Σχεδίαση και Ανάλυση Αλγορίθμων

Ενότητα 9.0

Δυναμικός & Γραμμικός Προγραμματισμός

Προβλήματα

Ροές σε Δίκτυα

Σταύρος Δ. Νικολόπουλος | 2016-17

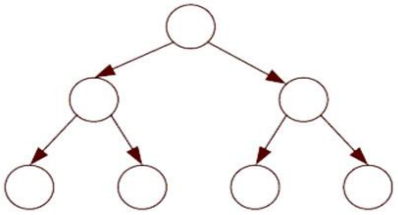
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Ιωαννίνων

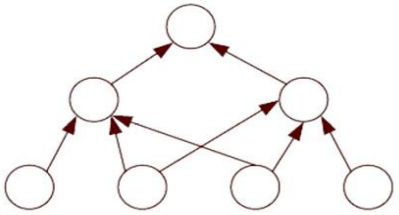
Webpage: www.cs.uoi.gr/~stavros

Δύο Σημαντικά Εργαλεία της Τέχνης των Αλγορίθμων

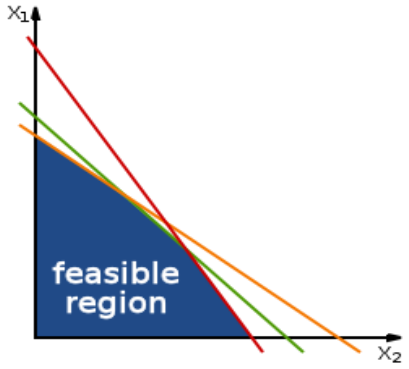
- ✓ **Δυναμικός Προγραμματισμός**
- ✓ **Γραμμικός Προγραμματισμός**



DIVIDE AND CONQUER



DYNAMIC PROGRAMMING



Δυναμικός & Γραμμικός Προγραμματισμός

Τι έχουμε δει έως τώρα ?

- Μερικές τεχνικές σχεδίασης αλγορίθμων, όπως:
  - ✓ Διαίρει-και-Βασίλευε
  - ✓ Άπληστη Επιλογή
- Το *μειονέκτημα* αυτών των τεχνικών είναι ότι μπορούν να χρησιμοποιηθούν για *ειδικούς τύπους προβλημάτων* !!!

Δυναμικός Προγραμματισμός και ***Γραμμικός Προγραμματισμός***

Δύο Γενικές Αλγοριθμικές Τεχνικές !!!

- **Αλγοριθμική Τεχνική**

Δυναμικός Προγραμματισμός

Those who cannot remember the past are condemned to repeat it.

-Dynamic Programming

- **Αλγοριθμική Τεχνική**

Δυναμικός Προγραμματισμός

- **Είναι… Τεχνική Σχεδίασης Αλγορίθμων !!!**
  - ✓ με **ευρύτατο πεδίο εφαρμογών !**
  - ✓ και με χρήση εκεί όπου **άλλες** πιο ειδικευμένες **τεχνικές αποτυγχάνουν !**

Δεν είναι… Τεχνική Προγραμματισμού !!!

Δυναμικός Προγραμματισμός

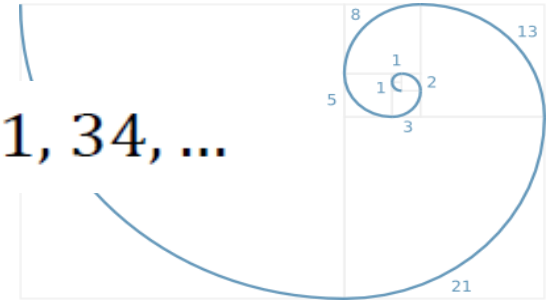
- **Γιατί αυτός ο όρος και η ονομασία**

Δυναμικός Προγραμματισμός

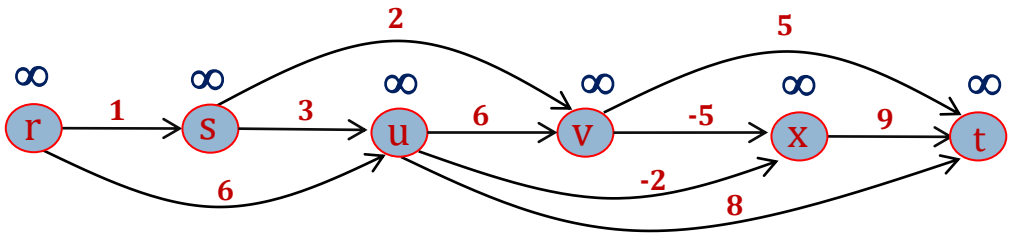
- **Επινοήθηκε από τον Richard Bellman το 1950**
  - Τότε ο **προγραμματισμός** ήταν μια απόκρυφη, σπάνια δραστηριότητα την οποία ασκούσαν τόσο πολύ λίγοι που δεν άξιζε καν να την αναφέρουν.
  - Ο όρος «προγραμματισμός» την εποχή εκείνη σήμαινε **«σχεδιασμός ενεργειών»** (planning), ενώ
  - ο όρος «δυναμικός προγραμματισμός» σήμαινε **«βέλτιστο σχεδιασμό πολλαπλών διεργασιών»** !!!

- **Ας θυμηθούμε δύο γνωστά μας προβλήματα !**
  - **Ακολουθία Fibonacci**

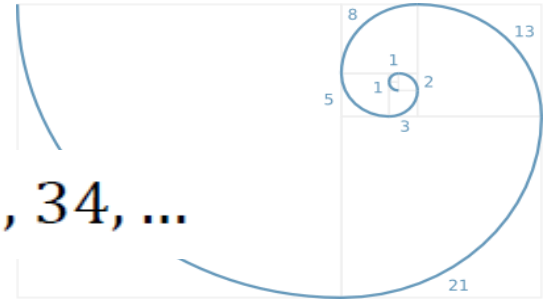
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …



  - **Ελάχιστες διαδρομές σε DAG**



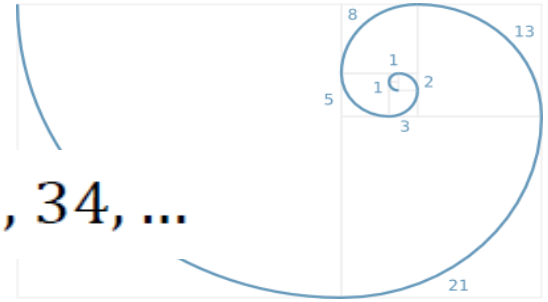
- **Ακολουθία Fibonacci**



0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

$$F_n = \begin{cases} 0 & \text{εάν } n = 0 \\ 1 & \text{εάν } n = 1 \\ F_{n-1} + F_{n-2} & \text{εάν } n > 1 \end{cases}$$

- **Ακολουθία Fibonacci**

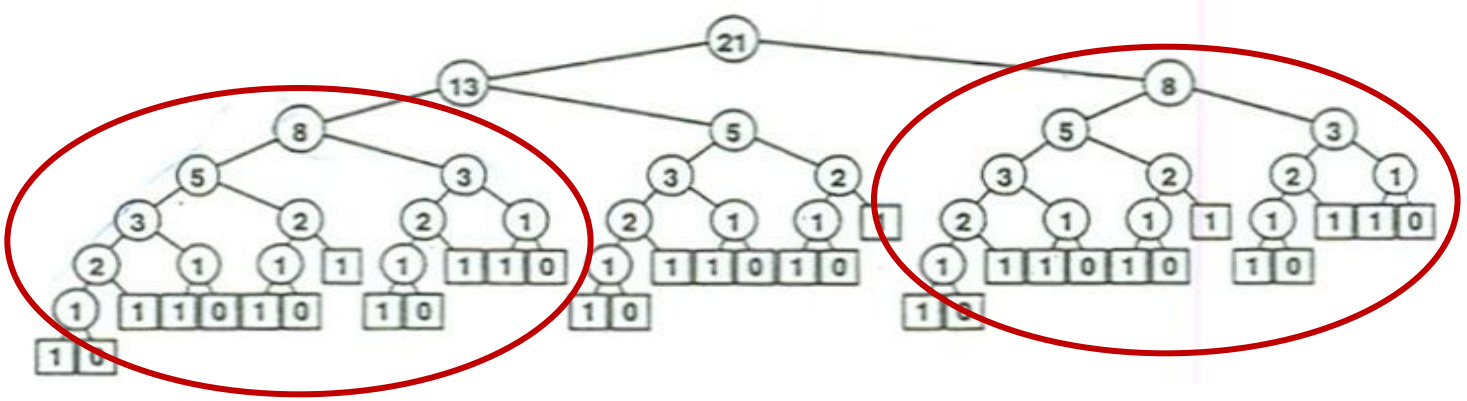


0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

```
fib1(n)
   if n = 0 then return 0
   if n = 1 then return 1
   return fib1(n-1) + fib1(n-2)
```

Ακολουθία Fibonacci

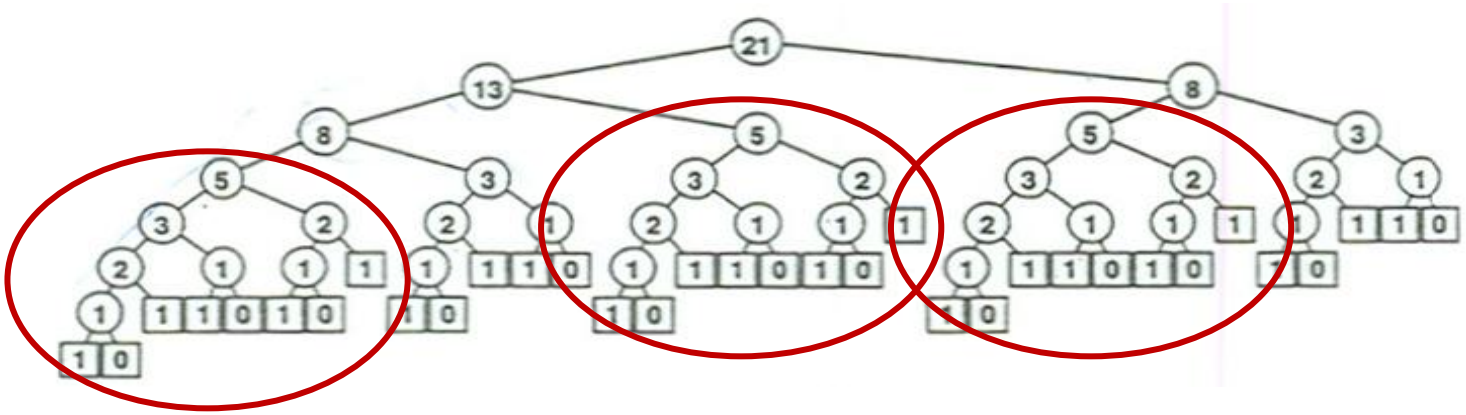
```
fib1(n)
   if n = 0 then return 0
   if n = 1 then return 1
   return fib1(n-1) + fib1(n-2)
```



```
8 F(6)
  5 F(5)
    3 F(4)
      2 F(3)
        1 F(2)
          1 F(1)
          0 F(0)
        1 F(1)
      1 F(2)
        1 F(1)
        0 F(0)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
  3 F(4)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
    1 F(2)
      1 F(1)
      0 F(0)
```

Ακολουθία Fibonacci

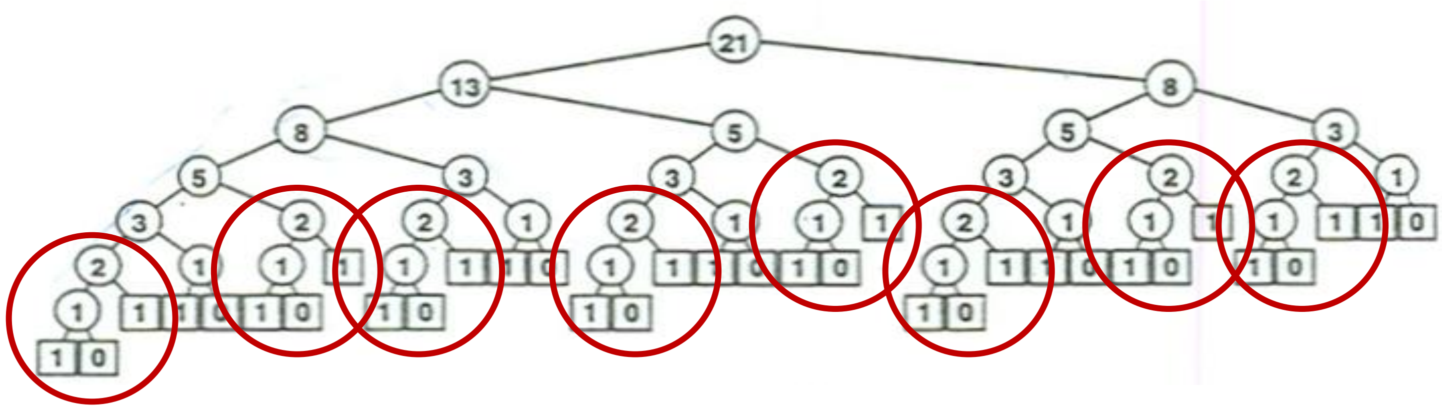
```
fib1(n)
   if n = 0 then return 0
   if n = 1 then return 1
   return fib1(n-1) + fib1(n-2)
```

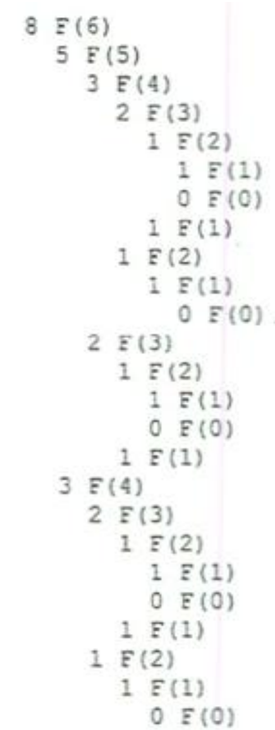


```
8 F(6)
  5 F(5)
    3 F(4)
      2 F(3)
        1 F(2)
          1 F(1)
          0 F(0)
        1 F(1)
      1 F(2)
        1 F(1)
        0 F(0)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
  3 F(4)
    2 F(3)
      1 F(2)
        1 F(1)
        0 F(0)
      1 F(1)
    1 F(2)
      1 F(1)
      0 F(0)
```

Ακολουθία Fibonacci

```
fib1(n)
   if n = 0 then return 0
   if n = 1 then return 1
   return fib1(n-1) + fib1(n-2)
```



