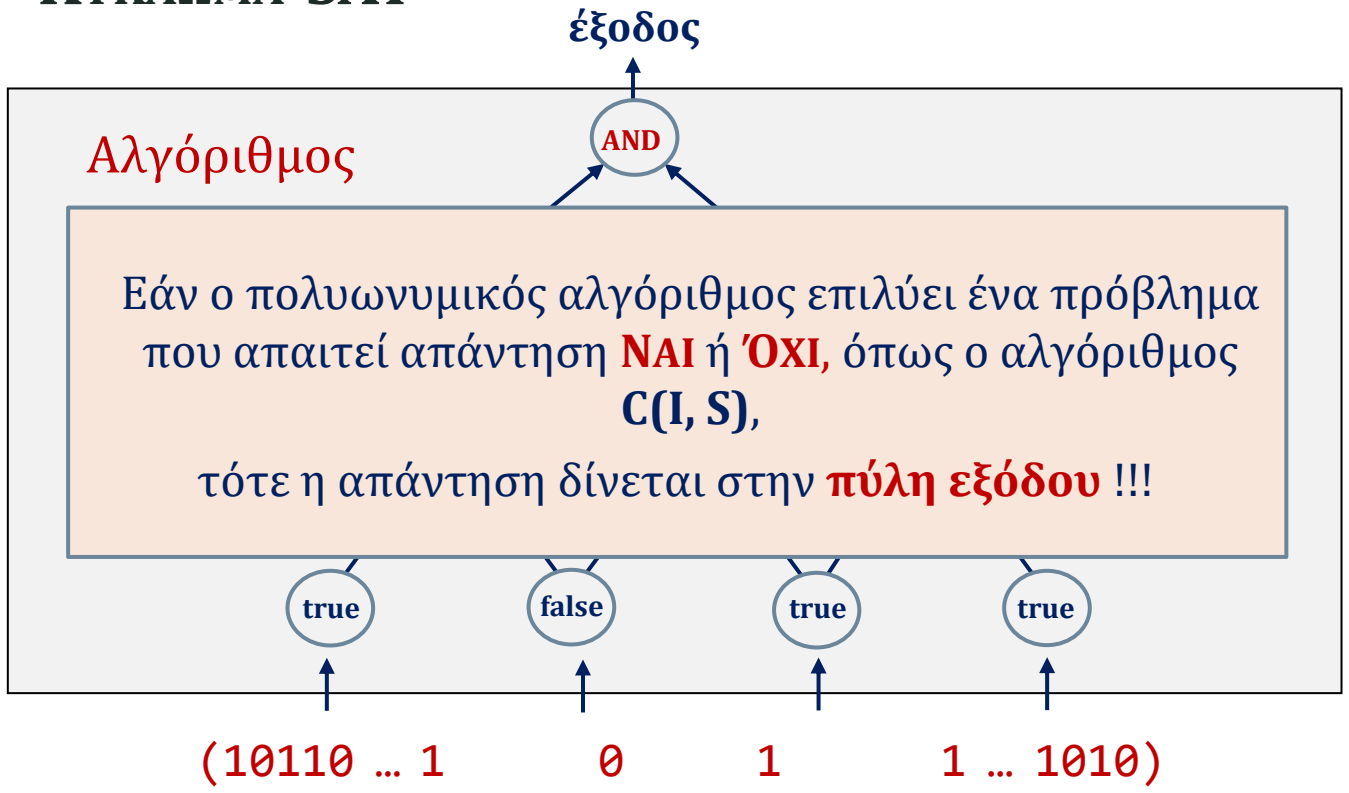
! Οποιοδηποτε-Προβλημα-NP → SAT

1 A → Κυκλωμα-SAT



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP → SAT

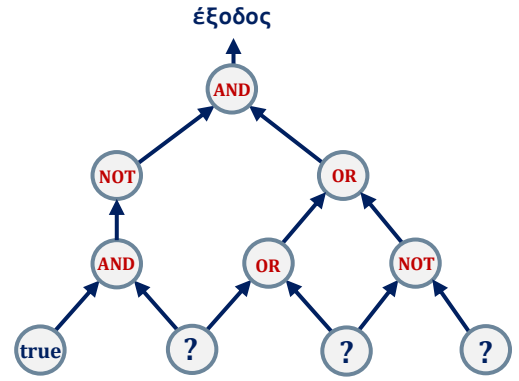
1 Α → ΚΥΚΛΩΜΑ-SAT



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

1 A ⟶ ΚΥΚΛΩΜΑ-SAT

A ∈ NP



Τι συμπεράνουμε ?

Για οποιοδήποτε στιγμιότυπο **I** του προβλήματος αναζήτησης **A**

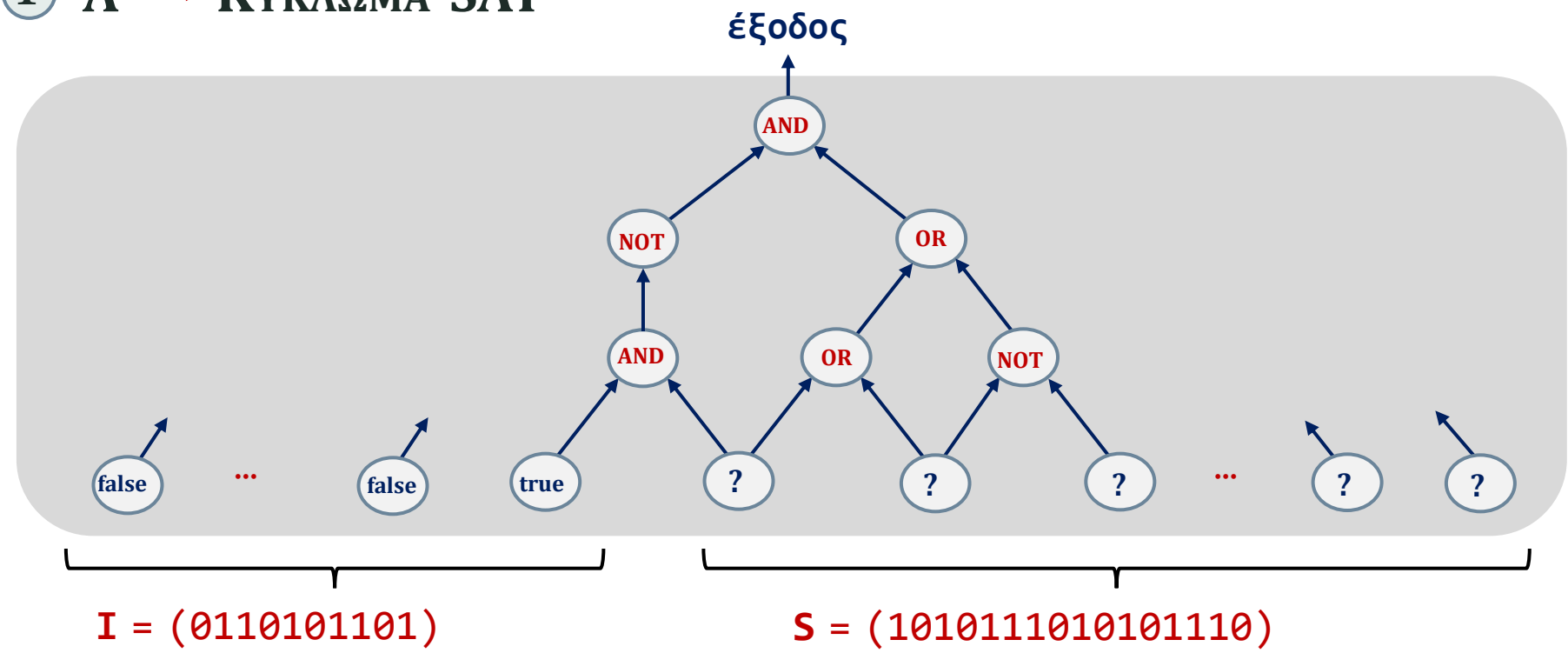
Μπορούμε να **κατασκευάσουμε** σε *πολυωνυμικό χρόνο* ένα **Κύκλωμα-SAT**

του οποίου
οι ***γνωστές είσοδοι*** είναι τα bit του **I** και οι ***άγνωστες είσοδοι*** είναι τα bit του **S** :

Η έξοδός του να είναι **true εάν-ν** οι *άγνωστες είσοδοι* αποτελούν μια λύση **S** του **I** !!!