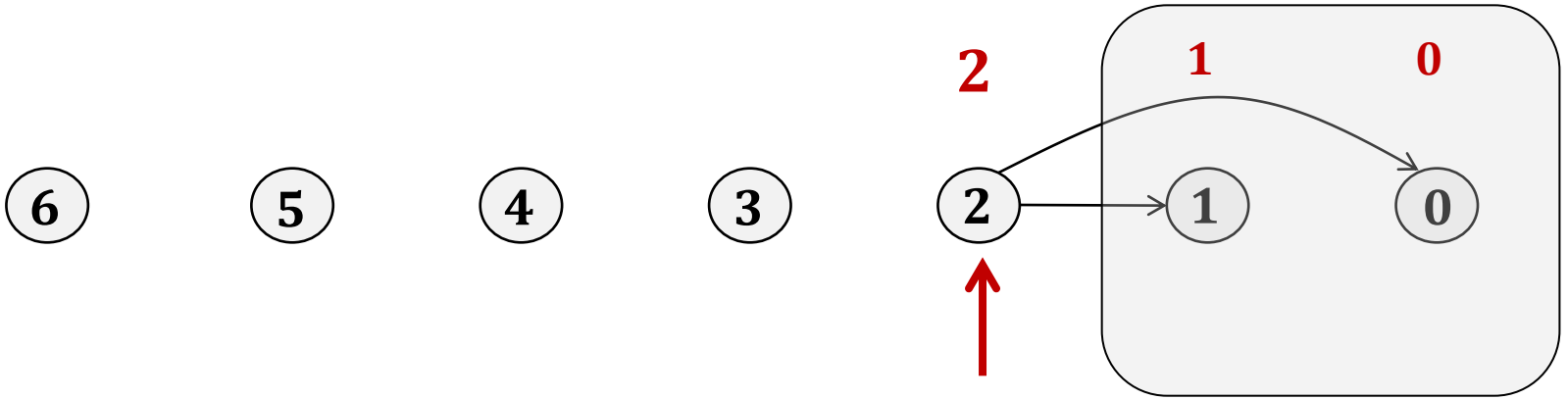


- **Ακολουθία Fibonacci**

6 5 4 3 2 1 0

```
Fib(n) = fib(n-1) + fib(n-2)
```

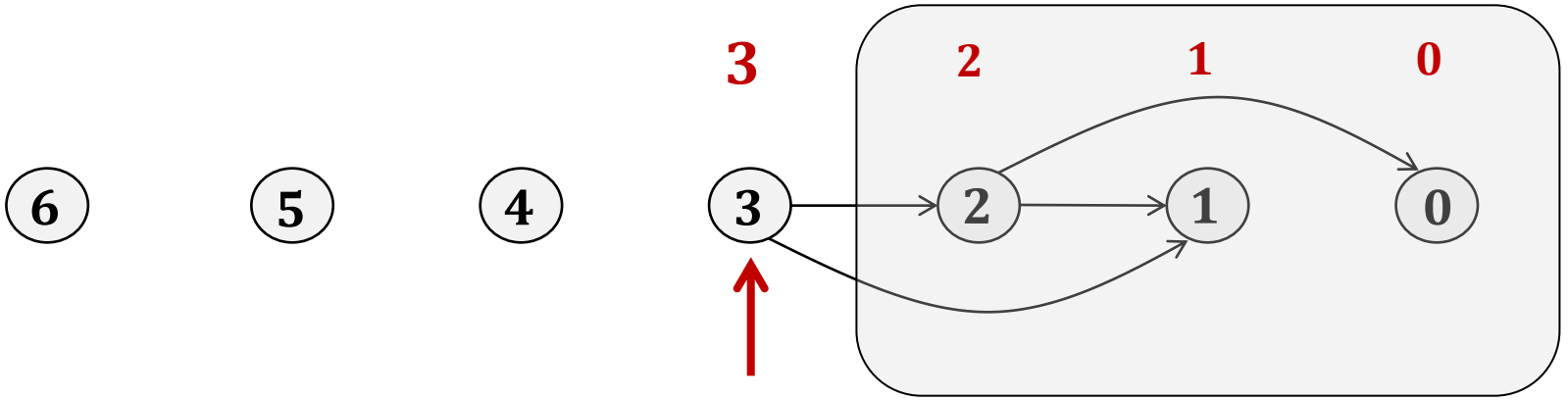
- **Ακολουθία Fibonacci**



`Fib(2) = fib(1) + fib(0)`

```
Fib(n) = fib(n-1) + fib(n-2)
```

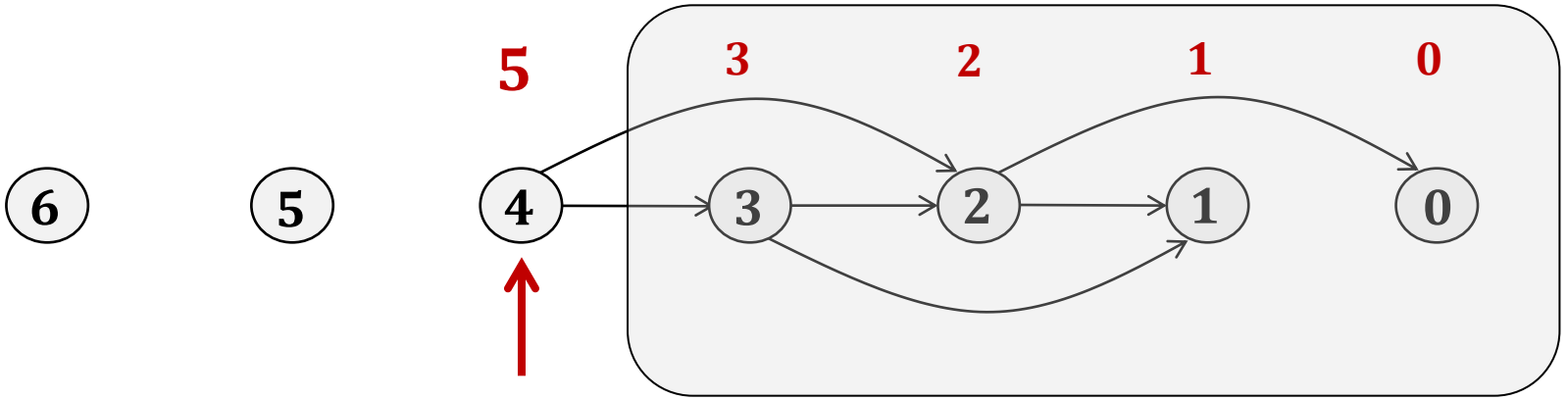
- **Ακολουθία Fibonacci**




```
Fib(3) = fib(2) + fib(1)
```

```
Fib(n) = fib(n-1) + fib(n-2)
```

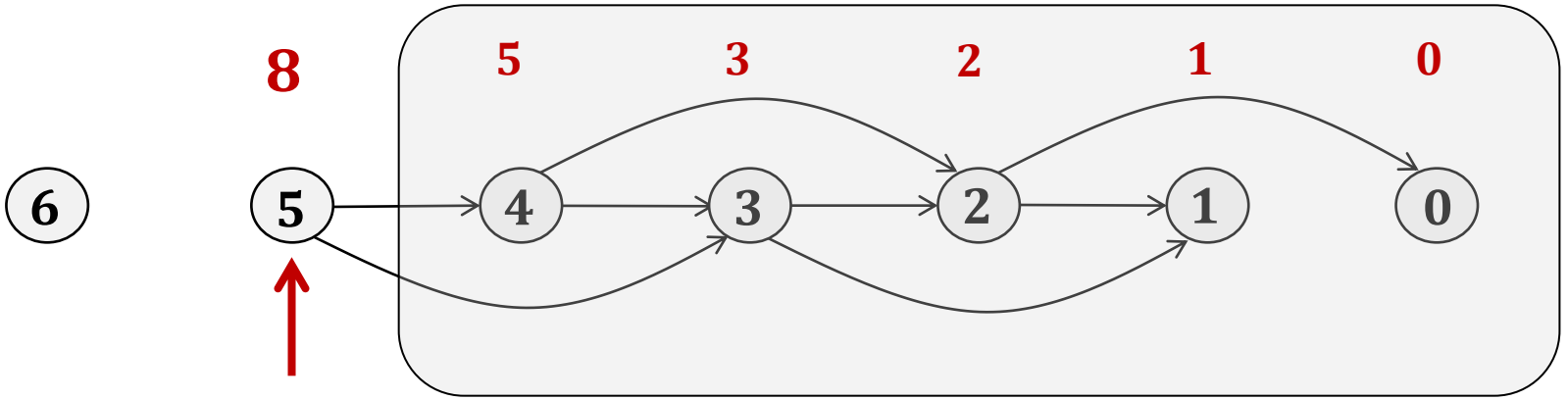
- **Ακολουθία Fibonacci**




```
Fib(4) = fib(3) + fib(2)
```

```
Fib(n) = fib(n-1) + fib(n-2)
```

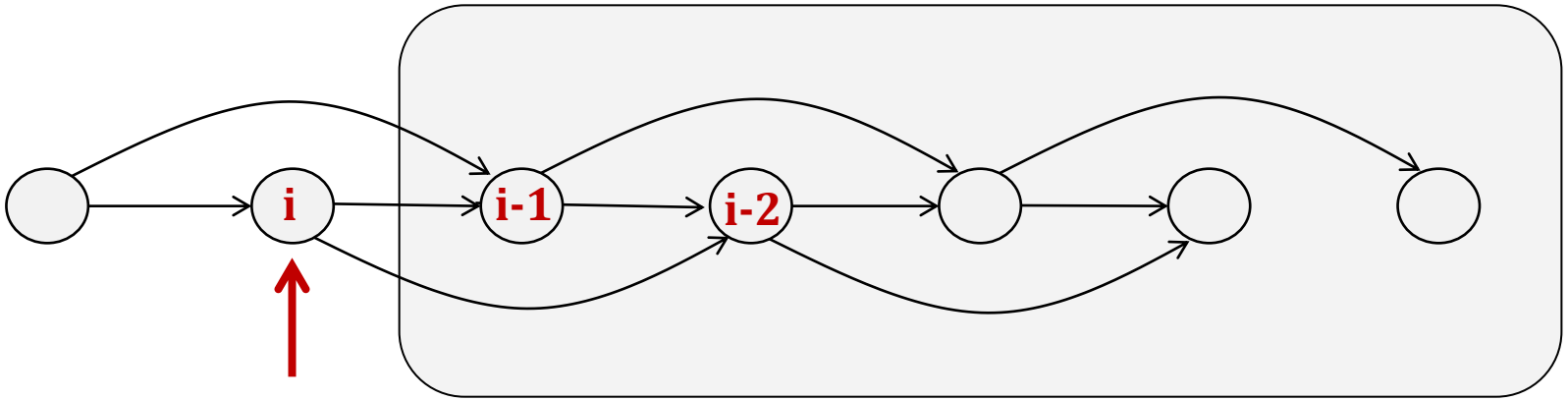
- **Ακολουθία Fibonacci**




```
Fib(5) = fib(4) + fib(3)
```

```
Fib(n) = fib(n-1) + fib(n-2)
```

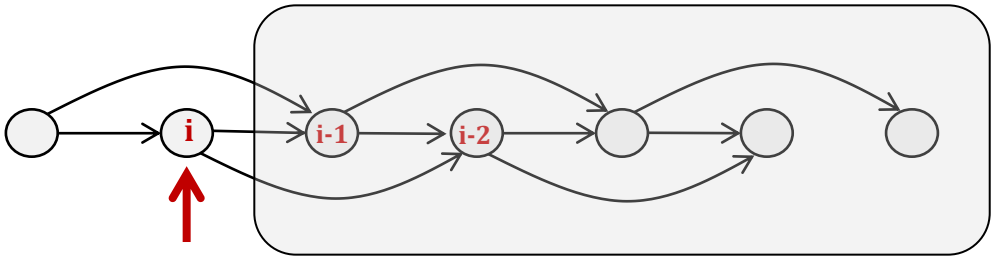
- **Ακολουθία Fibonacci**




```
Fib(i) = fib(i-1) + fib(i-2)
```

Μπορώ να λύσω ένα υποπρόβλημα εύκολα
εάν έχω **«βρει»** και έχω **«κρατήσει»** τις λύσεις **«μικρότερων»** υποπροβλημάτων !!!

Ακολουθία Fibonacci

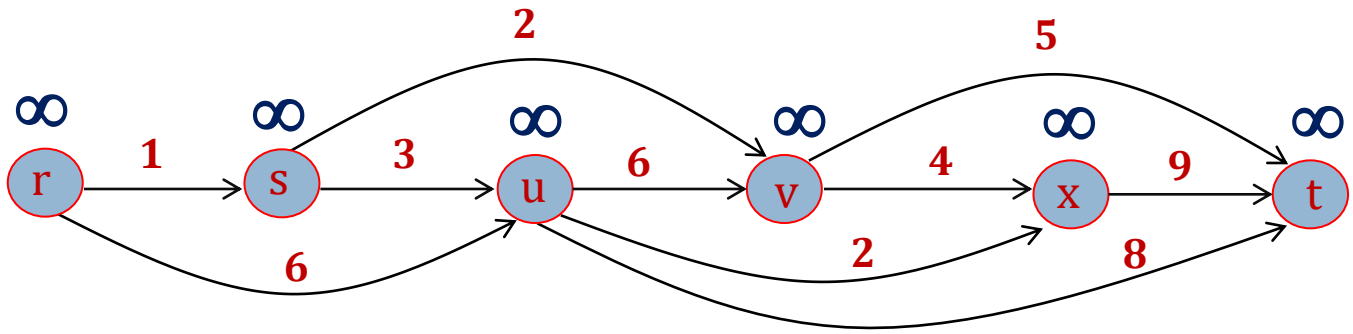


Αλγόριθμος FIBONACCI

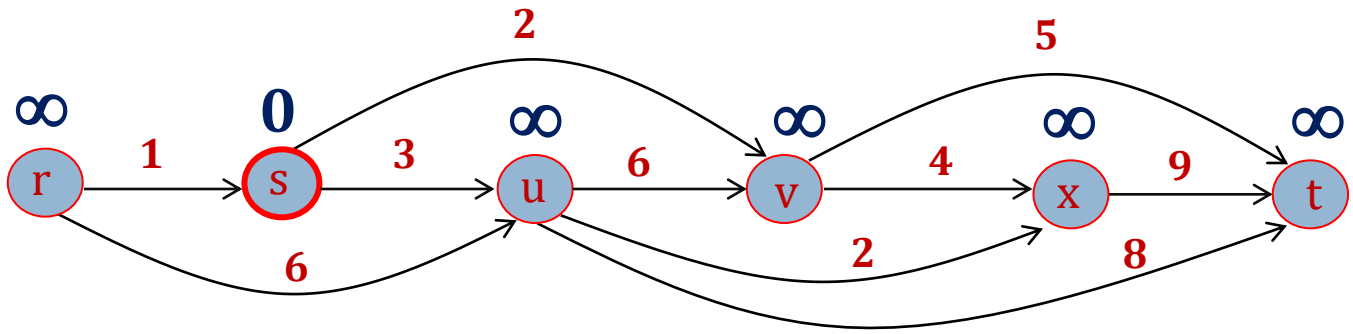
```
fib2(n)
1. if n = 0 then return 0
2. if n = 1 then return 1
3. F(0) = 0, F(1) = 1 {F πίνακας μήκους n}
4. for i = 2, 3, …, n
        F(i) = F(i-1) + F(i-2)
5. return F(n)
```

Κρατήστε αυτά τα 2 στοιχεία του αλγορίθμου !!!

- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση

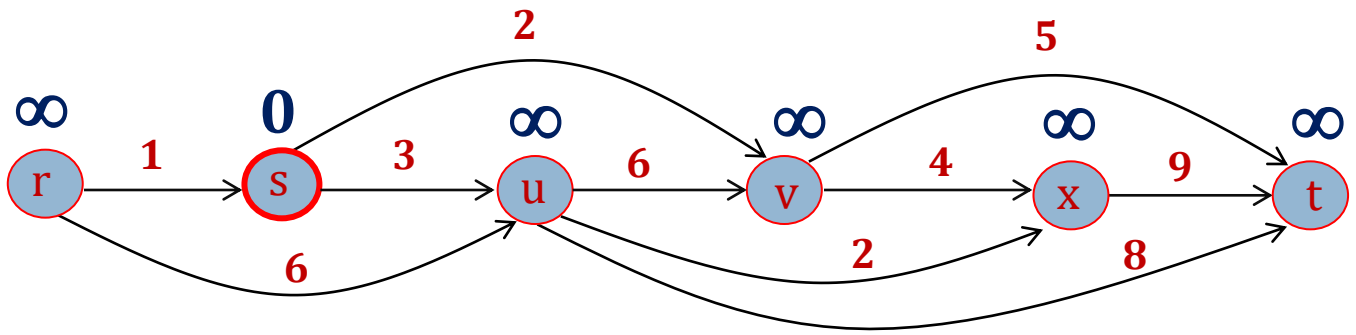


- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση



  - **Γιατί** αυτό το Χαρακτηριστικό βοηθάει στον υπολογισμό των ε.δ από ένα κόμβο, έστω τον **s** ?

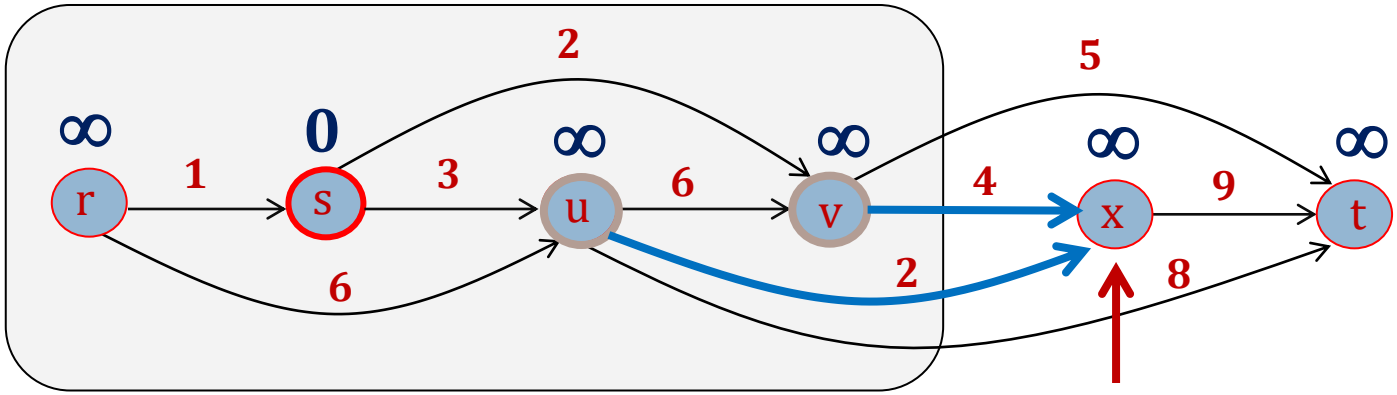
- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση



  - Θέλουμε να υπολογίσουμε τις ε.δ από τον κόμβο **s** :

```
d(r), d(s), d(u), d(v), d(x), d(t)
```

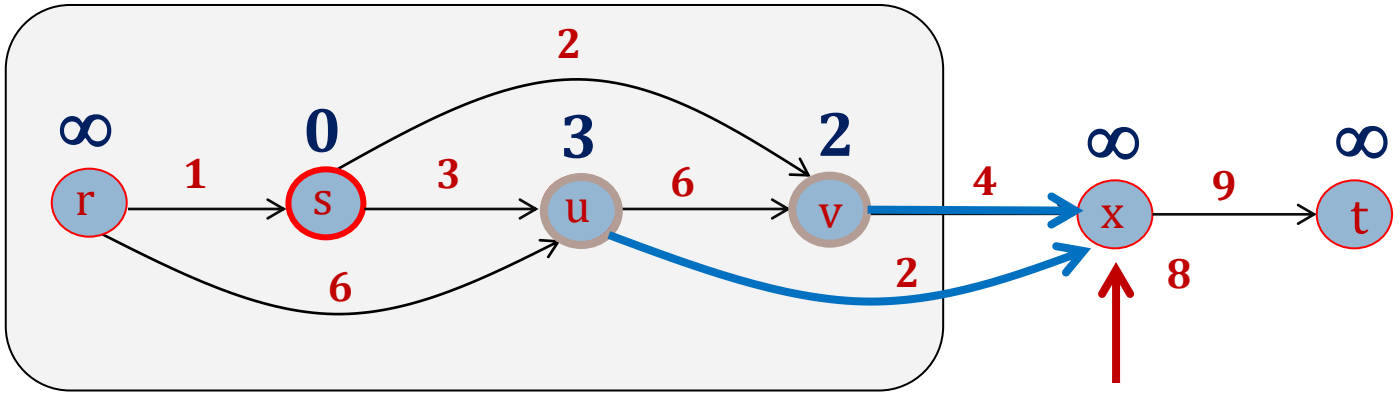
- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση



  - Ας εστιάσουμε σε ένα κόμβο, έστω στον κόμβο **x**

> Ο μόνος τρόπος για να φθάσουμε στον x είναι μέσω των προκατόχων του, δηλ. των κόμβων v και u !!!

- **Ελάχιστες διαδρομές σε DAG**
  - Χαρακτηριστικό ενός DAG: Τοπολογική Ταξινόμηση

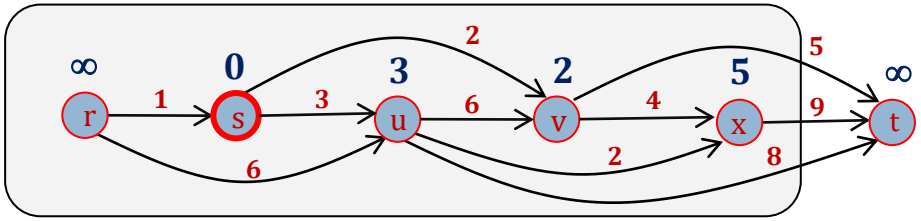


  - Εάν έχουμε υπολογίσει τις τιμές d(r), d(s), d(u) και d(v) των αντίστοιχων ε.δ, μπορούμε βρούμε την ε.δ **s ~> x, αρκεί μόνο να συγκρίνουμε τις δύο διαδρομές:**

$$d(x) = \min\{d(v) + 4,\ d(u) + 2\}$$

Ελάχιστες διαδρομές σε DAG

- Μια τέτοια σχέση ισχύει για ∀ κόμβο !!!
  π.χ., για τον κόμβο **t**:

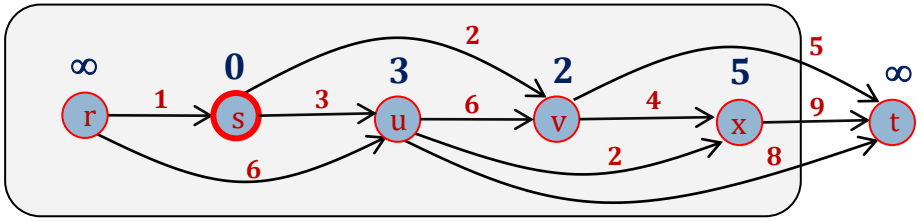


d(t) = min{ d(x)+9, d(v)+5, d(u)+8 }

- Αν υπολογίσουμε τις τιμές **d(.)** με τη σειρά της Τοπολογικής Ταξινόμησης τότε:

Μπορώ να λύσω ένα υποπρόβλημα εύκολα
εάν έχω **«βρει»** και έχω **«κρατήσει»** τις λύσεις **«μικρότερων»** υποπροβλημάτων !!!

- **Ελάχιστες διαδρομές σε DAG**



Αλγόριθμος SP-DAG

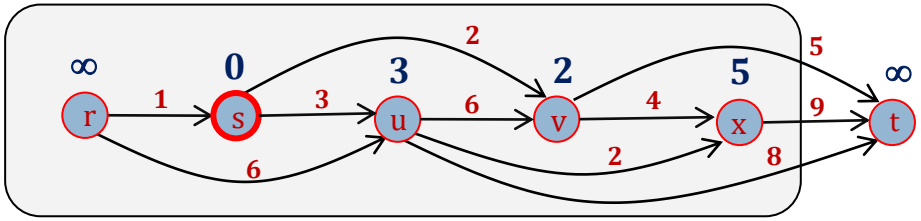
1. Initialize(G,s)
2. Topological-Sorting(G)
3. για κάθε κόμβο $x \in V-\{s\}$ σε τοπολογική σειρά:

$$d(x) = \min_{(u,x)\in E}\{d(u) + w(u,x)\}$$

4. return d(.)

Κρατήστε αυτά τα 2 στοιχεία του αλγόριθμου !!!

Ελάχιστες διαδρομές σε DAG



Παρατηρήσεις !!!

- Ο αλγόριθμος SP-DAG επιλύει υποπροβλήματα, της μορφής: $\{ d(x) \mid x \in V-\{s\} \}$

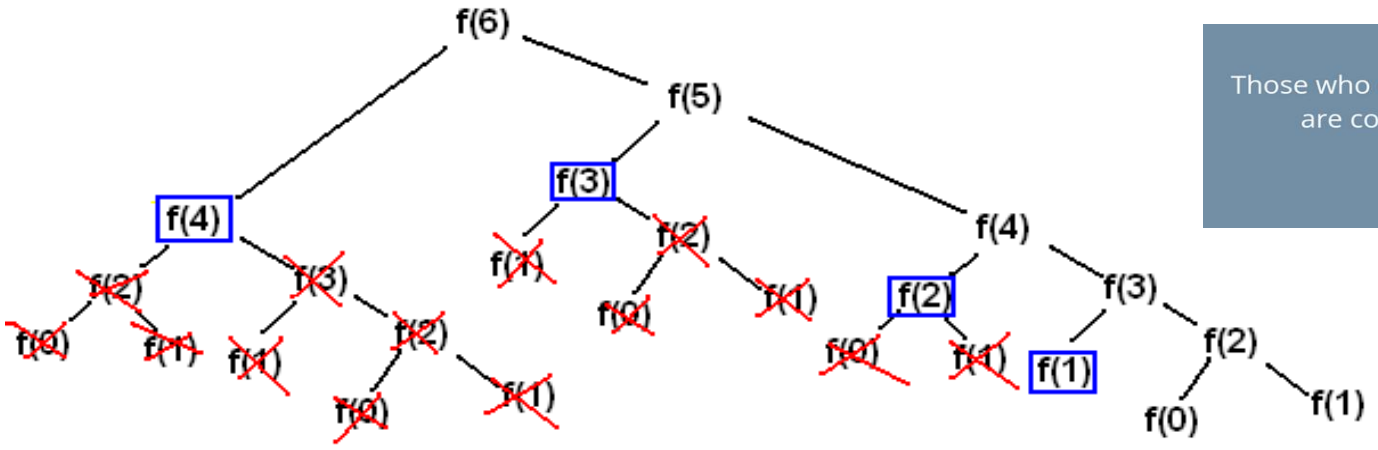
$$d(x_1),\ d(x_2),\ \ldots,\ d(x_i),\ \ldots,\ d(x_{n-1}),\ d(x_n)$$

- **Αρχίζει** από το **«μικρότερο»** υποπρόβλημα και **προσχωράει σε «μεγαλύτερα» υποπροβλήματα !!!**

> Ένα υποπρόβλημα το θεωρούμε **«μεγάλο»** εάν πρέπει να λύσουμε πολλά άλλα υποπροβλήματα πριν φθάσουμε σε αυτό !!!

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Η Βασική Ιδέα !!!**



Those who cannot remember the past are condemned to repeat it.

-Dynamic Programming

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Η Βασική Ιδέα !!!...** Λύσης ενός προβλήματος Π με ΔΠ:

1. **Υπολογίζουμε** ένα **σύνολο υποπροβλημάτων** του Π
2. **Επινοούμε** μια **σχέση** για την λύση ενός υποπροβλήματος
3. **Λύνουμε τα υποπροβλήματα** ξεκινώντας από «μικρότερα», **αποθηκεύουμε** τις λύσεις τους, και προχωράμε προς «μεγαλύτερα» **χρησιμοποιώντας τις αποθηκευμένες λύσεις των μικρότερων** !!!
4. **Παίρνουμε** τη λύση του αρχικού μας προβλήματος Π, λύνοντας όλα τα υποπροβλήματα με **καθορισμένη σειρά** !!!

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- Κοινά χαρακτηριστικά των προβλημάτων του ΔΠ :

  - **Ιδιότητα Βέλτιστης υποδομής** (Optimal substructure property):

    Μια βέλτιστη λύση του προβλήματος μπορεί να κατασκευαστεί από βέλτιστες λύσεις των υποπροβλημάτων του !

  - **Πολλαπλά Υποπροβλήματα** : Το αρχικό πρόβλημα διασπάται, με πολλαπλούς τρόπους, σε ένα σύνολο απλούστερων υποπροβλημάτων !!

  - **Επικάλυψη Υποπροβλημάτων** : Ανεξάρτητα υποπροβλήματα είναι δυνατόν να μοιράζονται βέλτιστες λύσεις κοινών υποπροβλημάτων !!!

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Θεμελιώδης Ιδιότητα του ΔΠ !!!**

  - ✓ Στο ΔΠ το «**μοντέλο της τεχνικής**» θα μπορούσαμε να το θεωρήσουμε ότι είναι ένα γράφημα **G** τύπου DAG !!!

  - ✓ Οι κόμβοι του **G** αντιστοιχούν στα υποπροβλήματα και οι ακμές του στις εξαρτήσεις ανάμεσα σε αυτά:

    - Το **A** θεωρείται «μικρότερο» υποπρόβλημα από το **B,** ή
    - Για να λύσουμε το **B** χρειαζόμαστε την λύση του **A** !!!

Χαρακτηριστικά Δυναμικού Προγραμματισμού

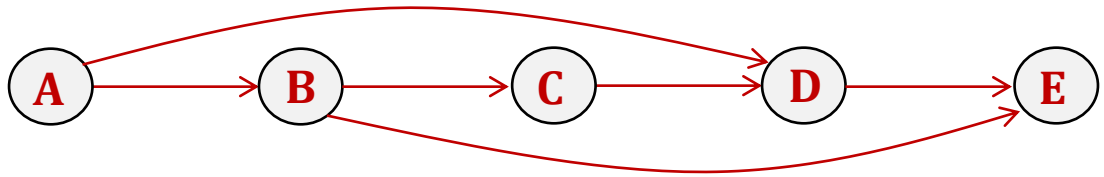
- **Θεμελιώδης Ιδιότητα του ΔΠ !!!**

Αυτό σημαίνει ότι υπάρχει **διάταξη** των υποπροβλημάτων και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα

έχοντας τις λύσεις «μικροτέρων» υποπροβλημάτων,

δηλαδή, υποπροβλημάτων που εμφανίζονται νωρίτερα στη διάταξη !

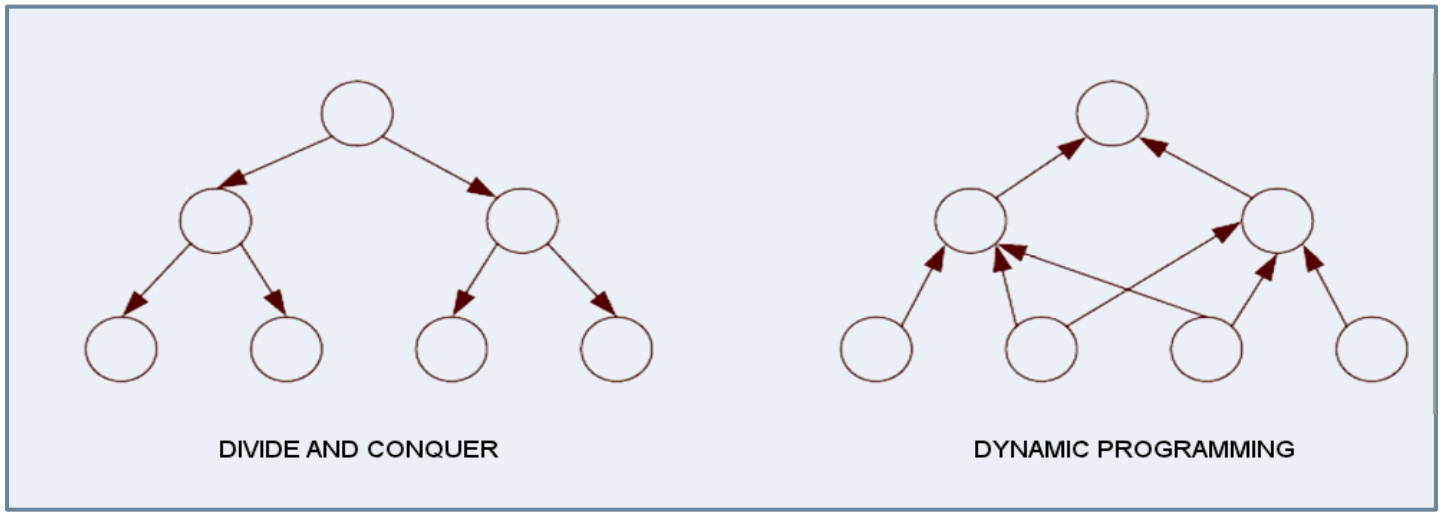
Διάταξη:



Σχέση: `Μεγάλο-ΥΠ = f(Μικρότερα-ΥΠ)`

Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Διαίρει-και-Βασίλευε vs ΔΠ !!!**

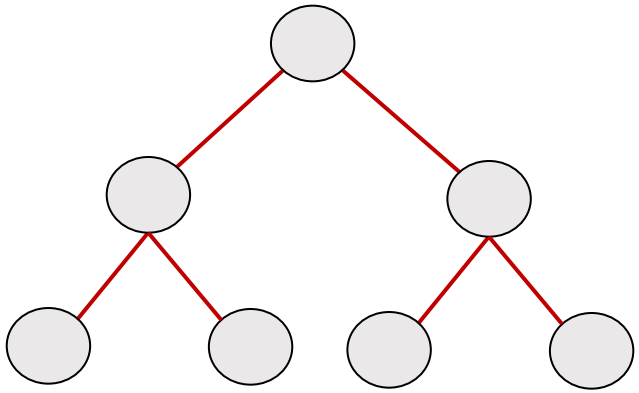


Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Διαίρει-και-Βασίλευε vs ΔΠ !!!**
  - ✓ Στην τεχνική Δ-και-Β ένα Π μεγέθους **n** εκφράζεται συναντήσει υποπροβλημάτων Π(1), Π(2), …, Π(k) που είναι ***σημαντικά μικρότερα***, για παράδειγμα **n/2**, και ***δεν επικαλύπτονται*** !!!
  - ✓ Λόγω αυτής της απότομης μείωσης του μεγέθους του Π, το δένδρο της αναδρομής έχει:

    O(logn) βάθος

    O(n) πλήθος κόμβων !!!



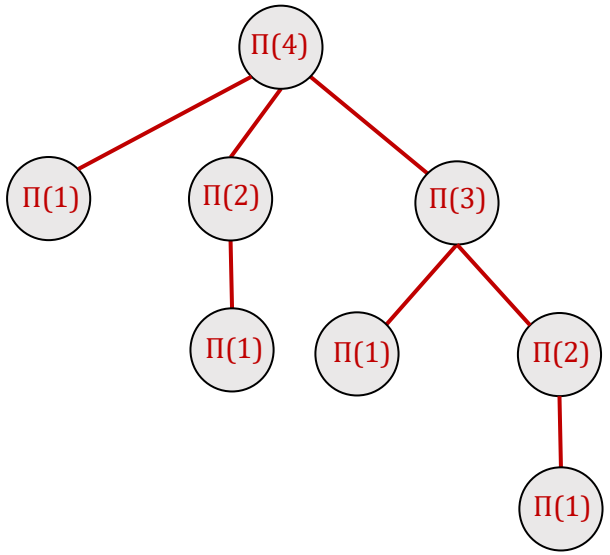
Χαρακτηριστικά Δυναμικού Προγραμματισμού

- **Διαίρει-και-Βασίλευε vs ΔΠ !!!**
  - ✓ Αντίθετα, στην τεχνική του ΔΠ τα υποπροβλήματα Π(1), Π(2), …, Π(k) είναι ***ελάχιστα μικρότερα***, για παράδειγμα το **Π(i)** βασίζεται στο **Π(i-1)**, και ***επικαλύπτονται*** !!!
  - ✓ Εδώ συνήθως, το δένδρο της αναδρομής έχει:

    $O(n)$ βάθος

    $O(c^n)$ πλήθος κόμβων, $c>1$

    Εκθετικό πλήθος κόμβων !!!



Χαρακτηριστικά Δυναμικού Προγραμματισμού

- *Ο ΔΠ είναι μια πολύ ισχυρή αλγοριθμική τεχνική !!!... με ευρύτατο πεδίο εφαρμογών !!!*

Συνοπτικά, στο ΔΠ:

- υπολογίζουμε ένα σύνολο υποπροβλημάτων του Π,
- λύνουμε κάθε υποπρόβλημα μία μόνο φορά,
- αποθηκεύουμε τη λύση του, και
- χρησιμοποιούμε αυτή, εάν χρειαστεί να το ξαναλύσουμε!!!

Αποφεύγουμε έτσι να λύνουμε ξανά-και-ξανά πολλές φορές τα ίδια-και-ίδια υποπροβλήματα !!!

Προβλήματα

Μέγιστες Αύξουσες Υποακολουθίες

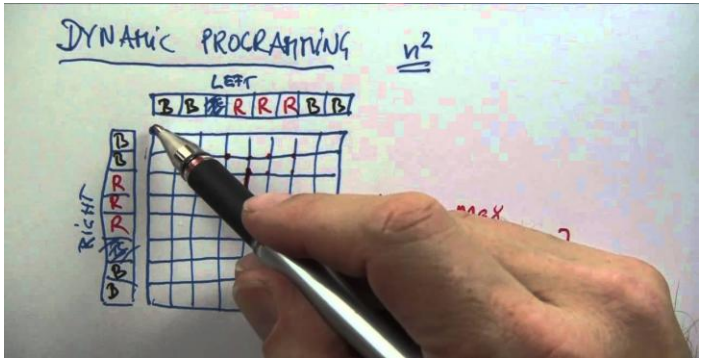
Διορθωτική Απόσταση

Πολλαπλασιασμός Ακολουθίας Πινάκων

Ελάχιστες Αξιόπιστες Διαδρομές

Πανζευκτικές Ελάχιστες Διαδρομές

Σακίδιο (knap-sack)



1 Μέγιστες Αύξουσες Υποακολουθίες

Πρόβλημα

Μας δίδεται μια ακολουθία n αριθμών

$$A = (a_1, a_2, \ldots, a_n)$$

και μας ζητείται να υπολογίσουμε μία (γνησίως) αύξουσα υπακολουθία της A με το μέγιστο μήκος.

Μια ακολουθία $A' = (a_{i1}, a_{i2}, \ldots, a_{ik})$ είναι γνησίως αύξουσα υπακολουθία της A εάν:

$$a_{i1} < a_{i2} < \ldots < a_{ik} \quad \text{και} \quad 1 \leq i_1 < i_2 < \ldots < i_k \leq n$$

1 Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

Μια μέγιστη γνησίως αύξουσα υπακολουθία της

$$A = (5, \mathbf{2}, 8, 6, \mathbf{3}, \mathbf{6}, \mathbf{9}, 7)$$

είναι η $A' = (2, 3, 6, 9)$ με μήκος 4.