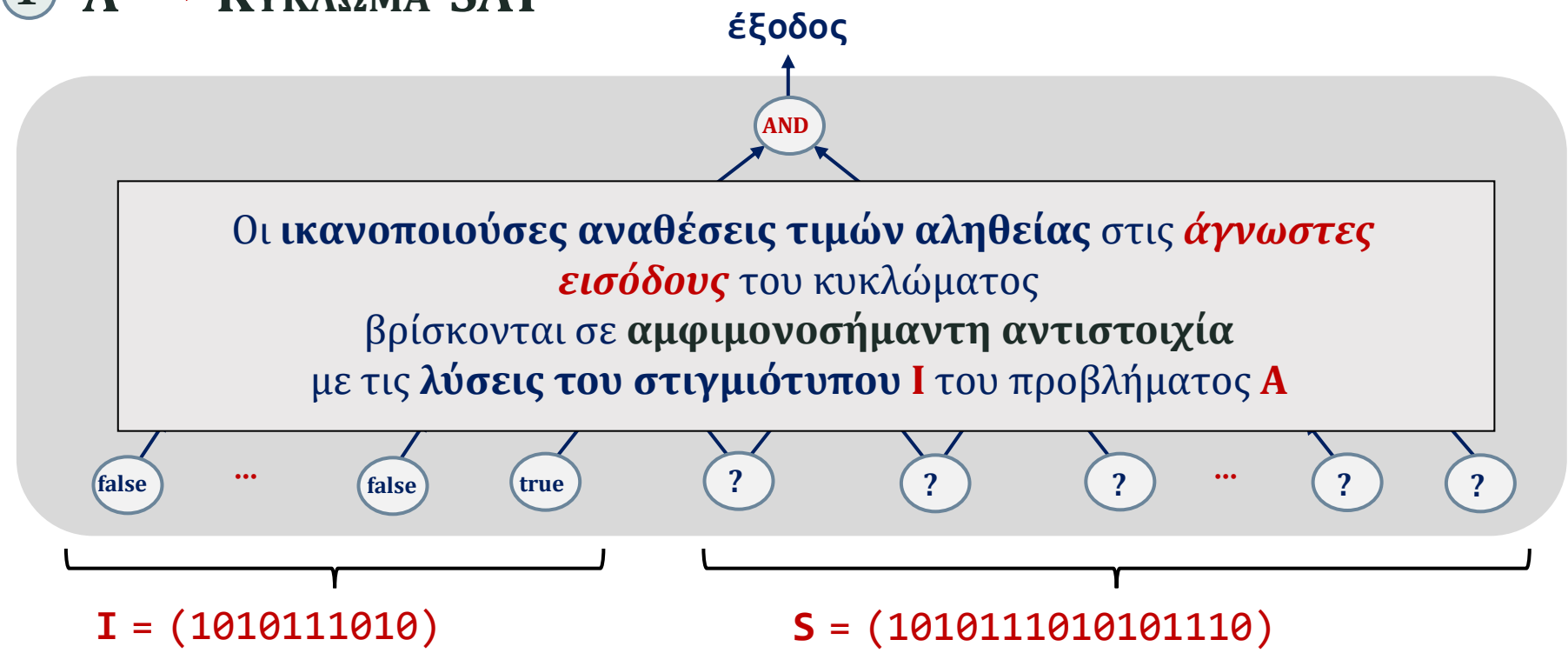
! Οποιοδηποτε-Προβλημα-NP → SAT

1 A → Κυκλωμα-SAT



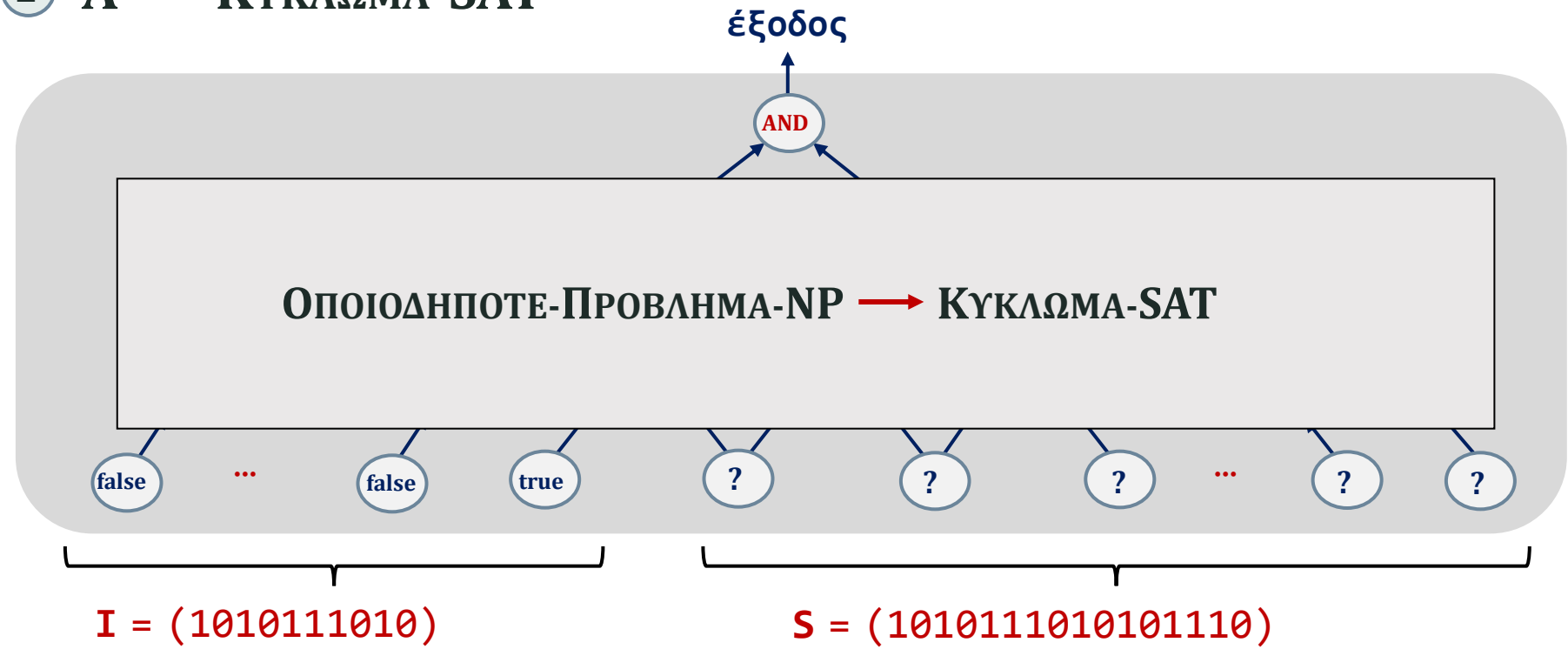
! Οποιοδηποτε-Προβλημα-NP → SAT

1 A → Κυκλωμα-SAT



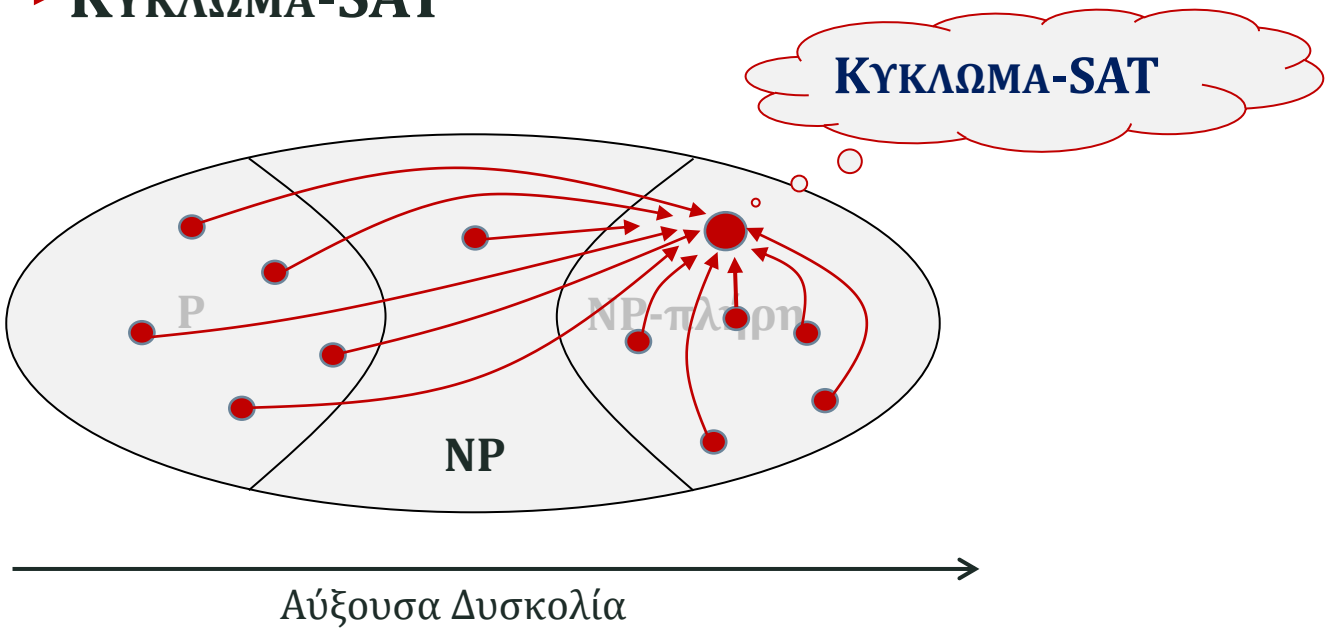
! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP → SAT