Οι υπακολουθίες $B' = (2, 6, 6, 9)$ και $C' = (5, 6, 6, 7)$ της A δεν είναι ζητούμενες διότι δεν είναι γνησίως αύξουσες.

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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

Θεμελιώδη Ιδιότητα του ΔΠ !!!

Υπάρχει μια **διάταξη** των υποπροβλημάτων,

και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα **έχοντας τις λύσεις «μικροτέρων»** υποπροβλημάτων !!!

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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

Επομένως… Θεμελιώδες Κρίσιμο Ερώτημα !!!

Ποια είναι τα υποπροβλήματα ?…

που θα μας οδηγήσουν σε μια βέλτιστη λύση του προβλήματός μας !!!

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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

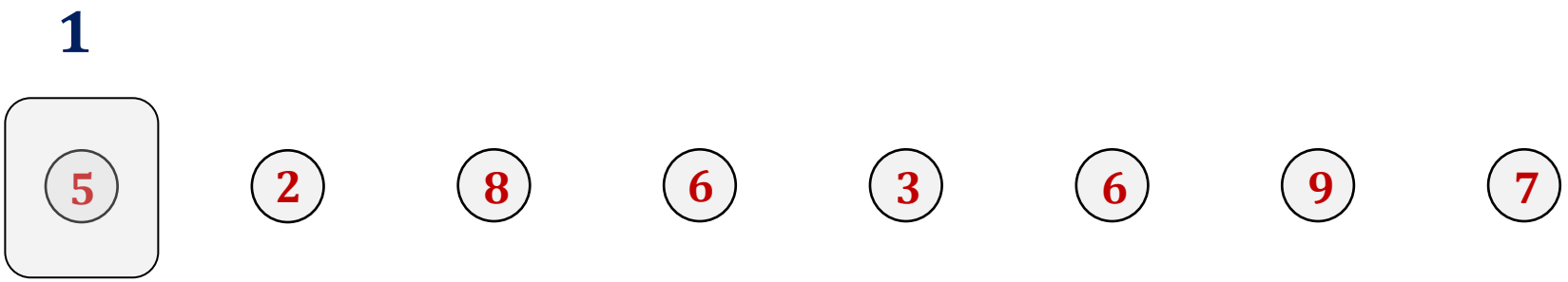
Μπορούμε να **«συρρικνώσουμε»** το αρχικό πρόβλημα με τον εξής τρόπο:

> Μπορούμε να εξετάσουμε τον υπολογισμό μέγιστων υπακολουθιών σε ακολουθίες **`A(i)`** που αποτελούνται από τα **`i`** πρώτα στοιχεία της **`A`**, για **`i ≤ n`**

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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

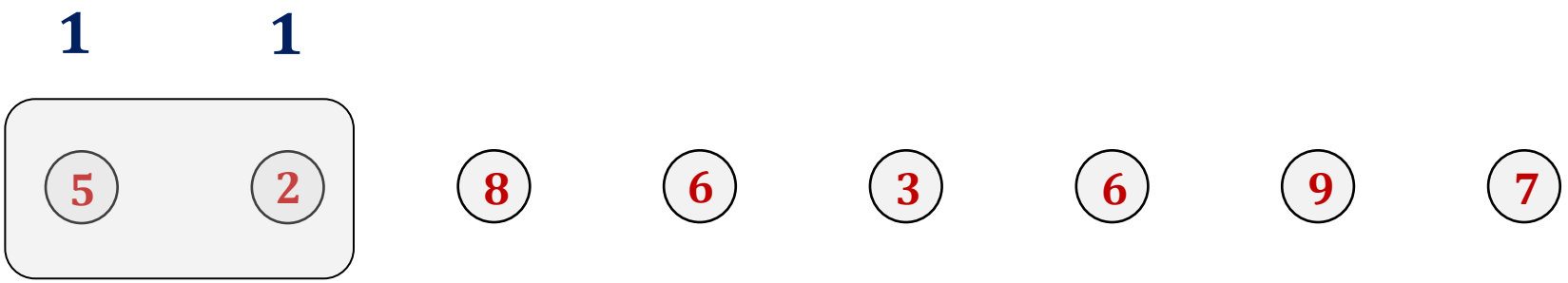


Ξεκινάμε λύνοντας το πρόβλημα στην ακολουθία **A(1)** = (5)

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



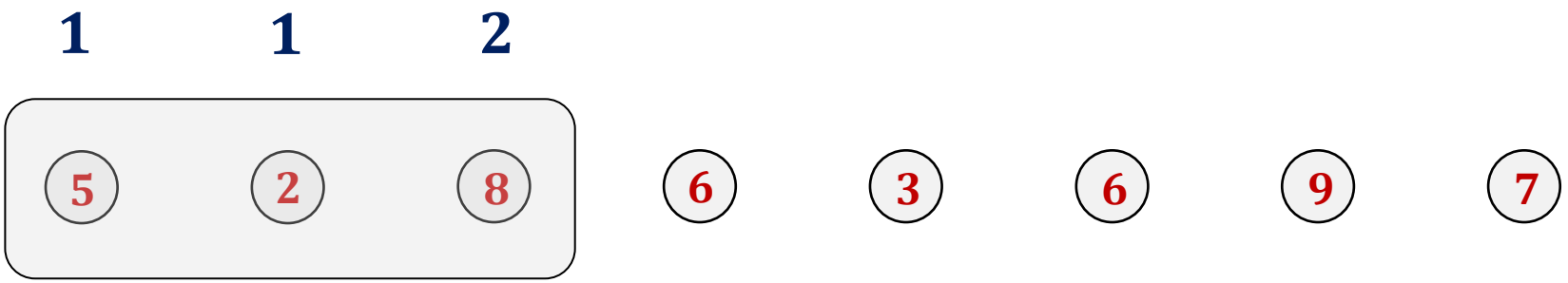
Ξεκινάμε λύνοντας το πρόβλημα στην ακολουθία **A(1) =** (5)

κατόπιν στην **A(2) =** (5, 2)

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



Ξεκινάμε λύνοντας το πρόβλημα στην ακολουθία **A(1) =** (5)

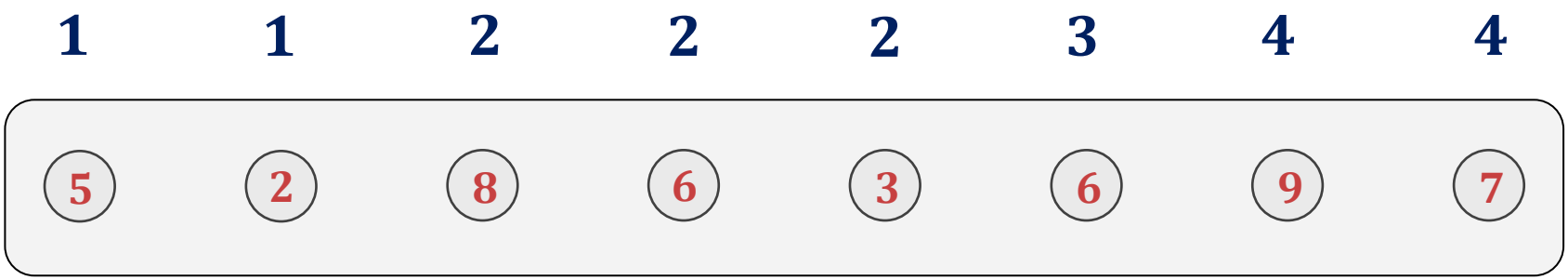
κατόπιν στην **A(2) =** (5, 2)

στην **A(3) =** (5, 2, 8)

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



Ξεκινάμε λύνοντας το πρόβλημα στην ακολουθία **A(1)** = (5)

κατόπιν στην **A(2)** = (5, 2)

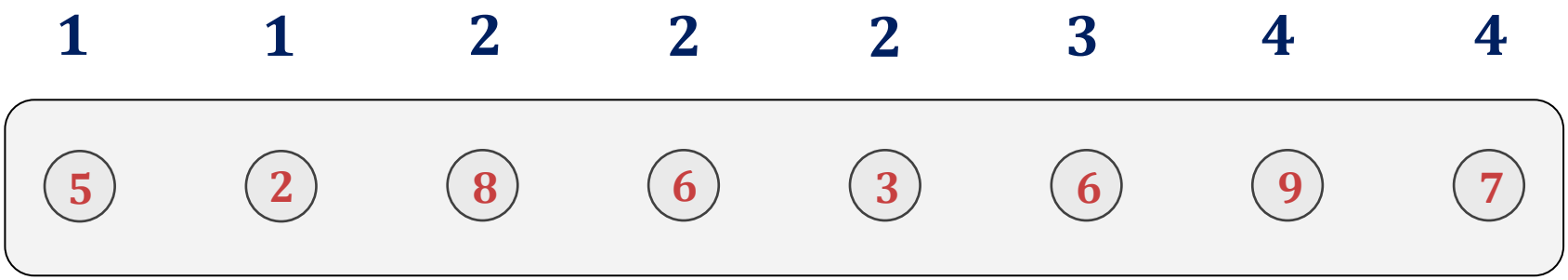
στην **A(3)** = (5, 2, 8)

και τέλος στην **A(8)** = (5, 2, 8, 6, 3, 6, 9, 7) !!! ⇒ Λύση : (2, 3, 6, 7)

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



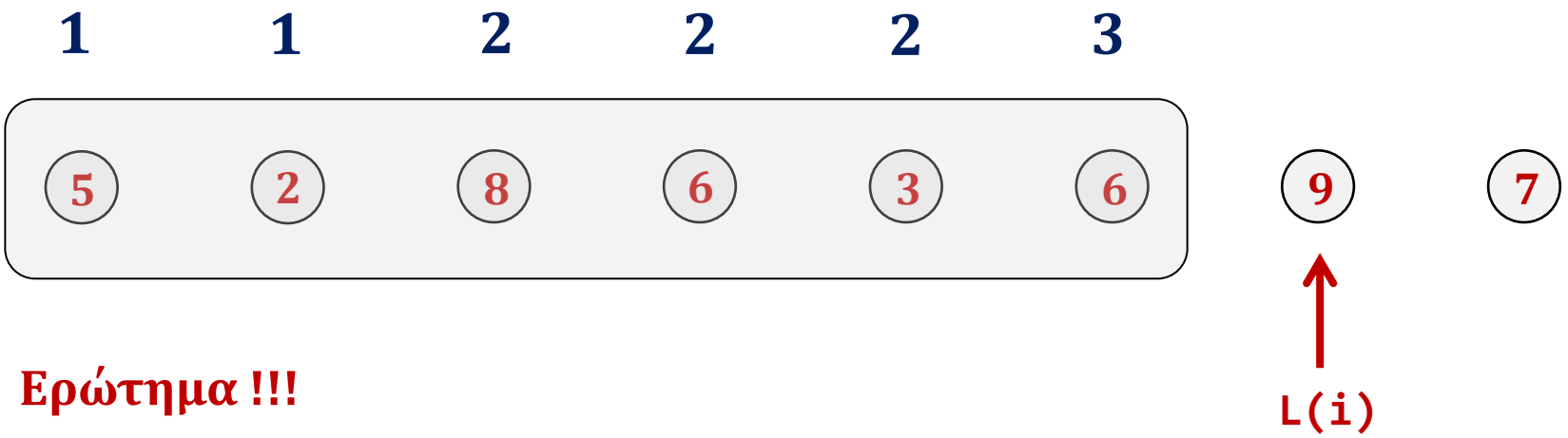
Ορίζουμε:

L(i) = μήκος της μέγιστης υπακολουθίας της ακολουθίας **A(i)**, **i ≤ n**

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



Ερώτημα !!!

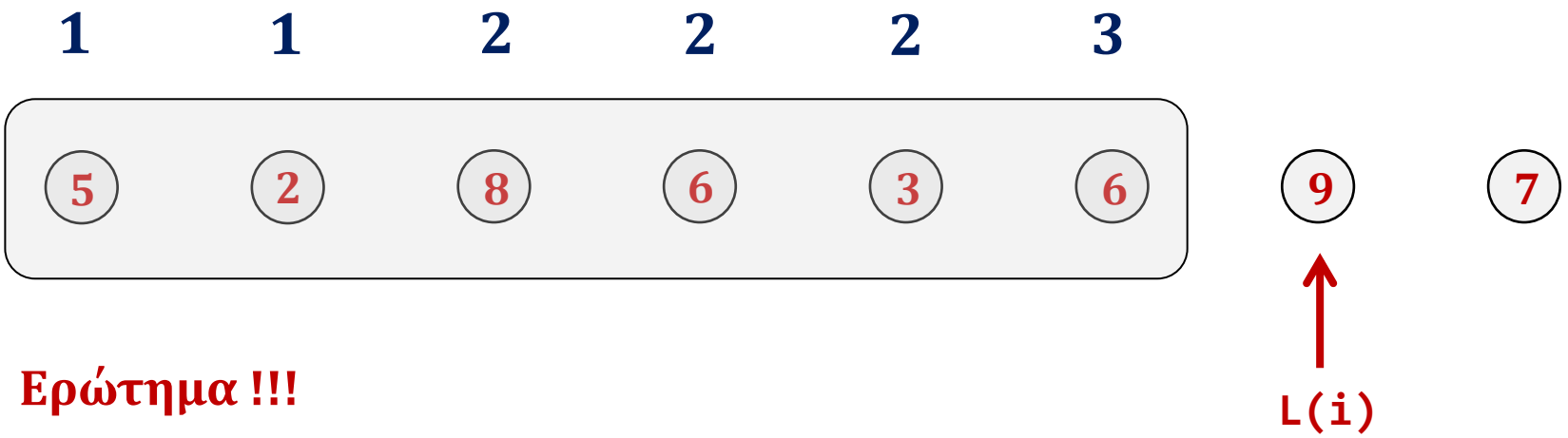
Πως βρίσκουμε την βέλτιστη λύση για το υποπρόβλημα **`A(i)`**

έχοντας τις βέλτιστες λύσεις των υποπροβλημάτων `A(1), A(2),…, A(i-1)`?

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



Ερώτημα !!!

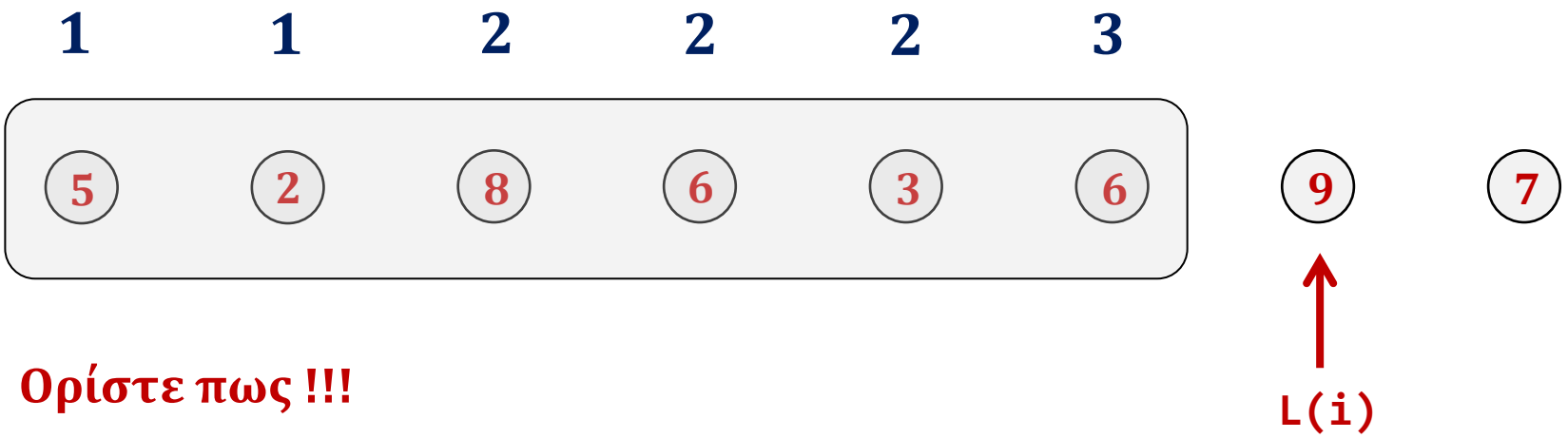
Είναι απλό !!!... Αρκεί να εκφράσουμε το υποπρόβλημα **`A(i)`**

συναρτήσει των μικρότερων υποπροβλημάτων `A(1), A(2),…, A(i-1)` `!!!`

Μέγιστες Αύξουσες Υποακολουθίες

- Θα χρησιμοποιήσουμε ΔΠ

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



Ορίστε πως !!!

$$L(i) = 1 + \max_{1 \le k \le i-1}\{L(k) : A[k] < A[i]\}$$

Μέγιστες Αύξουσες Υποακολουθίες

- **Αλγόριθμος**

```
Longest-Subsequence(A[1..n])
1. C(1) = 1
2. for i ← 2 to n
       for k ← 1 to i-1
           L[i] ← 1 + max{L[k] : A[k] < A[i]}
3. return L[n]
```

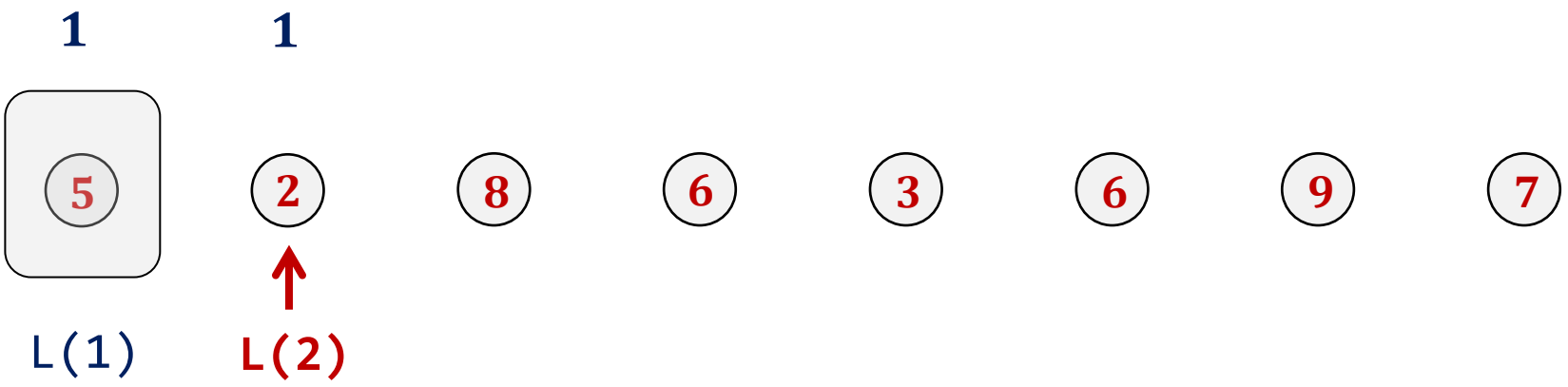
Συνολική Πολυπλοκότητα

$\mathbf{O(n^2)}$

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$$



$$L(2) \leftarrow 1 + \max\{\}$$

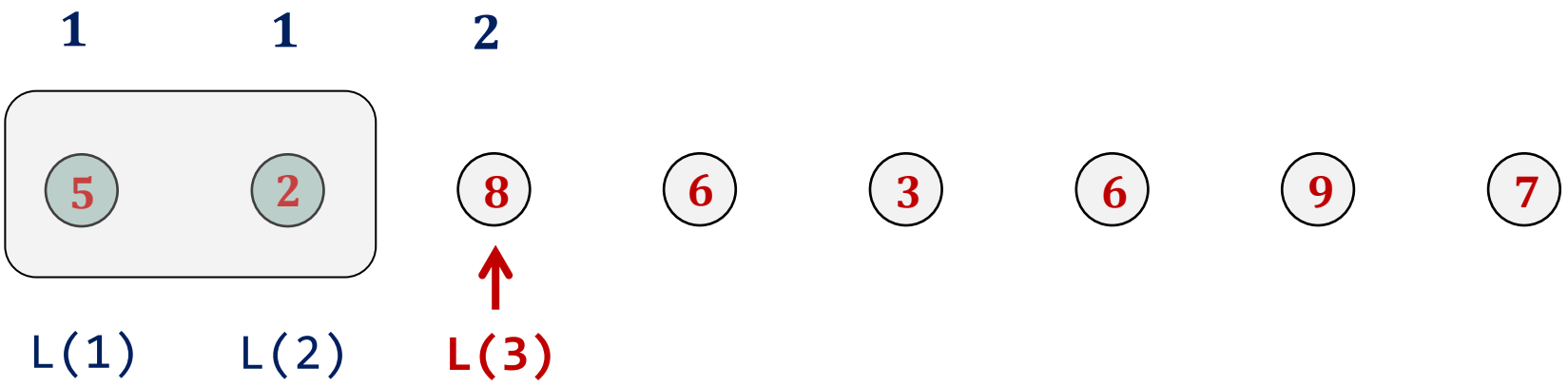
$$1 + 0 = 1$$

όπου, ως συνήθως, η σύμβασή μας είναι ότι η μέγιστη τιμή ενός κενού συνόλου είναι 0 !!!

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$$



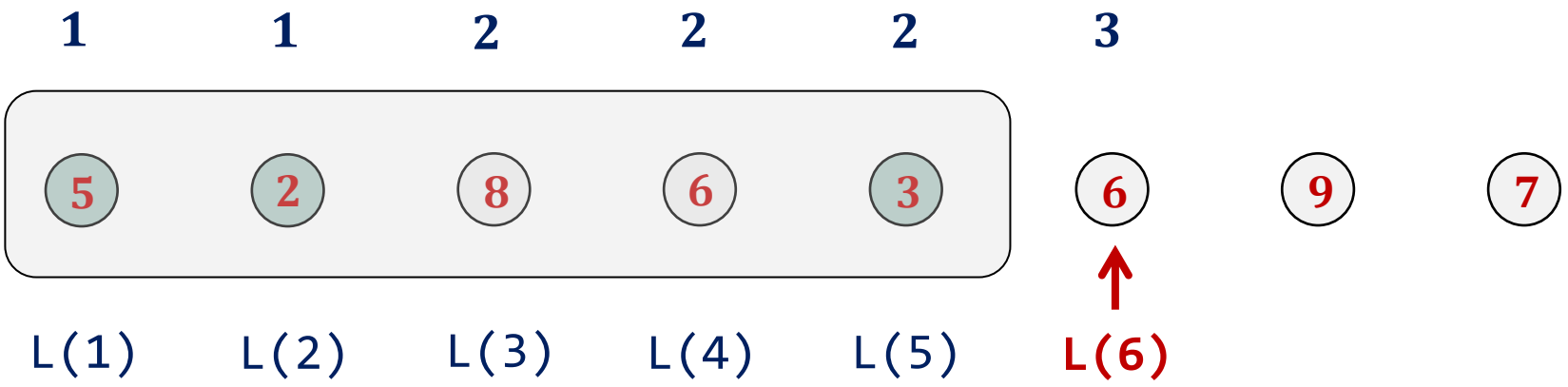
$$L(3) \leftarrow 1 + \max\{L(1), L(2)\}$$

$$1 + \max\{1, 1\} = 2$$

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$$



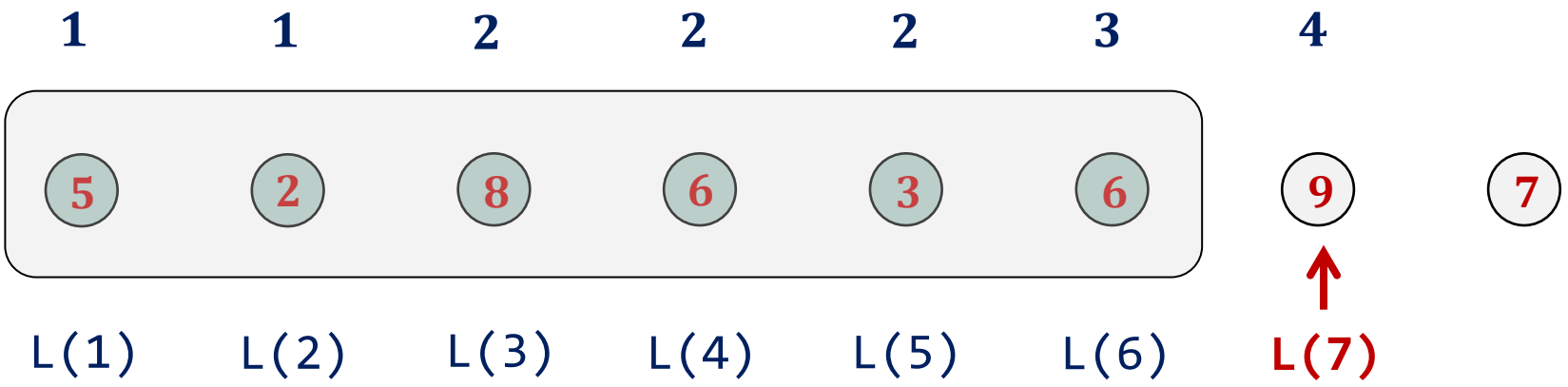
$$L(6) \leftarrow 1 + \max\{L(1), L(2), L(5)\}$$

$$1 + \max\{1, 1, 2\} = 3$$

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$$



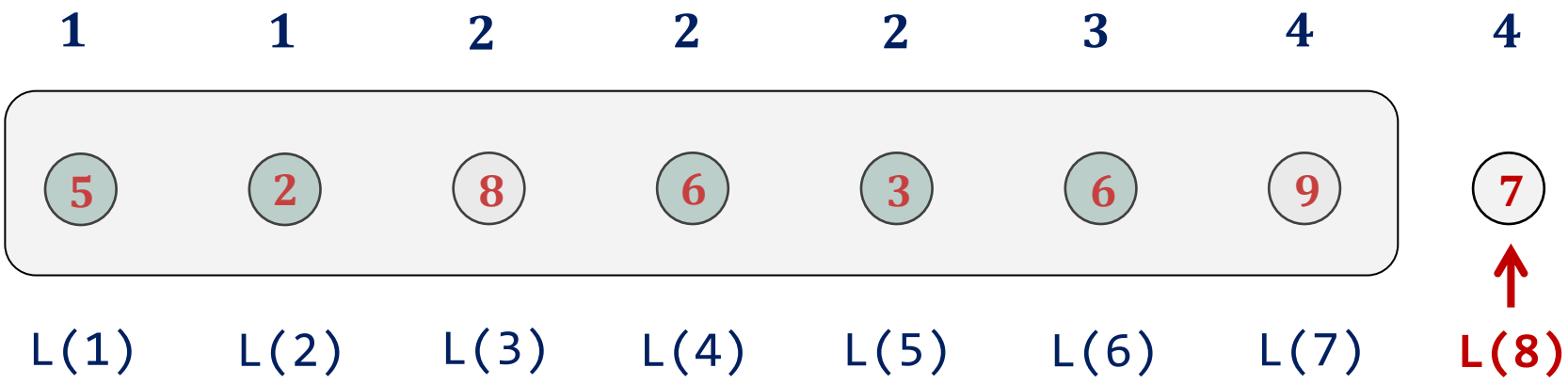
$$L(7) \leftarrow 1 + \max\{L(1), L(2), L(3), L(4), L(5), L(6)\}$$

$$1 + \max\{1, 1, 2, 2, 2, 3\} = \mathbf{4}$$

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$$



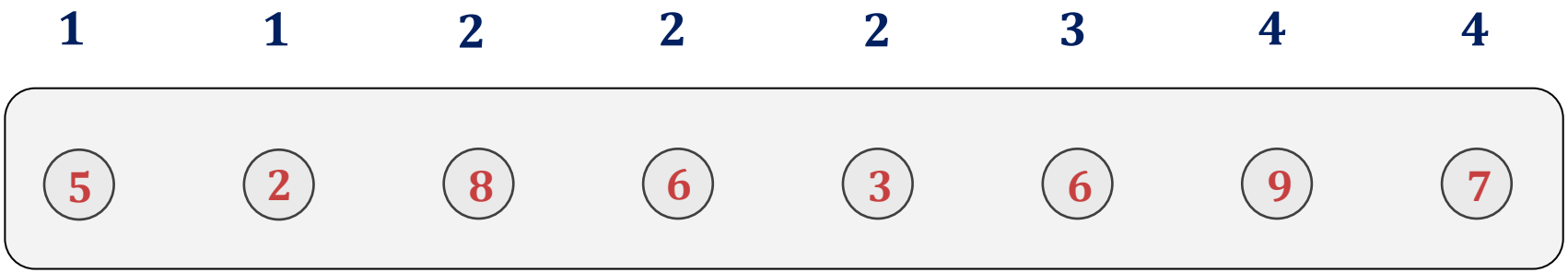
$$L(8) \leftarrow 1 + \max\{L(1), L(2), L(4), L(5), L(6)\}$$

$$1 + \max\{1, 1, 2, 2, 3\} = \mathbf{4}$$

Μέγιστες Αύξουσες Υποακολουθίες

Παράδειγμα

$L(i) \leftarrow 1 + \max\{L(k) : A[k] < A[i]\}$



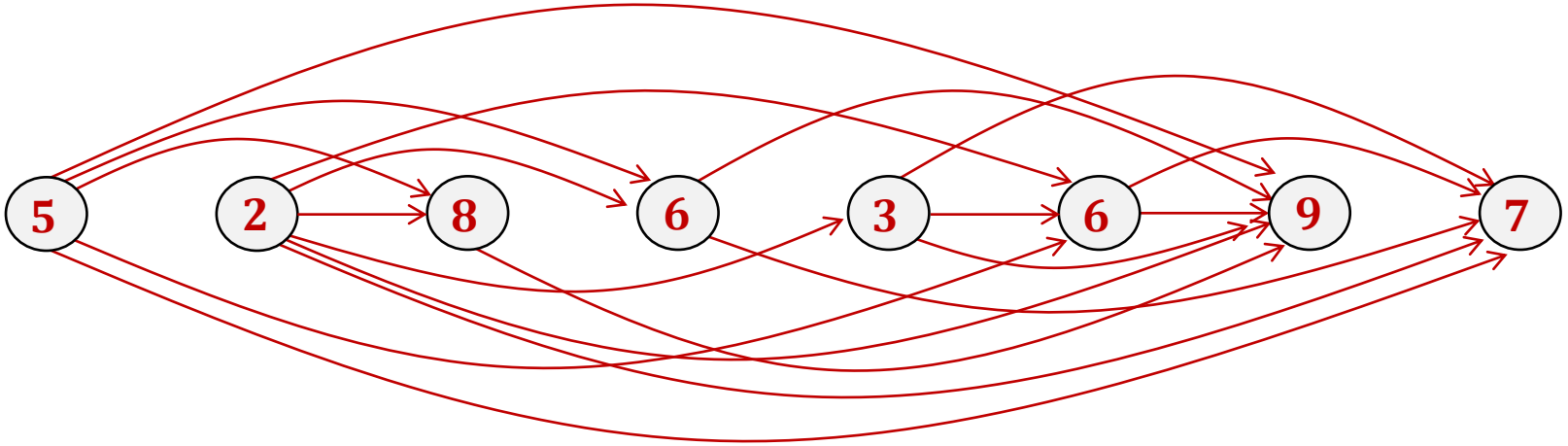
A = (5, 2, 8, 6, 3, 6, 9, 7)

Λύση !

(2, 3, 6, 7)

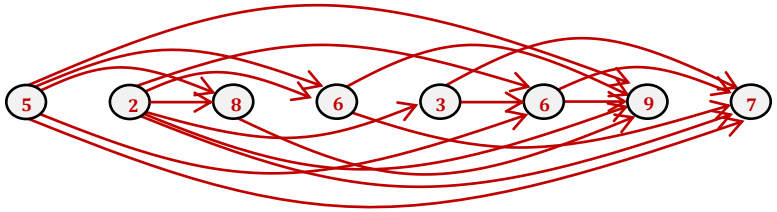
- Εναλλακτική Λύση

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



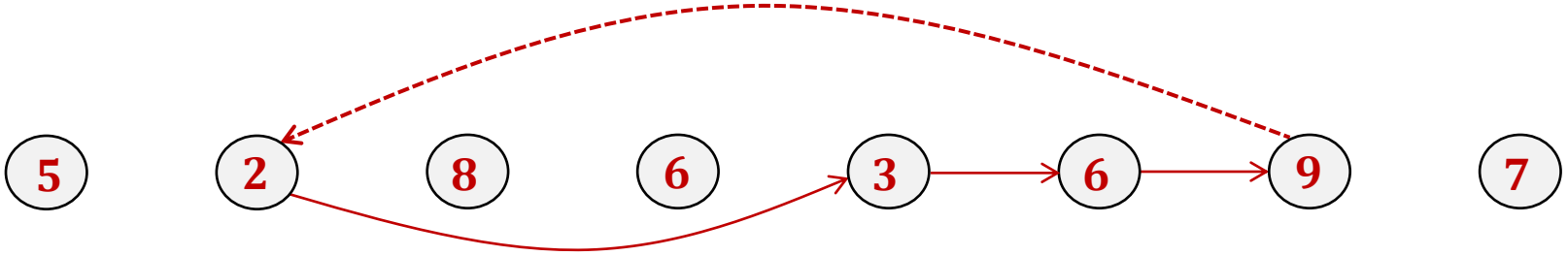
Δημιουργούμε ένα γράφημα **G** με **ΟΛΕΣ** τις **επιτρεπτές μεταβάσεις** !!!

- Εναλλακτική Λύση

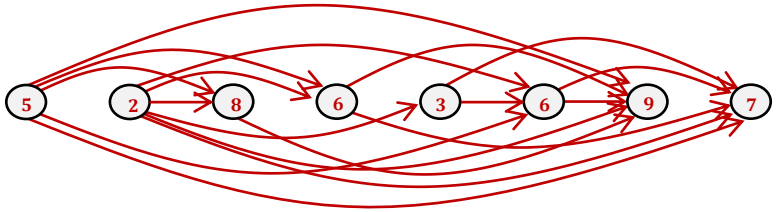


  - Παρατηρήστε ότι:

    (1) Το γράφημα G είναι **DAG** !!!



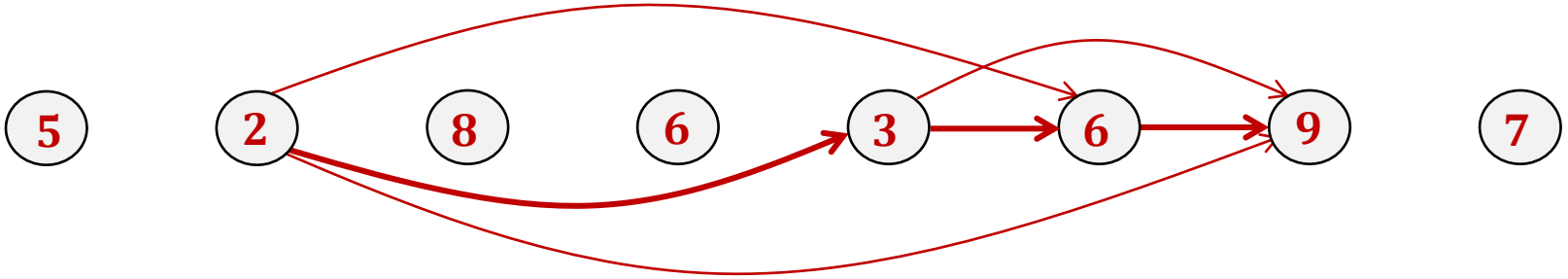
- Εναλλακτική Λύση



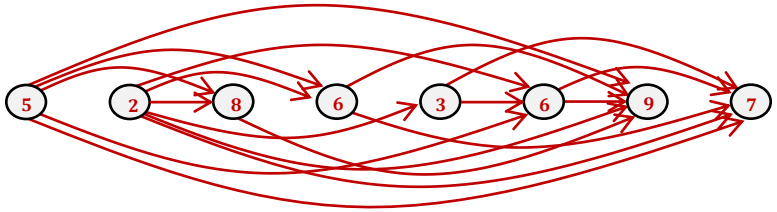
  - Παρατηρήστε ότι:

    (1) Το γράφημα G είναι **DAG** !!!