1 A → ΚΥΚΛΩΜΑ-SAT



! Οποιοδηποτε-Προβλημα-NP → SAT

1 A → Κυκλωμα-SAT

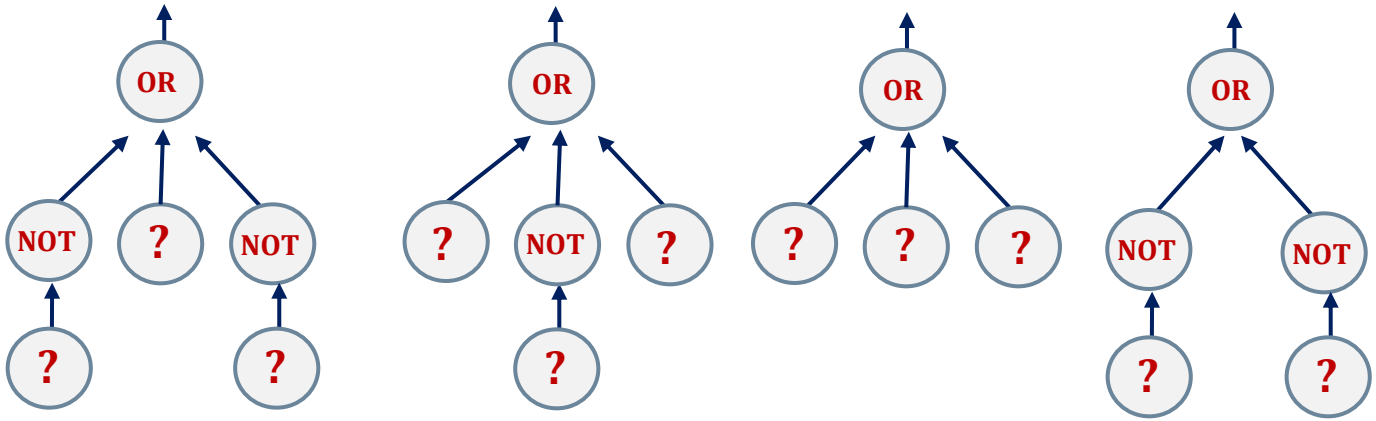


! Οποιοδηποτε-Προβλημα-NP → SAT

- **Το Κυκλωμα-SAT είναι γενίκευση του SAT**

  Το SAT **αναζητά** μια *ικανοποιούσα ανάθεση τιμών αληθείας* για ***ένα λογικό κύκλωμα*** που έχει την εξής ***απλή δομή*** !!!

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$



! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

② ΚΥΚΛΩΜΑ-SAT ⟶ SAT

Μπορούμε να ξαναγράψουμε οποιοδήποτε λογικό κύκλωμα σε *κανονική συζευκτική μορφή* (CNF) :

Για ***κάθε πύλη g*** του λογικού κυκλώματος **δημιουργούμε** μια

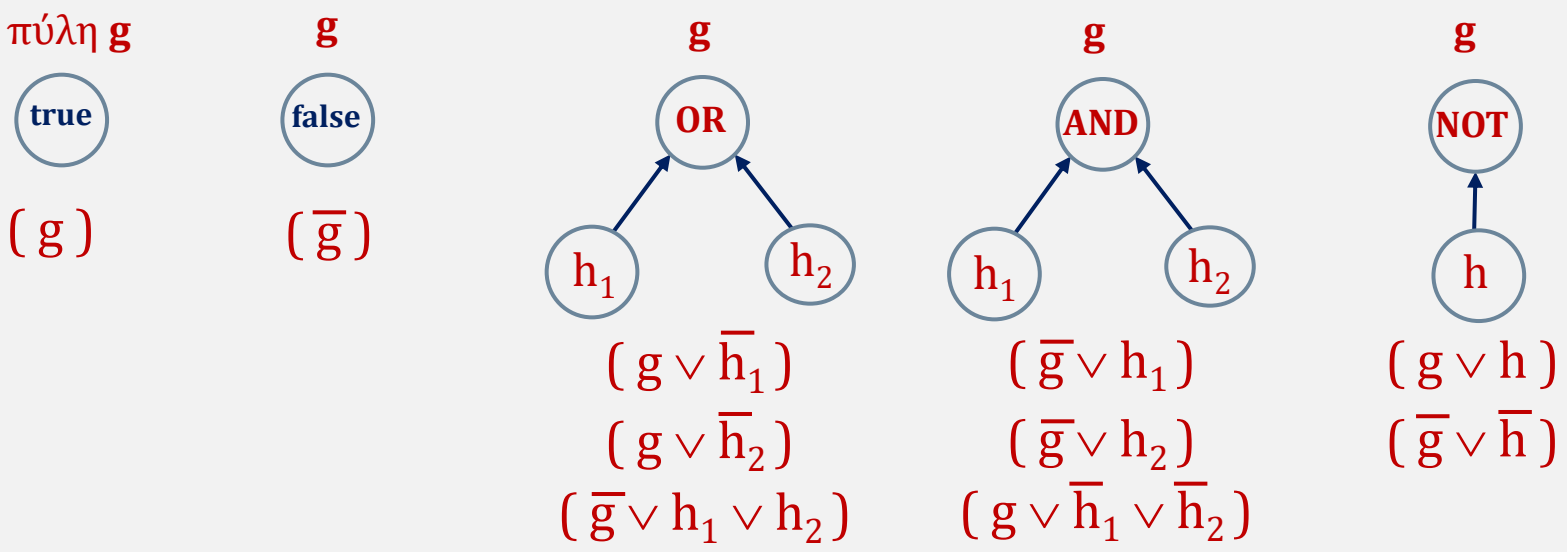
μεταβλητή g

και μοντελοποιούμε την **επίδραση της πύλης** χρησιμοποιώντας μερικούς **όρους** !!!

Να πως !!!...

! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP → SAT

2 ΚΥΚΛΩΜΑ-SAT → SAT



Βλέπετε γιατί αυτοί οι όροι επιβάλουν ακριβώς το επιθυμητό αποτέλεσμα !!!!

! ΟΠΟΙΟΔΗΠΟΤΕ-ΠΡΟΒΛΗΜΑ-NP ⟶ SAT

② ΚΥΚΛΩΜΑ-SAT ⟶ SAT

Για να ολοκληρώσουμε !!!

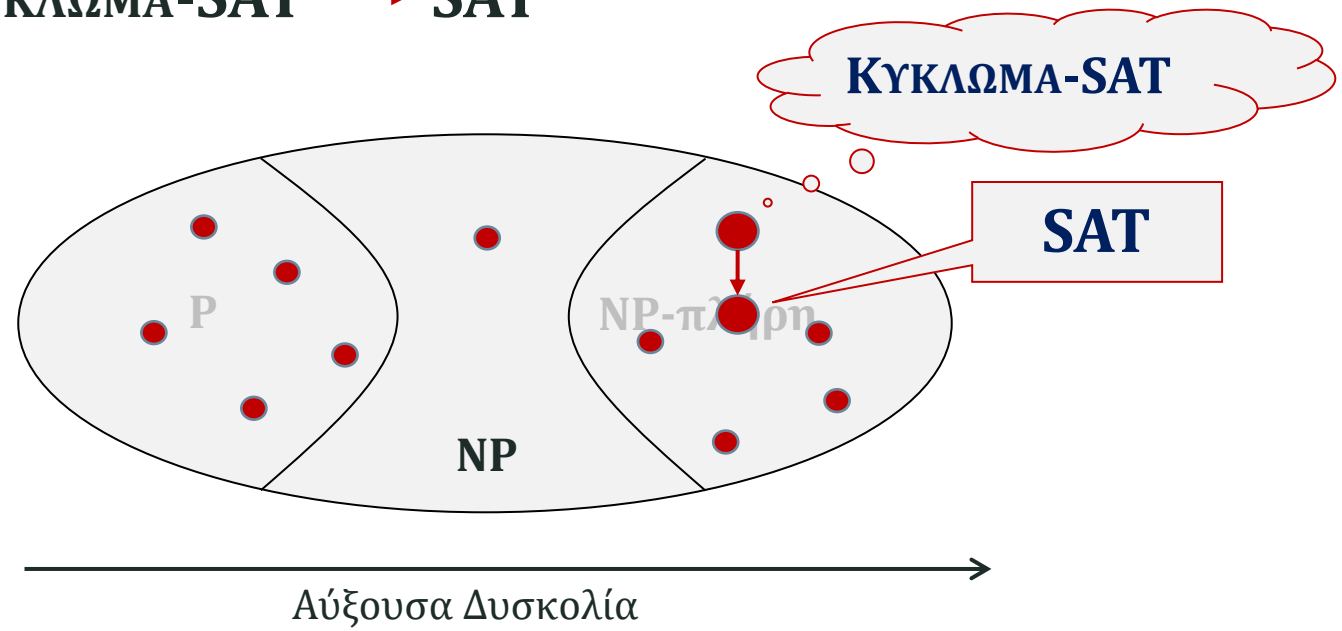
Εάν **g** είναι η **πύλη εξόδου**, την **αναγκάζουμε** να είναι **true** με την προσθήκη του όρου **(g)** στο λογικό τύπο του SAT !!!

{ το **στιγμιότυπο** του **SAT** που προκύπτει } ⟺ { με το δεδομένο **στιγμιότυπο** του **SAT-ΚΥΚΛΩΜΑΤΟΣ** }

Πράγματι: οι ***ικανοποιούσες αναθέσεις τιμών αληθείας*** της προκύπτουσας κανονικής συζευκτικής μορφής **(.∨.∨.) ∧ (.∨.∨.) ∧ ... ∧ (.∨.) ∧ (g)** βρίσκονται σε **αμφιμονοσήμαντη αντιστοιχία** με ***εκείνες του κυκλώματος*** !!!