    (2) Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές αυτού του DAG** !!!



- Εναλλακτική Λύση



  - Παρατηρήστε ότι:

    (1) Το γράφημα G είναι **DAG** !!!

    (2) Υπάρχει 1-1 αντιστοιχία ανάμεσα στις **αύξουσες υπακολουθίες** και στις **διαδρομές αυτού του DAG** !!!

  - Επομένως, το πρόβλημά μας **ανάγεται** στον **υπολογισμό της μεγαλύτερης διαδρομής** στο γράφημα G !!!

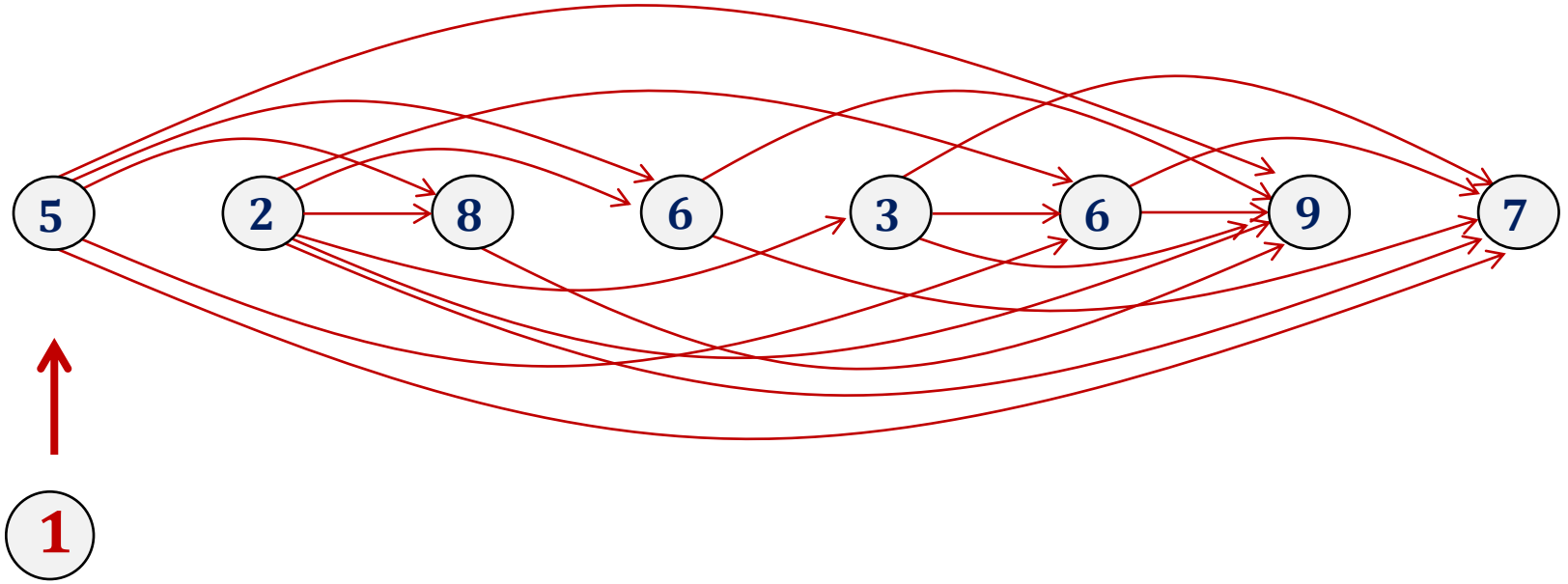
Υπολογισμός:

Μεγαλύτερης διαδρομής σε DAG !!!

Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

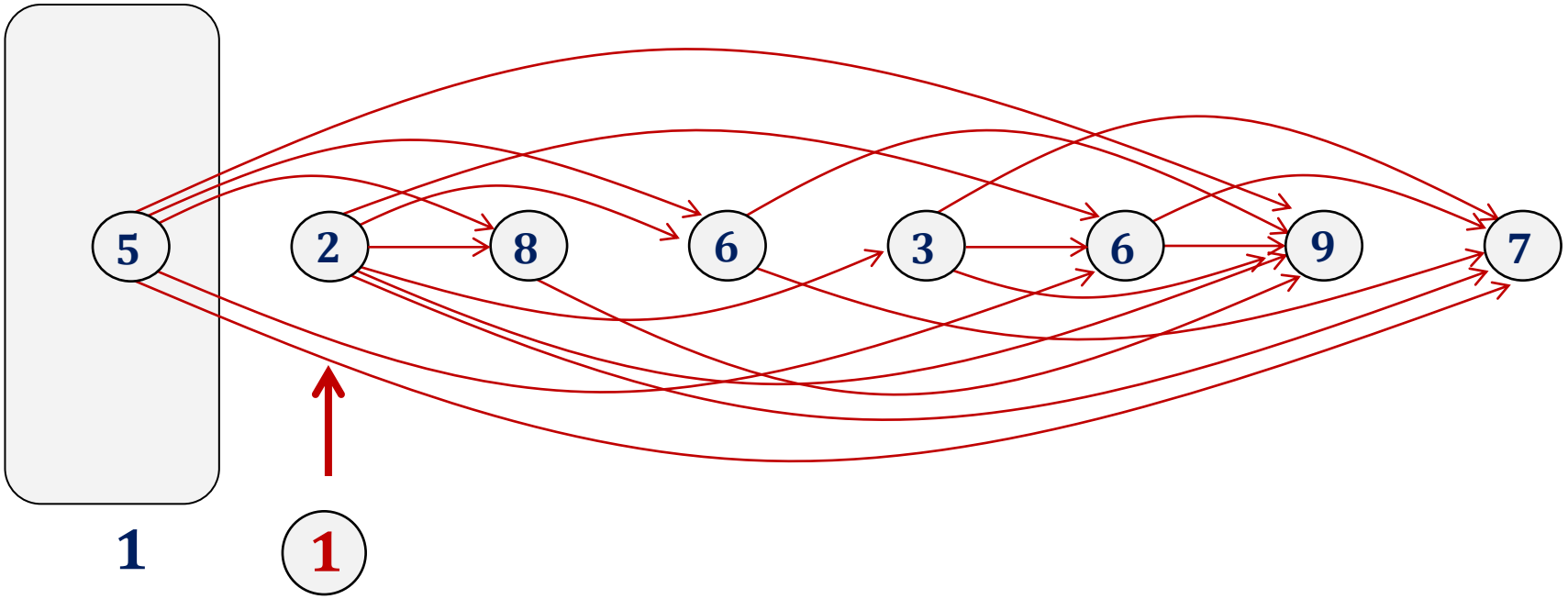
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

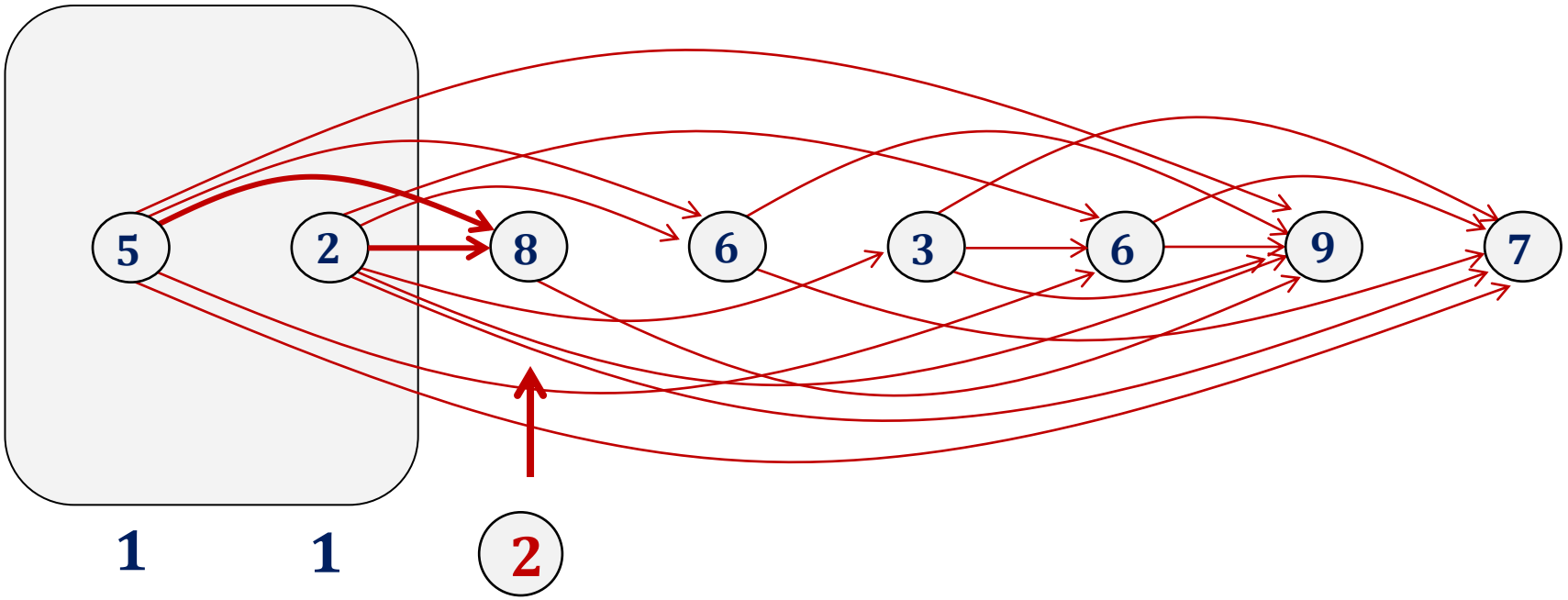
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

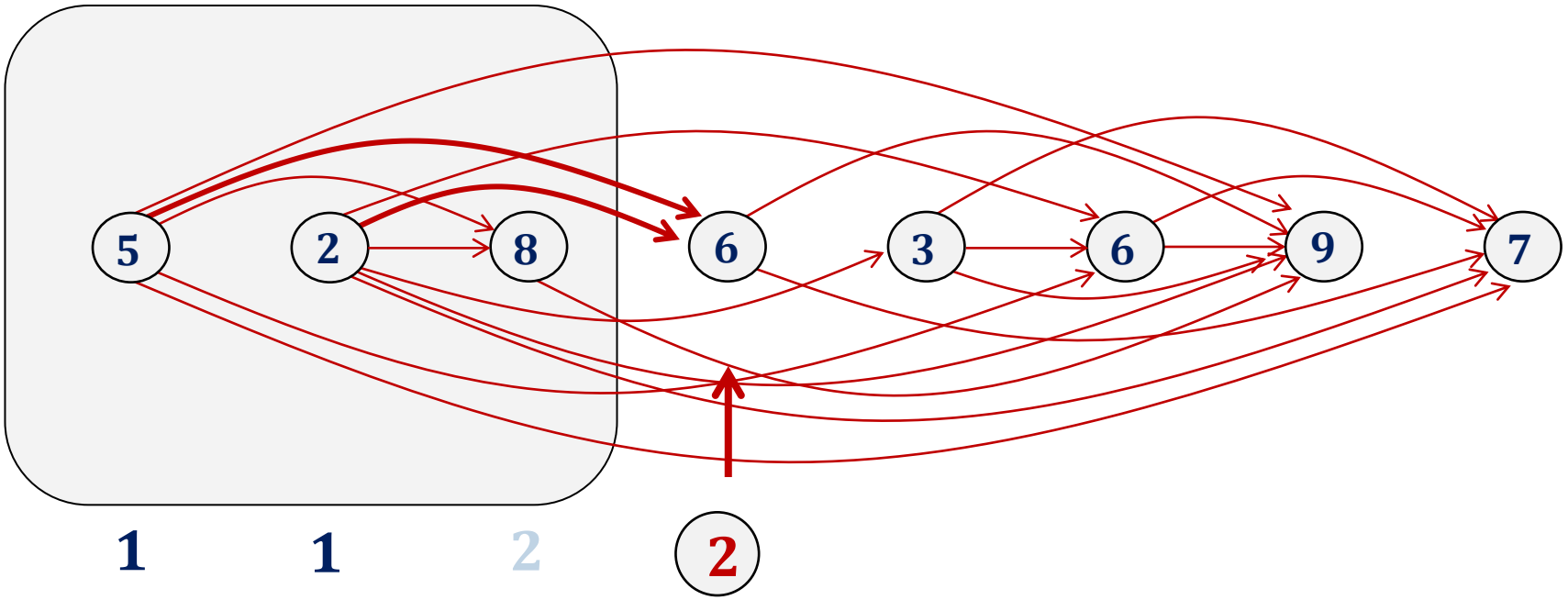
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

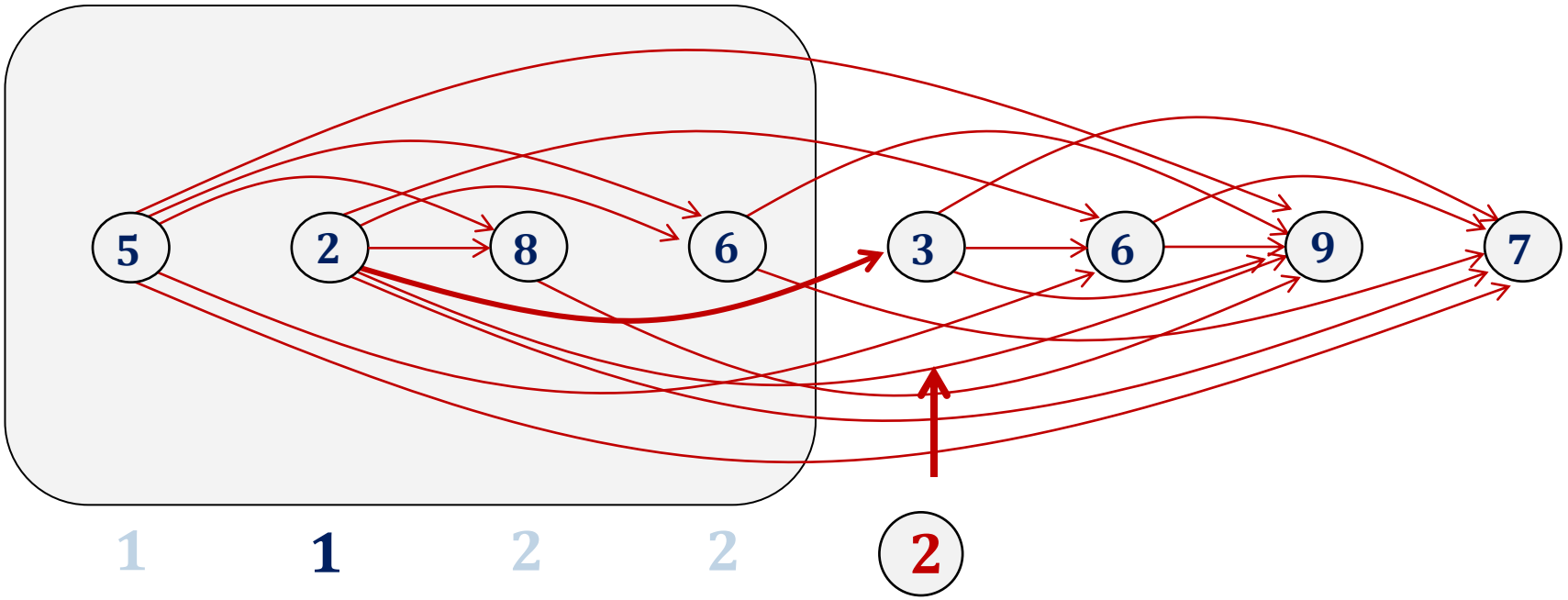
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

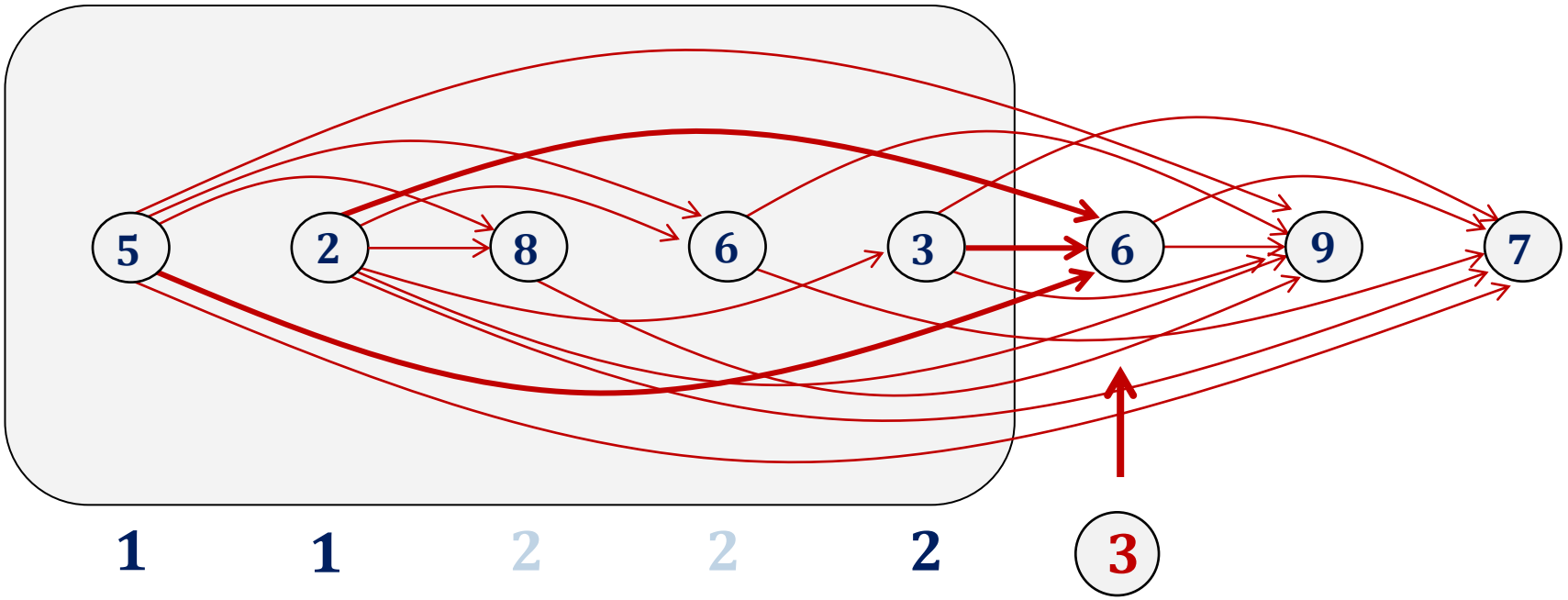
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