! Οποιοδηποτε-Προβλημα-NP → SAT

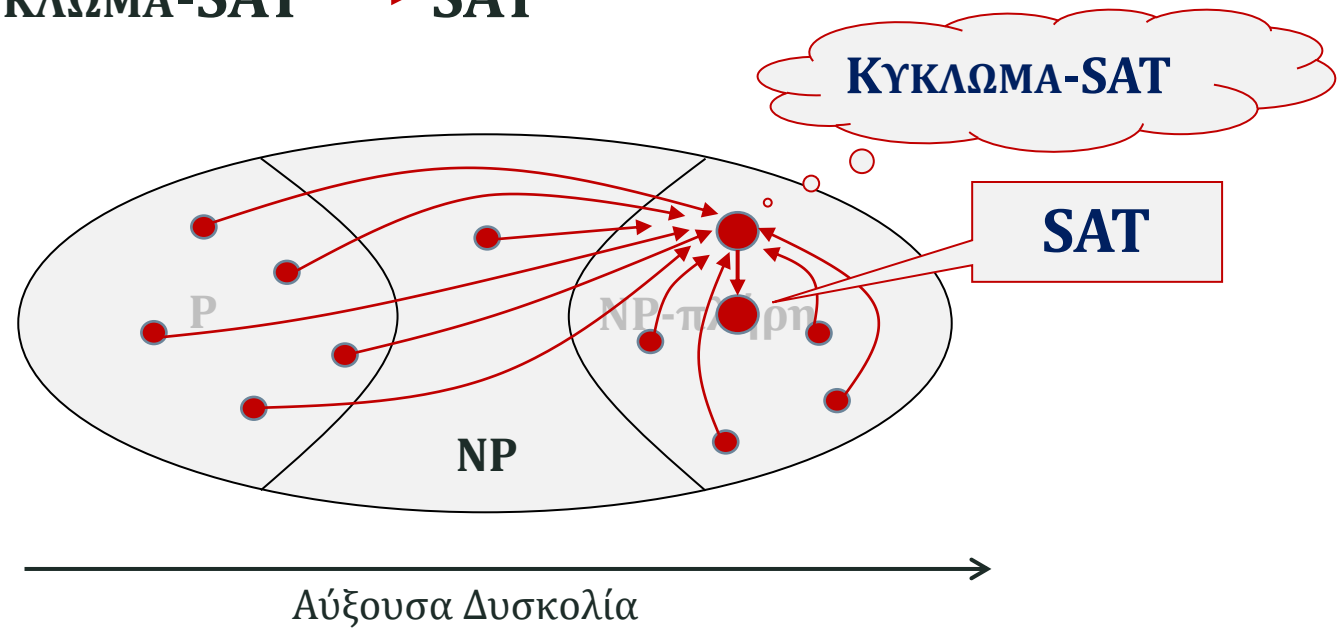
2 Κυκλωμα-SAT → SAT



Αναγωγές Προβλημάτων NP

! Οποιοδηποτε-Προβλημα-NP → SAT

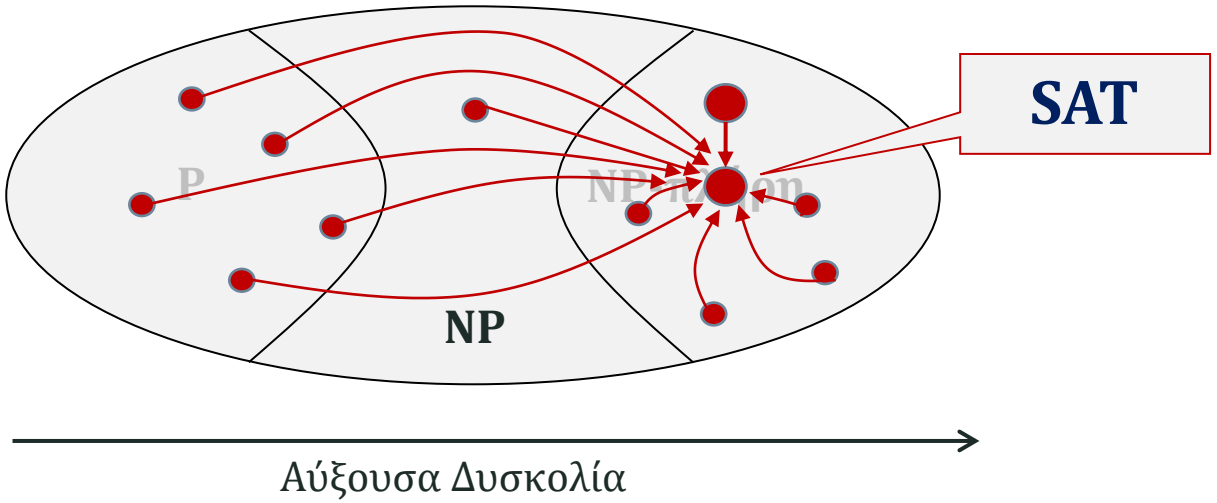
② Κυκλωμα-SAT → SAT



① Οποιοδηποτε-Προβλημα-NP → Κυκλωμα-SAT

! Οποιοδηποτε-Προβλημα-NP → SAT

- Από (1) και (2)

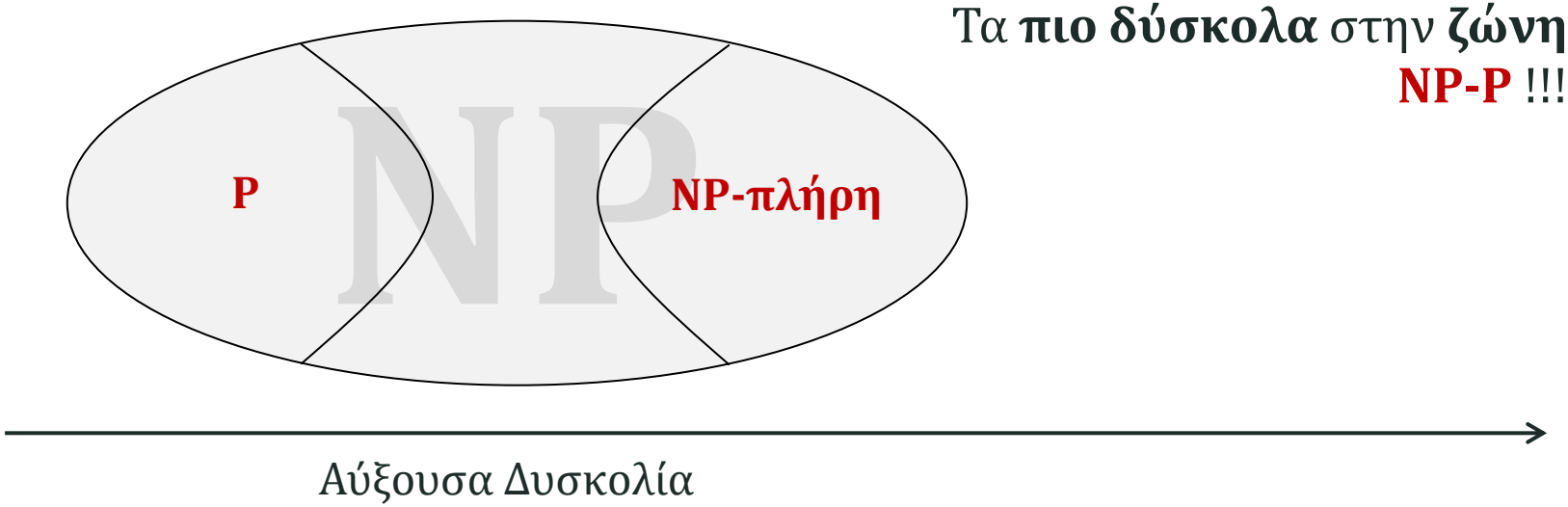


Προβλήματα NP-πλήρη !!!

- Εάν **P ≠ NP**

NP-πλήρη προβλήματα:

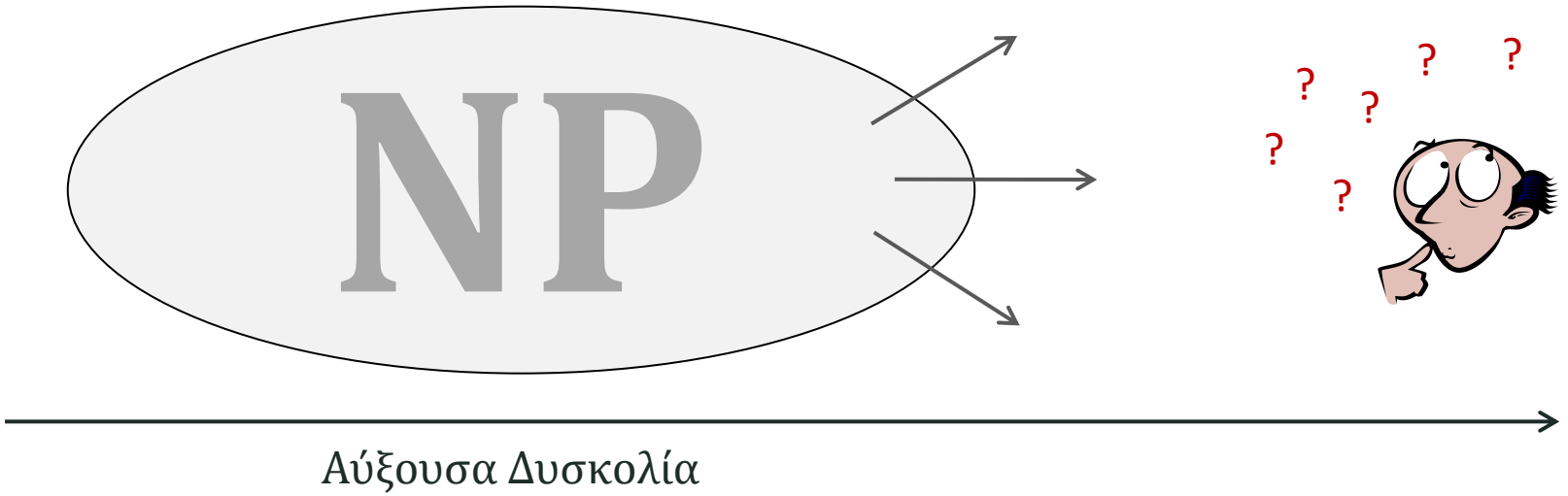
Τα **πιο δύσκολα** στην **ζώνη NP-P** !!!



Μια Σχηματική Παρουσίαση !!!

Εύλογο **Ερώτημα** !!!

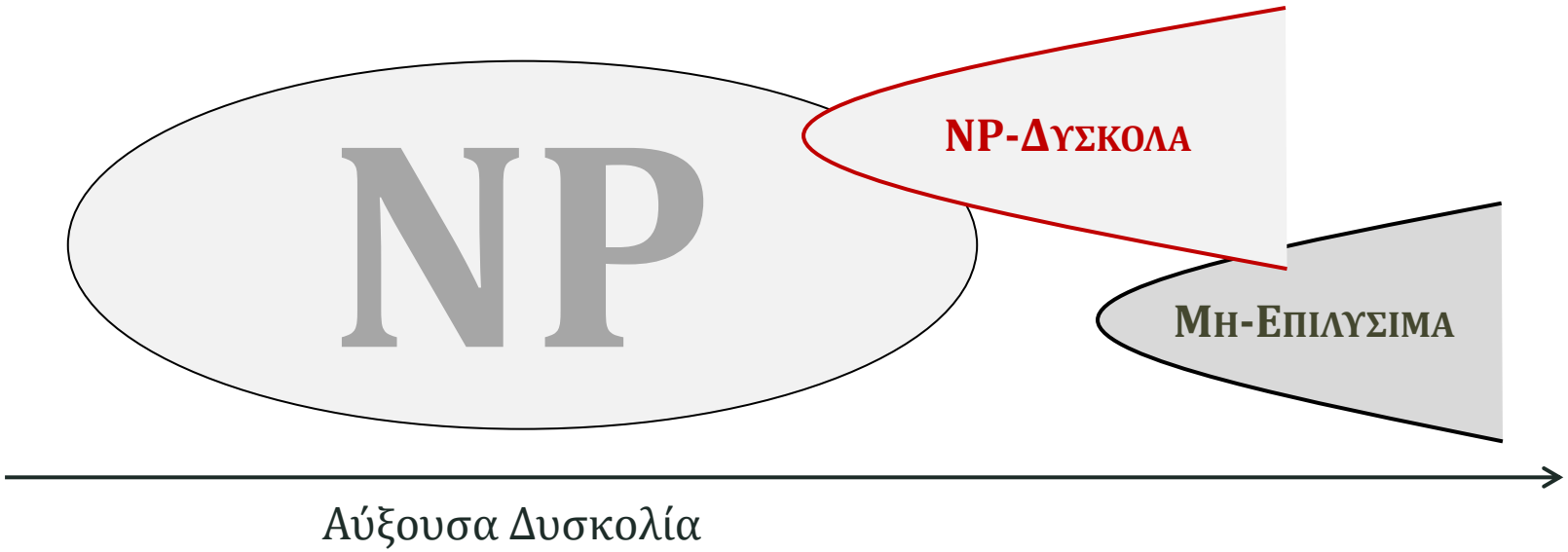
- Τι γίνεται **«πέρα»** και **«έξω»** από την περιοχή **NP** !!!



Μια Σχηματική Παρουσίαση !!!

Εύλογο **Ερώτημα** !!!

- **Τι γίνεται «πέρα» και «έξω» από την περιοχή NP !!!**

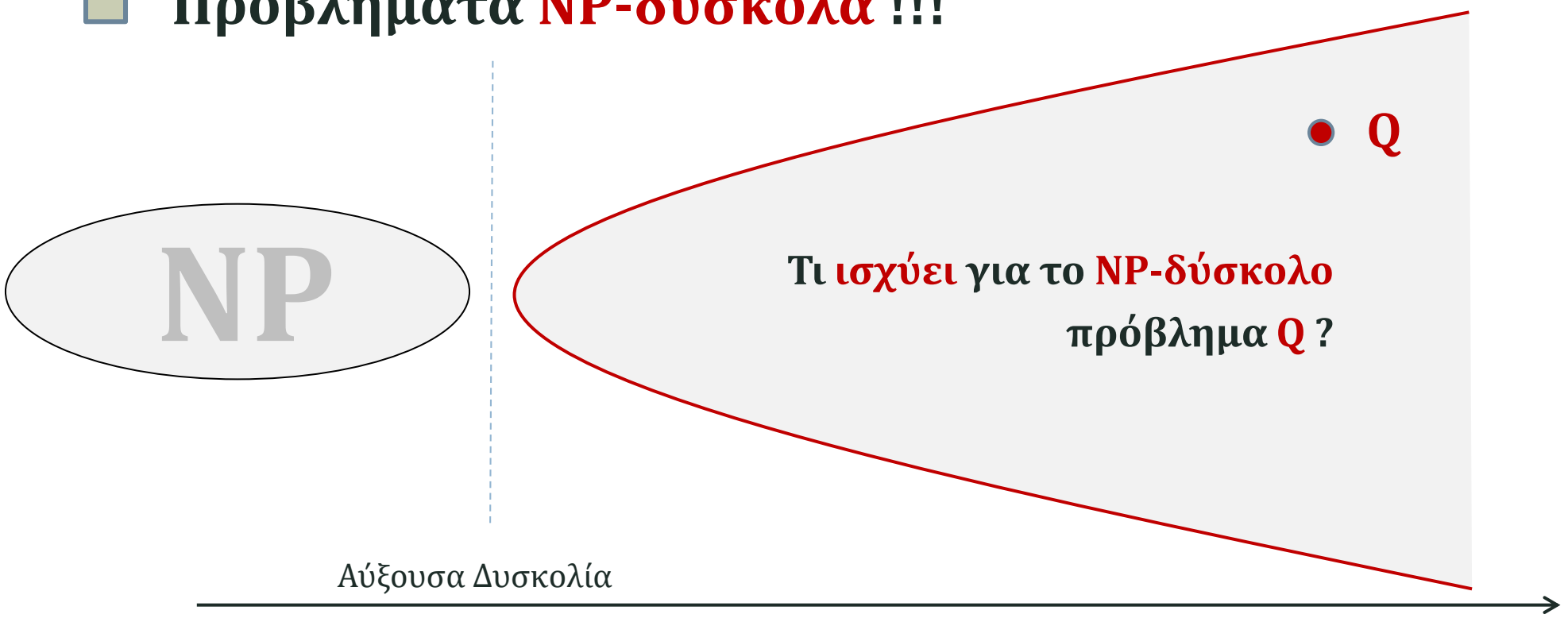


Μια Σχηματική Παρουσίαση !!!

- **Προβλήματα NP-δύσκολα !!!**

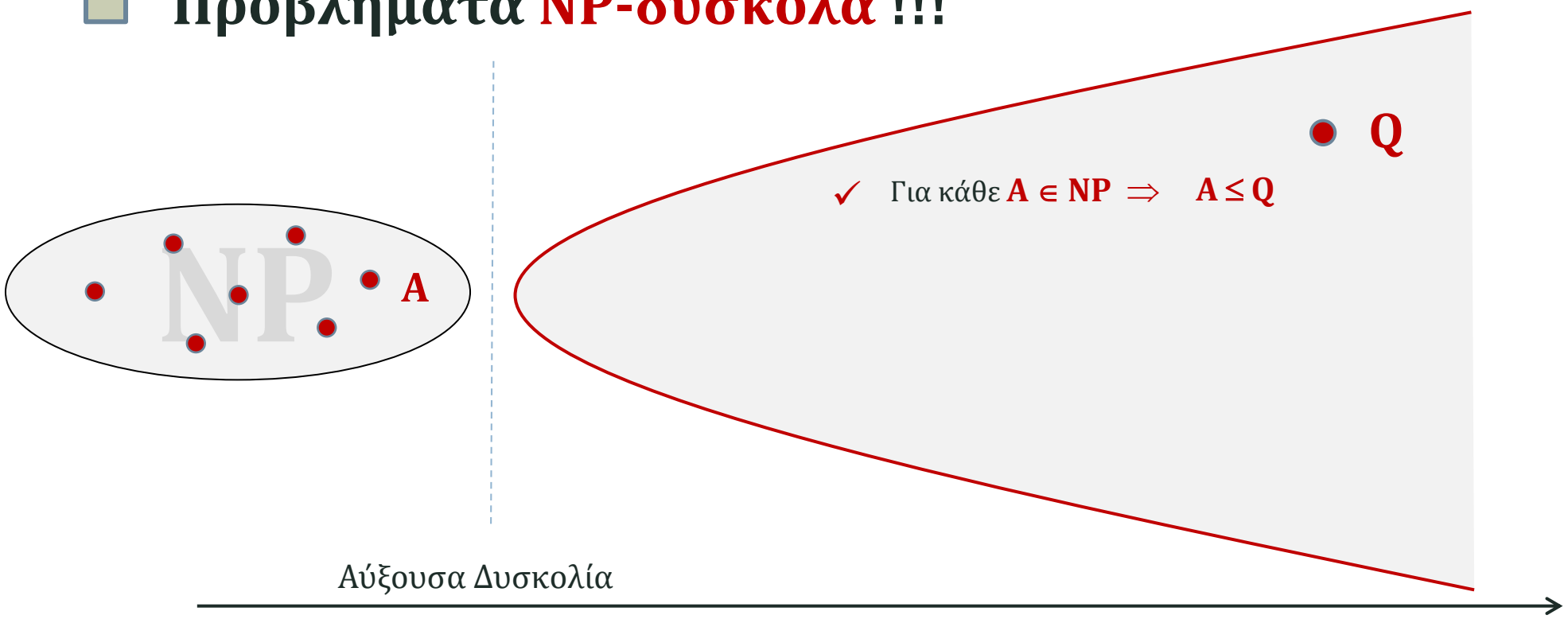
Μια Σχηματική Παρουσίαση !!!