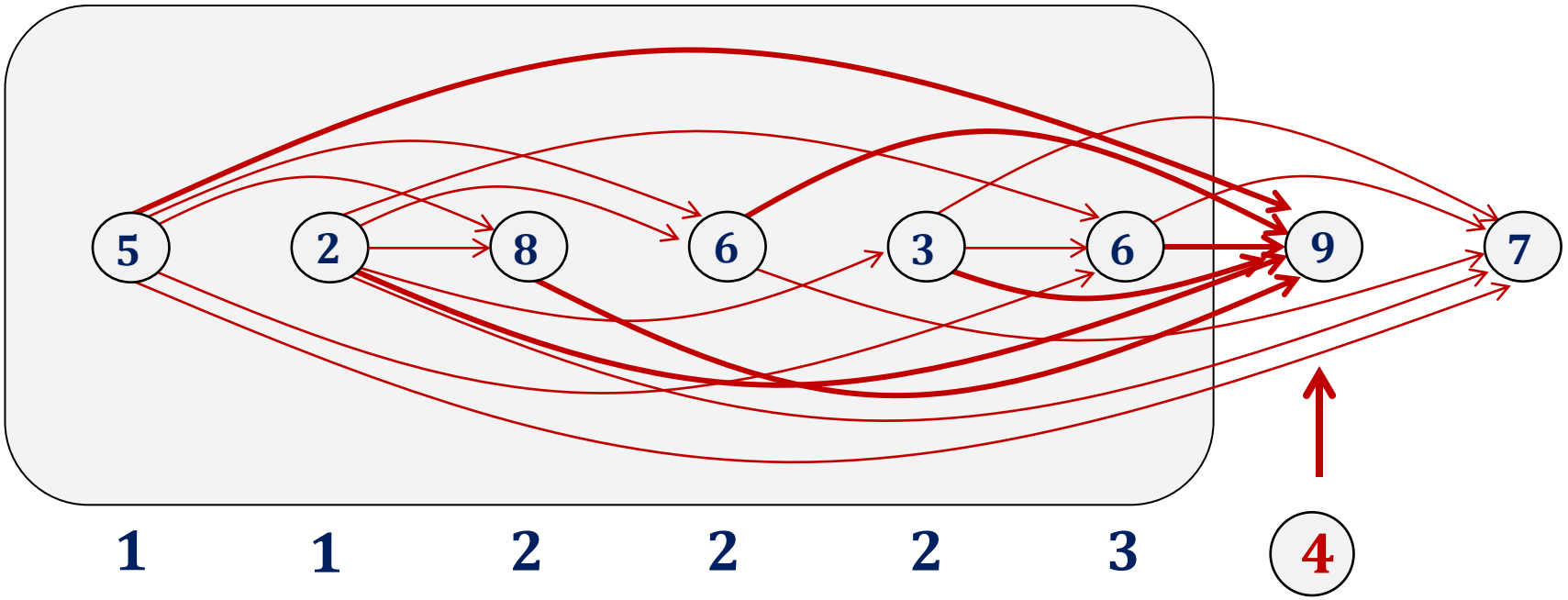
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

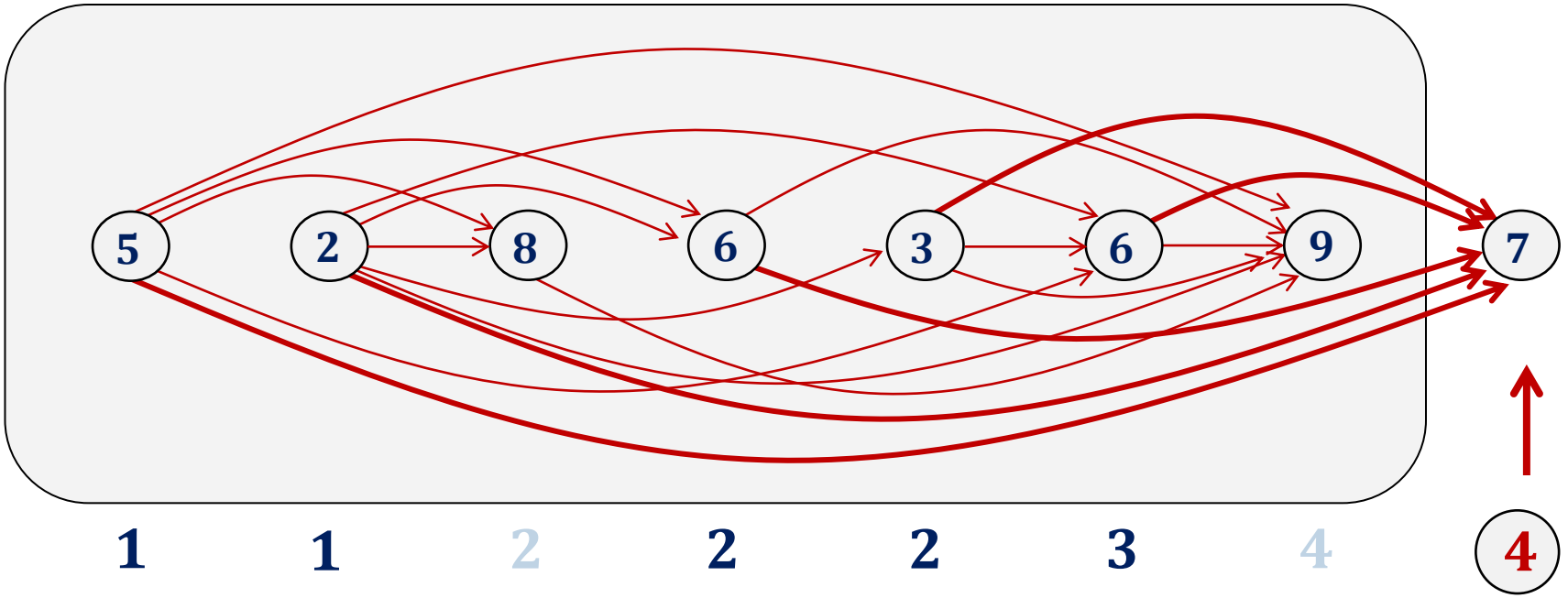
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

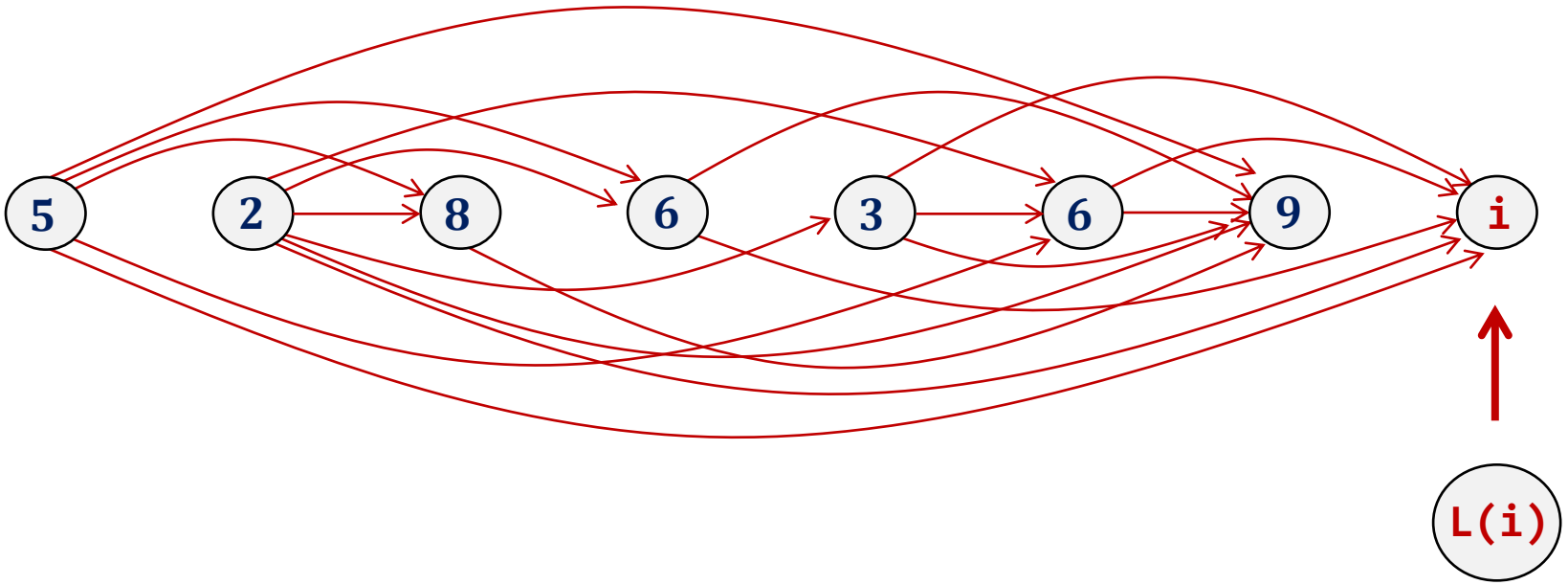
Μέγιστη Διαδρομή σε DAG



Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

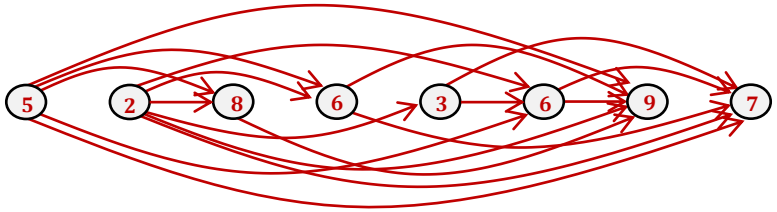
Μέγιστη Διαδρομή σε DAG



$$L(i) = 1 + \max\{L(k) : (k,i) \in E(G)\}$$

Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση



  - Ορίστε ο αλγόριθμος:

1. για i = 1, 2, …, n:

$$L(i) = 1 + \max\{L(k) : (k,i) \in E(G)\}$$

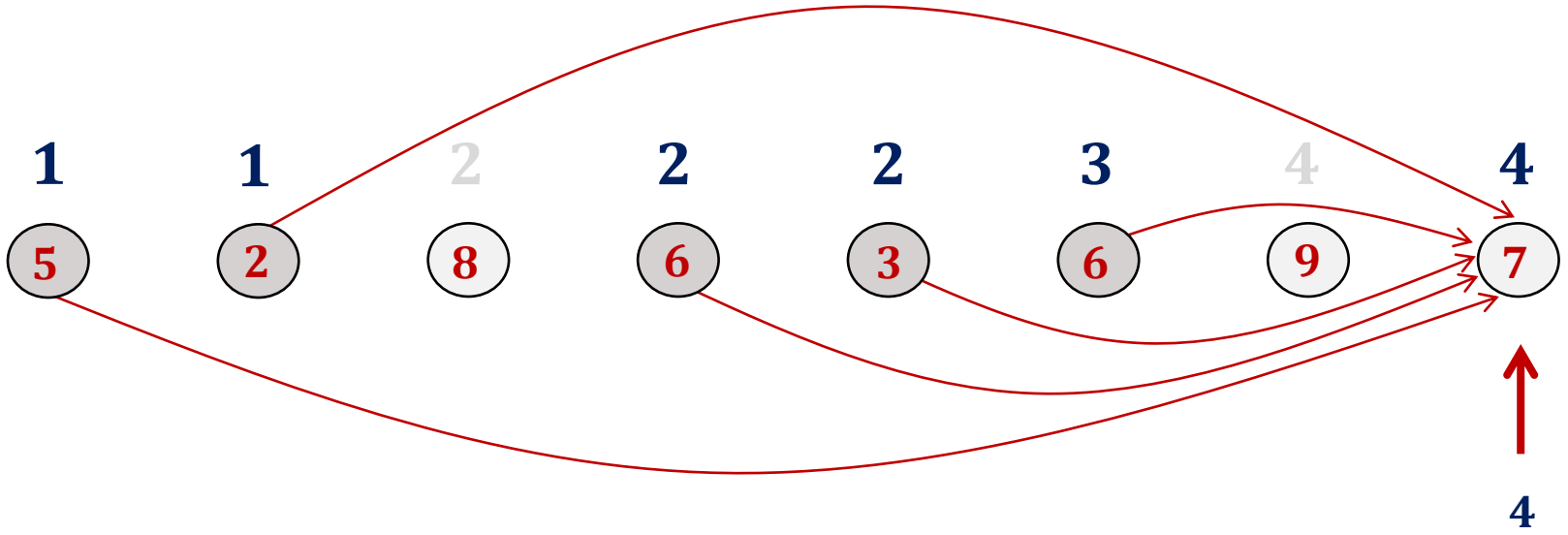
2. return $\max_{1 \le i \le n}\{L(i)\}$

L(i) = είναι το μήκος της μεγαλύτερης διαδρομής ή, ισοδύναμα, της μέγιστης αύξουσας υπακολουθίας,

που τελειώνει στον κόμβο i = 1, 2, …, n

- Εναλλακτική Λύση

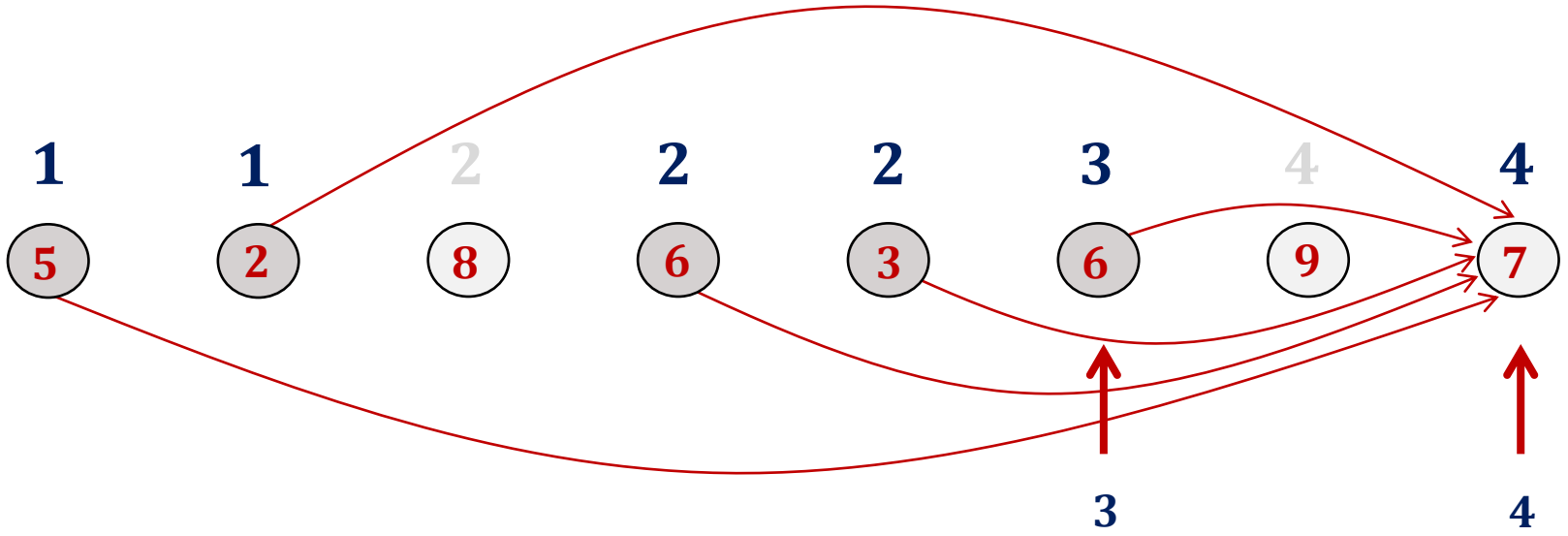
A = (5, 2, 8, 6, 3, 6, 9, 7)