- **Προβλήματα NP-δύσκολα !!!**



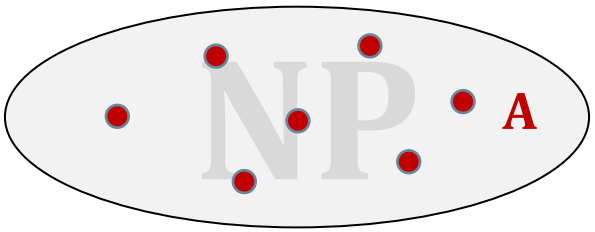
Μια Σχηματική Παρουσίαση !!!

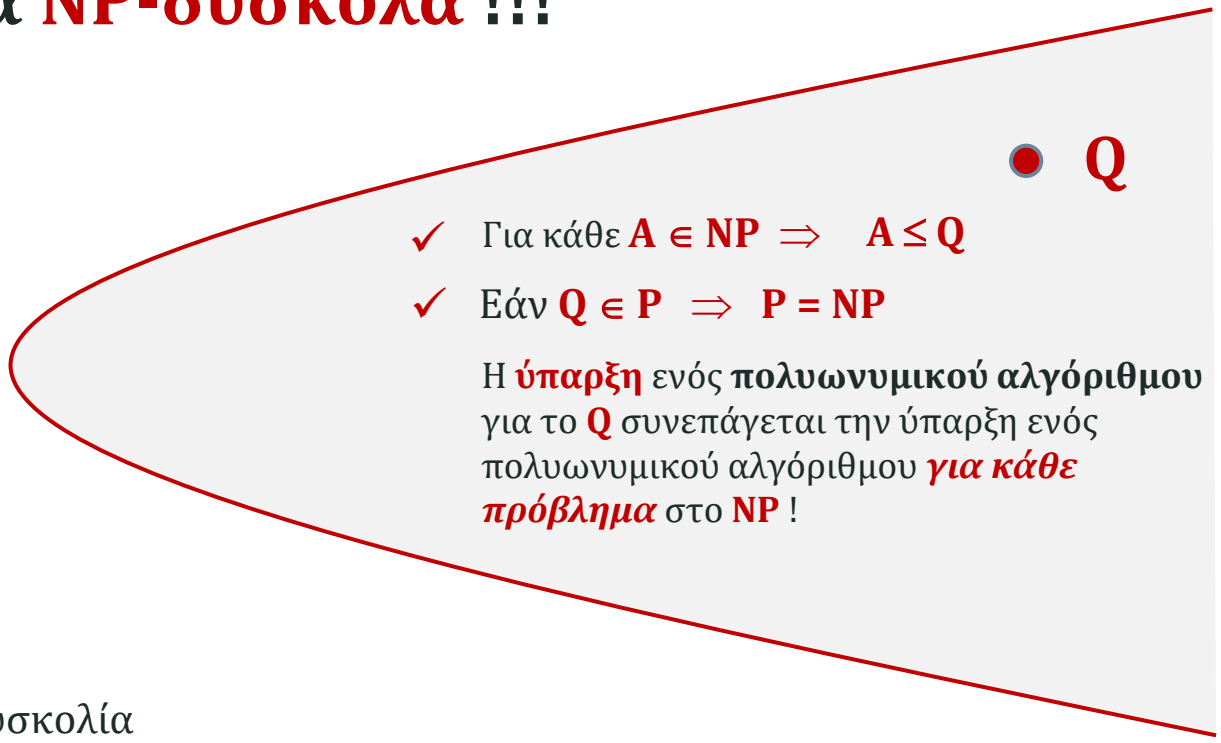
Προβλήματα NP-δύσκολα !!!



Μια Σχηματική Παρουσίαση !!!

Προβλήματα NP-δύσκολα !!!





Μια Σχηματική Παρουσίαση !!!

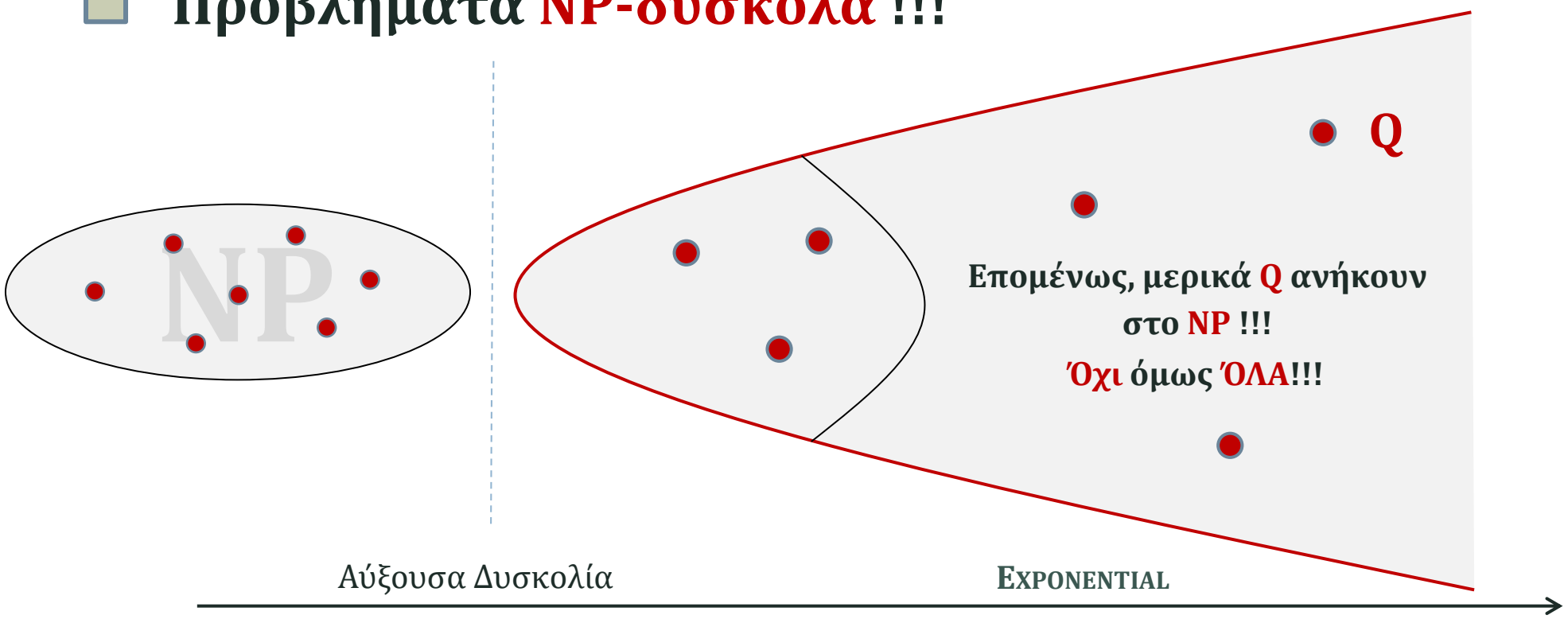
Προβλήματα NP-δύσκολα !!!

Μια Σχηματική Παρουσίαση !!!

Προβλήματα NP-δύσκολα !!!

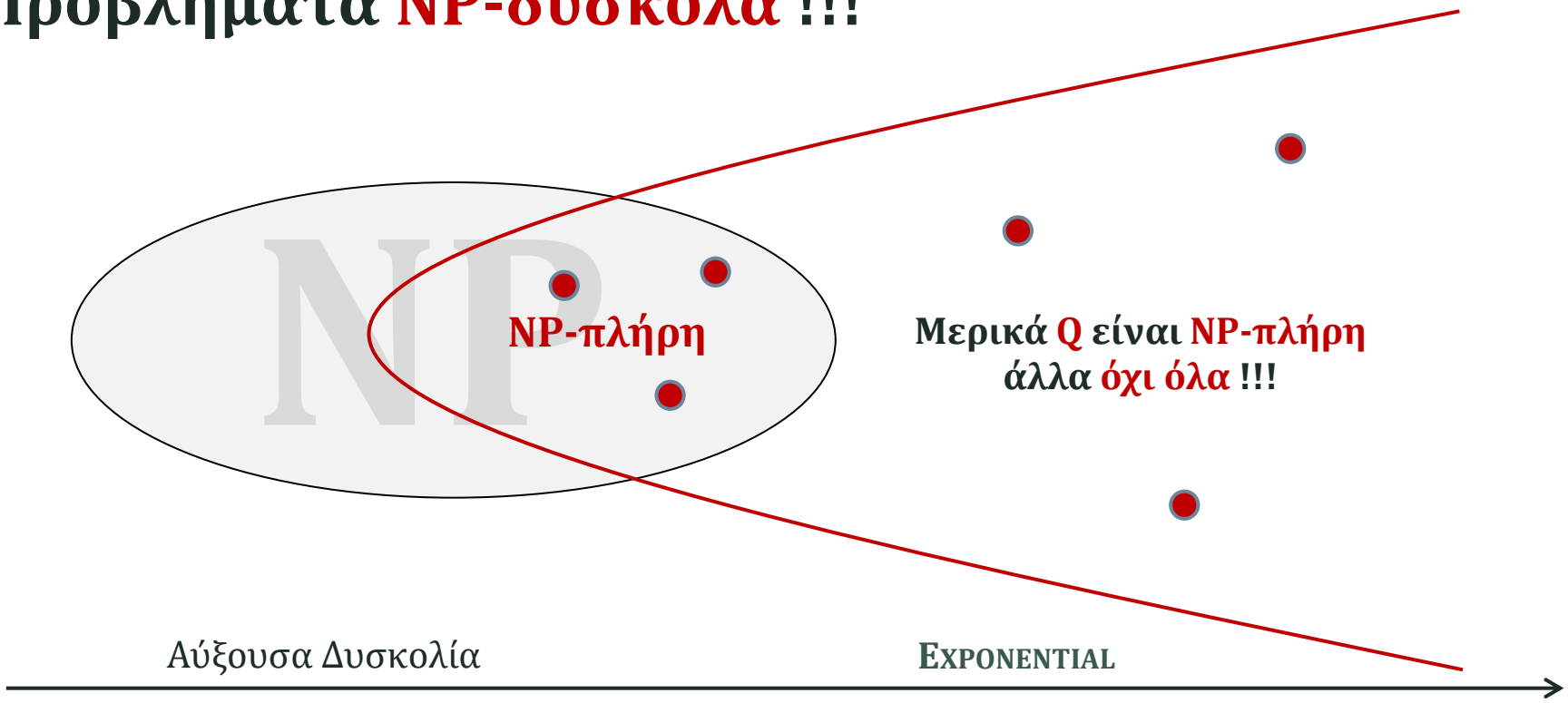
Μια Σχηματική Παρουσίαση !!!

- **Προβλήματα NP-δύσκολα !!!**



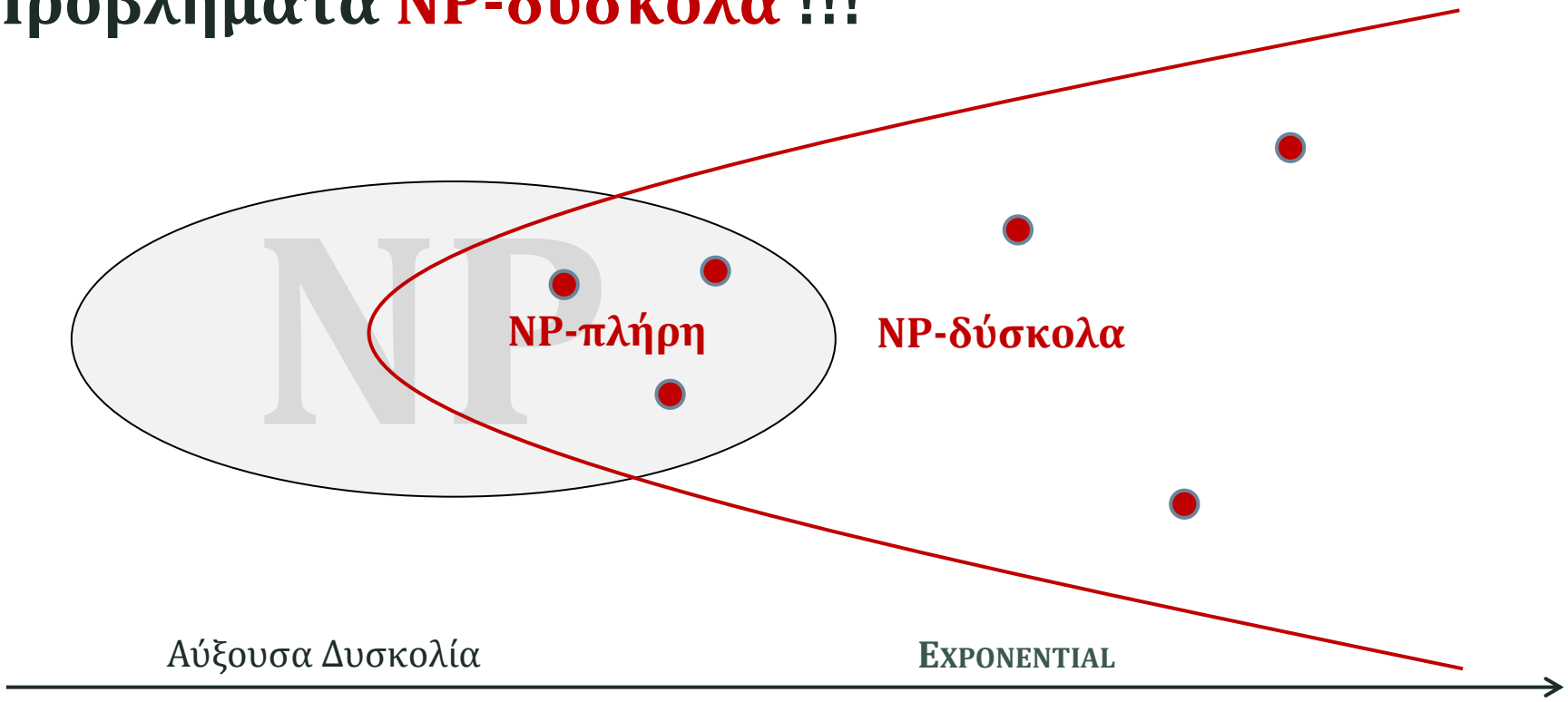
Μια Σχηματική Παρουσίαση !!!

- **Προβλήματα NP-δύσκολα !!!**



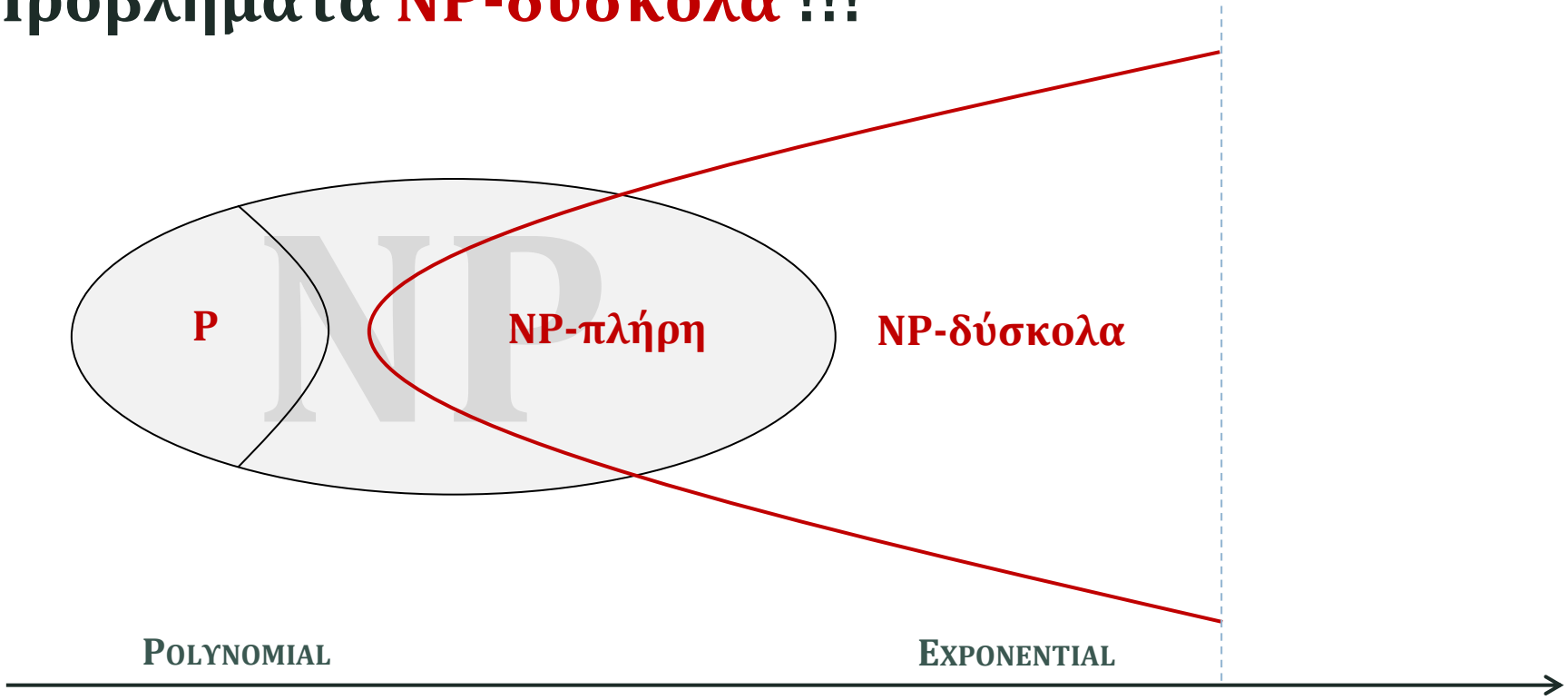
Μια Σχηματική Παρουσίαση !!!

- **Προβλήματα NP-δύσκολα !!!**



Μια Σχηματική Παρουσίαση !!!

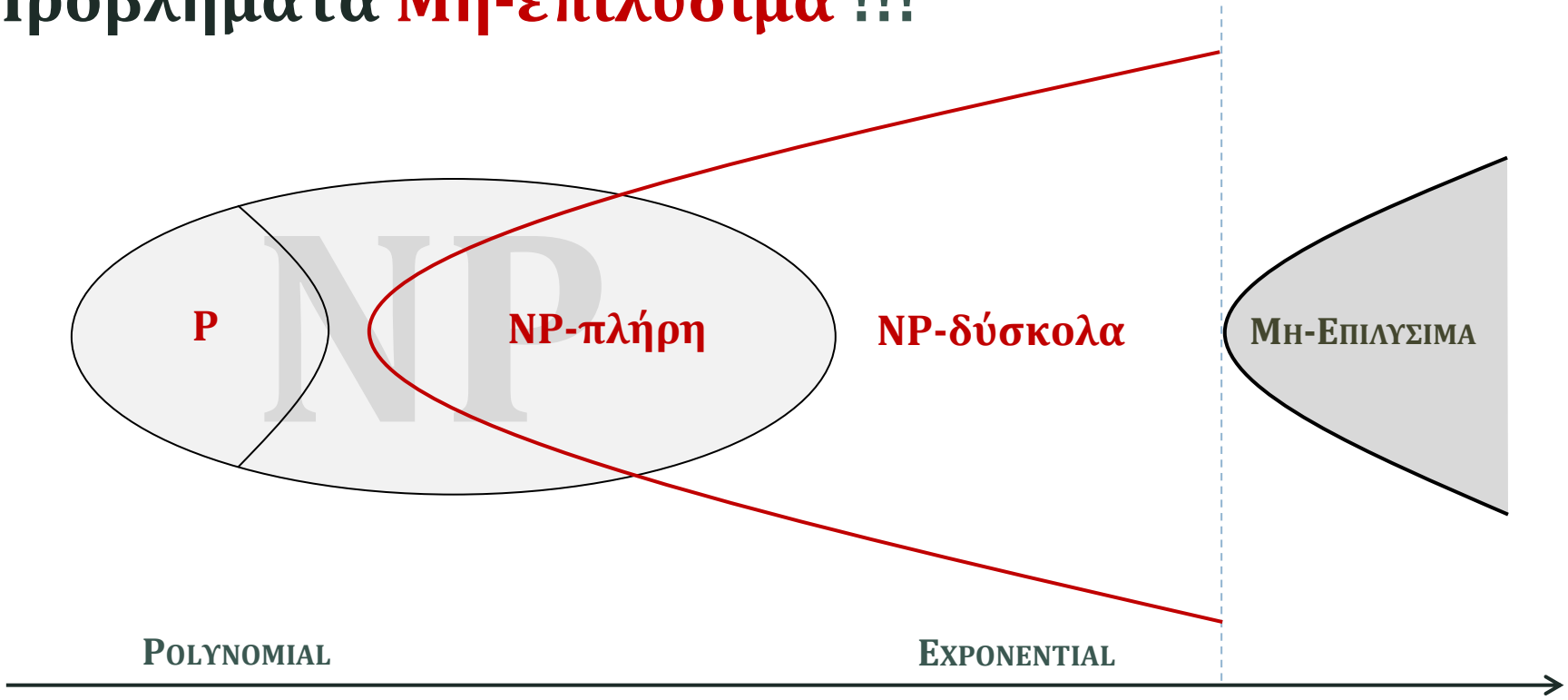
- **Προβλήματα NP-δύσκολα !!!**



Μια Σχηματική Παρουσίαση !!!

NP-πλήρη, NP-δύσκολα, Μη-επιλύσιμα

- **Προβλήματα Μη-επιλύσιμα !!!**



Μια Σχηματική Παρουσίαση !!!

Αριθμητική Εκδοχή του SAT !!!

- Ένα διάσημο πρόβλημα ***αυτού του είδους*** (μη-επιλύσιμο) είναι η ***αριθμητική εκδοχή*** του **SAT**

Με δεδομένη μια πολυωνυμική εξίσωση πολλών μεταβλητών, όπως για παράδειγμα

$$x^3yz + 2y^4z^2 - 7xy^5z = 6$$

υπάρχουν ακέραιες τιμές των μεταβλητών ***x, y, z*** οι οποίες να την ικανοποιούν ?

Το Πρόβλημα του Τερματισμού !!!

- Το **πρώτο μη-επιλύσιμο** πρόβλημα ανακαλύφθηκε το **1936** από τον **Alan Turing**, τότε φοιτητή των μαθηματικών στο Cambridge της Αγγλίας !!!

Το Πρόβλημα του Τερματισμού - HP

- ✓ Όταν το διατύπωσε ο Turing δεν υπήρχαν Η/Υ ή γλώσσες προγραμματισμού.
- ✓ Εμφανίστηκαν αργότερα ακριβώς επειδή είχε αυτή την ευφυή ιδέα !!!

Το Πρόβλημα του Τερματισμού !!!

- Έστω ένα *πρόγραμμα* **p** σε μια γλώσσα προγραμματισμού, μαζί με μια *είσοδο* **x** !!!

  Μπορείτε να απαντήσετε εάν το **p** με την είσοδο **x** θα μπορέσει ποτέ να **τερματίσει** ?

Τι θα θέλαμε ?

Έναν Αλγόριθμο, ας τον πούμε **`terminates(p,x)`**, ο οποίος μετά από υπολογισμούς θα μας έλεγε εάν το **p** με είσοδο **x**

- θα τερματίσει την εκτέλεσή του (`terminates(p,x) = YES`) ή
- θα εκτελείται επ' άπειρον (`terminates(p,x) = NO`) !!!

Σημείωση: Υπάρχουν προγράμματα (αλγόριθμοι) που δέχονται ως είσοδο προγράμματα, π.χ., οι μεταγλωττιστές !

Το Πρόβλημα του Τερματισμού !!!

- Λοιπόν, μην προσπαθήσετε… δεν μπορείτε!...

Τέτοιος αλγόριθμος δεν υπάρχει !!!

Να και η απόδειξη !!!

- Υποθέτουμε ότι έχουμε το πρόγραμμα **terminates(p,x)**.
- Τότε μπορούμε να κατασκευάσουμε το ακόλουθο πρόγραμμα:

```
paradox(z: file)
if terminates(z,z) then loop_forever else halt
```

Το Πρόβλημα του Τερματισμού !!!

```
paradox(z: file)
if terminates(z,z) then loop_forever else halt
```

- Το **paradox** ***τερματίζει*** **εάν-ν** **terminates(z,z) = NO** όταν του δοθεί ως είσοδος ο δικός του κώδικας.

  Δηλαδή:

  το **paradox** ***τερματίζει*** **εάν-ν** το **z** ***δεν τερματίζει*** !!!

Το Πρόβλημα του Τερματισμού !!!

```
paradox(z: file)
if terminates(z,z) then loop_forever else halt
```

- Το **paradox** ***τερματίζει*** **εάν-ν** το **z** ***δεν τερματίζει*** !!!

  Εάν εκτελούσαμε το πρόγραμμα **paradox(paradox)** ?

  Τότε,

  το **paradox** **τερματίζει** **εάν-ν** το **paradox** **δεν τερματίζει** **!!!**

  Αντίφαση !!!

  Η μόνη υπόθεση που κάναμε είναι η ύπαρξη του προγράμματος **terminates()**, **άρα τέτοιο πρόγραμμα δεν μπορεί να υπάρχει !!!**

Το Πρόβλημα του Τερματισμού !!!

- Καταλήξαμε σε αντίφαση !!!

Επομένως, το πρόβλημα HP δεν μπορεί να λυθεί με *κανένα αλγόριθμο* !!!

Αυτό μας λέει κάτι σημαντικό:

- ✓ **Ο προγραμματισμός ποτέ δεν θα αυτοματοποιηθεί** !!!
- ✓ Και πάντοτε θα βασίζεται στην **επινοητικότητα** !!!.. και στη **σοβαρή μελέτη** !!!

Ευχαριστώ για τη Συμμετοχή σας !!!

Σταύρος Δ. Νικολόπουλος

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Ιωαννίνων

Webpage: www.cs.uoi.gr/~stavros