Η μέγιστη αύξουσα υπακολουθία έχει μήκος 4

- Εναλλακτική Λύση

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

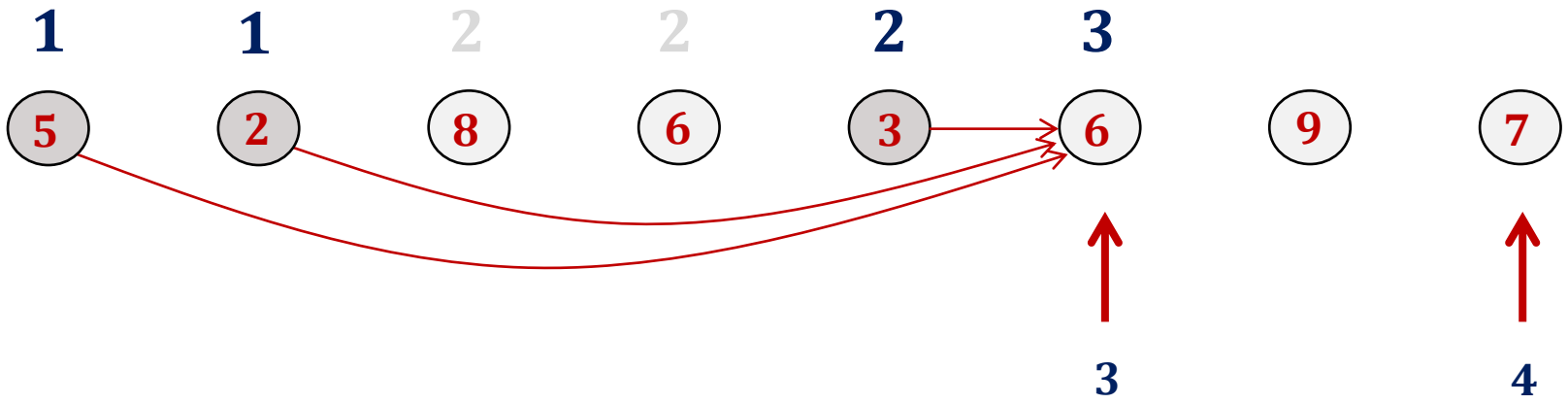


Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

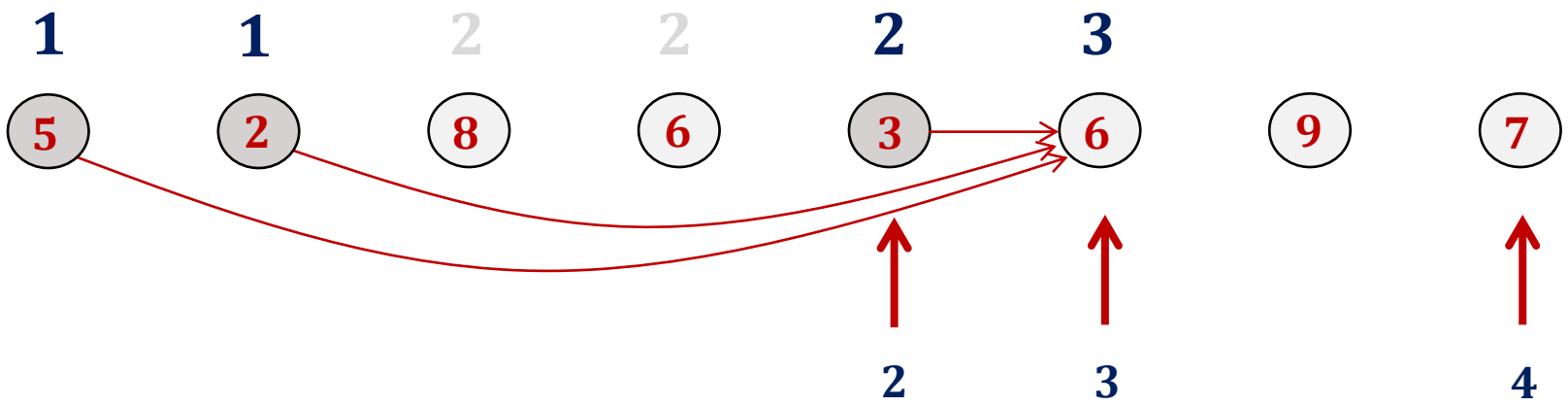
A = (5, 2, 8, 6, 3, 6, 9, 7)



Ποια είναι η υπακολουθία ?

- Εναλλακτική Λύση

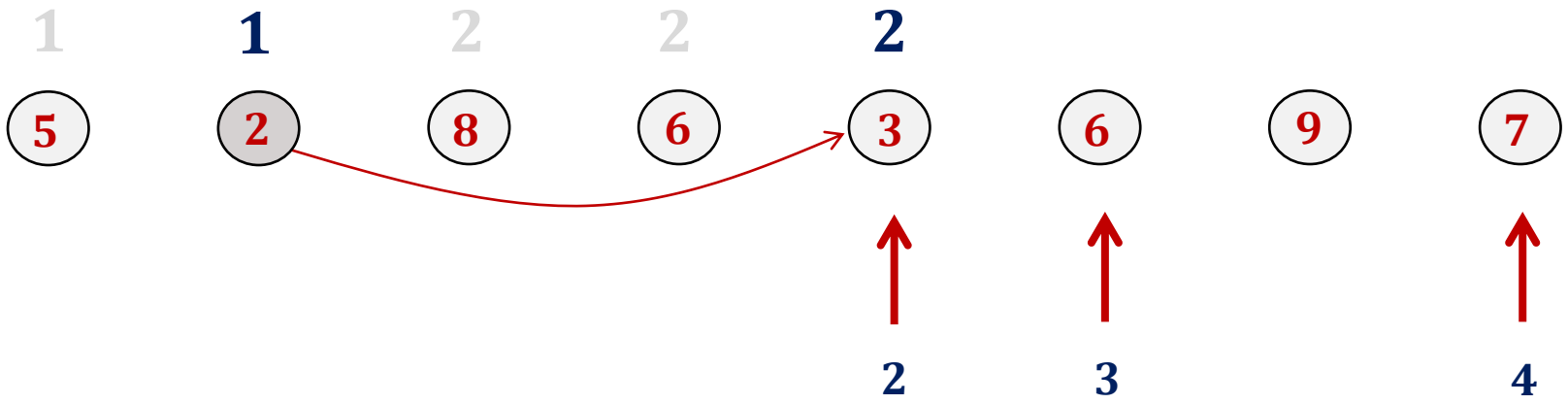
A = (5, 2, 8, 6, 3, 6, 9, 7)



Ποια είναι η υπακολουθία ?

- Εναλλακτική Λύση

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

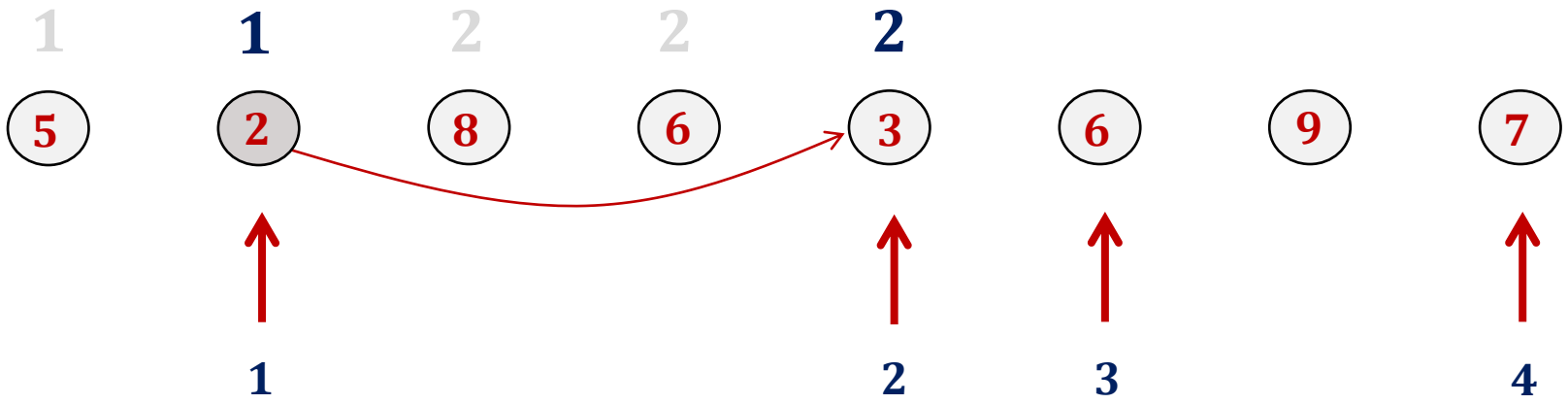


Ποια είναι η υπακολουθία ?

Μέγιστες Αύξουσες Υποακολουθίες

- Εναλλακτική Λύση

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

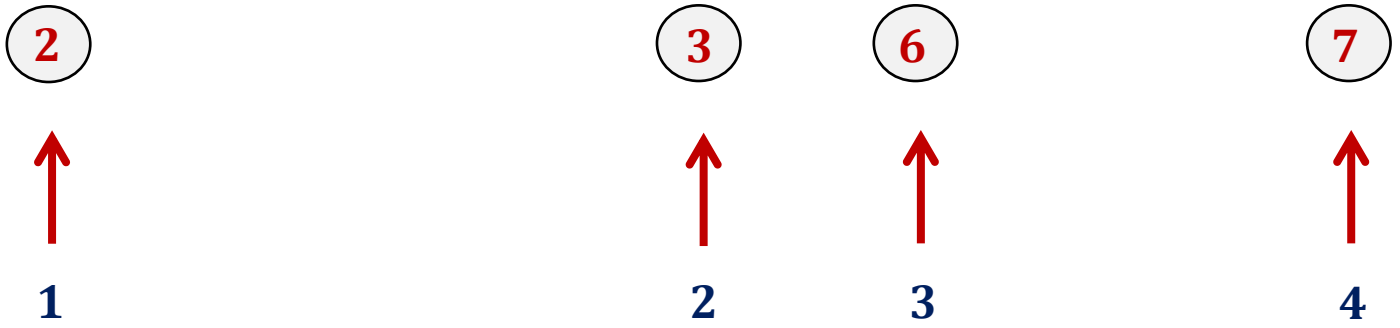


Ποια είναι η υπακολουθία ?

- Εναλλακτική Λύση

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

  - **Λύση !!!...**
    η υπακολουθία: **(2, 3, 6, 7)**



Ποια είναι η υπακολουθία ?

Γιατί είναι ΔΠ ?

- για i = 1, 2, …, n:

$$L(i) = 1 + \max\{L(k) : (k,i) \in E(G)\}$$

- Ορίσαμε, υποπροβλήματα: $\{L(i): 1 \le i \le n\}$

  που έχουν τη Θεμελιώδης Ιδιότητα του ΔΠ !!!

Υπάρχει μια **διάταξη** των υποπροβλημάτων,
και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα
έχοντας τις λύσεις «μικροτέρων» υποπροβλημάτων !!!

- **Πολυπλοκότητα**
  - για i = 1, 2, …, n:

    $L(i) = 1 + \max\{L(k) : (k,i) \in E(G)\}$

  - Το αρχικό μας πρόβλημα λύνεται με ένα «πέρασμα» των $n = |A|$ κόμβων
  - Ο υπολογισμός του $L(i)$ απαιτεί χρόνο ανάλογο του βαθμού εισόδου του κόμβου i του G

    Συνολικά, χρόνο τάξης $|E(G)| \Rightarrow O(n^2)$

2 Διορθωτική Απόσταση

Κίνητρο

Όταν ένας ελεγκτής ορθογραφίας συναντά μια πιθανή ανορθόγραφη λέξη

$$\alpha_1\alpha_2 \ldots \alpha_n$$

εξετάζει το λεξικό του για άλλες λέξεις που είναι «κοντά» στην ανορθόγραφη!!

Λεξικό

$\beta_1\beta_2 \ldots \beta_m$
$\gamma_1\gamma_2 \ldots \gamma_k$
⋮
$\omega_1\omega_2 \ldots \omega_p$

Ποια είναι η κατάλληλη έννοια της "εγγύτητας" δύο λέξεων ή συμβολοσειρών ?

2 Διορθωτική Απόσταση

Κίνητρο

Ένα μέτρο της "εγγύτητας" ανάμεσα σε δύο λέξεις ή συμβολοσειρές α και β, είναι οι **διορθώσεις** που πρέπει να κάνουμε, έτσι ώστε $\alpha \rightarrow \beta$ ή $\beta \rightarrow \alpha$

Ποιες διορθώσεις μπορούμε να κάνουμε ?

- `Εισαγωγή συμβόλου`
- `Διαγραφή συμβόλου`
- `Αντικατάσταση συμβόλου`

} 3 πράξεις σε συμβολοσειρές

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων :

S U N N Y

S N O W Y

S N O W Y → S U N N Y

S U N N Y → S N O W Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

S	U	N	N	-	Y	
S	-	N	O	W	Y	

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

S	U	N	N	-	Y	
S	-	N	O	W	Y	

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

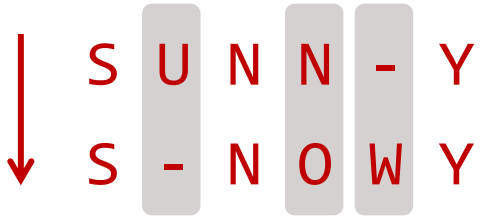
Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων



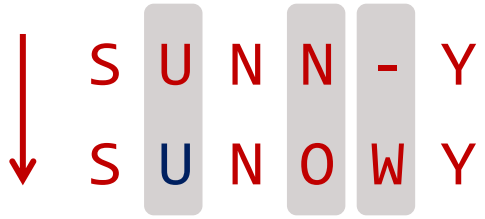
S N O W Y → S U N N Y

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων



S N O W Y → S U N N Y

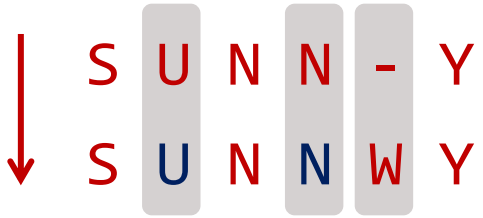
Εισαγωγή U

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων



S N O W Y → S U N N Y

Εισαγωγή U

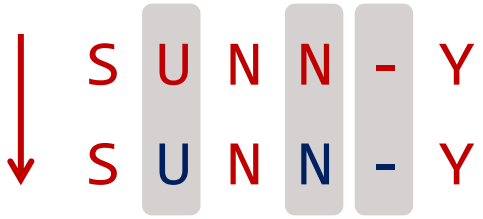
Αντικατάσταση του O σε N

2 Διορθωτική Απόσταση

Παράδειγμα

Έστω οι δύο λέξεις SUNNY και SNOWY.

Μπορούμε να πάρουμε την μία από την άλλη με διαφορετικά σύνολα αλλαγών/πράξεων



S N O W Y → S U N N Y

Εισαγωγή U
Αντικατάσταση του O σε N
Διαγραφή του W

2 Διορθωτική Απόσταση

Ορίζουμε ως **κόστος διόρθωσης** δύο συμβολοσειρών α και β, το πλήθος των πράξεων που απαιτούνται για να πάρουμε από την μία στην άλλη !!!

S	U	N	N	-	Y
S	-	N	O	W	Y

Κόστος **3**

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

Κόστος **5**

2 Διορθωτική Απόσταση

Το **ελάχιστο κόστος διόρθωσης** δύο συμβολοσειρών ονομάζεται **διορθωτική απόσταση (edit distance)** αυτών.

S	U	N	N	-	Y	
S	-	N	O	W	Y	

Κόστος **3**

S	U	N	-	-	N	Y
-	S	N	O	W	-	Y

Κόστος **5**

2 Διορθωτική Απόσταση

Το **ελάχιστο κόστος διόρθωσης** δύο συμβολοσειρών ονομάζεται **διορθωτική απόσταση (edit distance)** αυτών.

S	U	N	N	-	Y
S	-	N	O	W	Y

Κόστος **3**

E(SUNNY, SNOWY) = **3**

2 Διορθωτική Απόσταση

Πρόβλημα

Μας δίνονται δύο συμβολοσειρές α και β με n και m σύμβολα, αντίστοιχα,

$\alpha_1, \alpha_2, \ldots, \alpha_n$ και $\beta_1, \beta_2, \ldots, \beta_m$

και μας ζητείται να υπολογίσουμε την διορθωτική απόσταση αυτών.

$E(\alpha, \beta)$

- Στο ΔΠ το πιο κρίσιμο ερώτημα είναι το εξής:

Ποια είναι τα υποπροβλήματα ?

- Όταν αυτά επιλέγονται ώστε να έχουν την

Θεμελιώδη Ιδιότητα του ΔΠ !!!

Υπάρχει μια **διάταξη** των υποπροβλημάτων,

και μια **σχέση** που δείχνει **πως θα λυθεί** ένα υποπρόβλημα **έχοντας τις λύσεις «μικροτέρων»** υποπροβλημάτων !!!

η διατύπωση ενός αλγόριθμου είναι εύκολη υπόθεση !!!

Τότε, επαναληπτικά επιλύουμε τα υποπροβλήματα κατά σειρά αυξανόμενου μεγέθους !

Διορθωτική Απόσταση

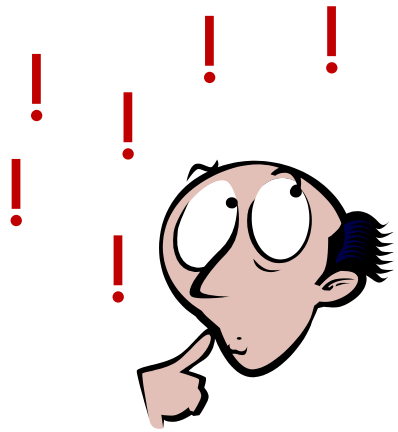
- Ποιο είναι ένα **καλό** υποπρόβλημα ?

  Θέλουμε την E(α,β) ανάμεσα στις δύο συμβολοσειρές

  $\alpha[1..n]=(\alpha_1, \alpha_2, \ldots, \alpha_n)$ και $\beta[1..m]= (\beta_1, \beta_2, \ldots, \beta_m)$

- Είναι **αυτό** που **«προχωράει»** τη λύση του συνολικού προβλήματος !!!

Πρέπει
να επινοήσουμε κάτι έξυπνο !



Τι θα λέγατε να εξετάσουμε την **E()** σε προθέματα (prefix) των **α** και **β** !

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

$$\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i) \quad \text{και} \quad \beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$$

και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

Παράδειγμα:

E X P O N E N T I A L α[1..7]

P O L Y N O M I A L β[1..5]

Το υποπρόβλημα: E(7,5)

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

  Παράδειγμα:

  E X P O N E N T I A L α[1..4]

  P O L Y N O M I A L β[1..2]

  Το υποπρόβλημα: E(4,2)

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

  $\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i)$ και $\beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$

  και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

  Παράδειγμα:

  E X P O N E N T I A L α[1..1]

  P O L Y N O M I A L β[1..5]

  Το υποπρόβλημα: E(1,5)

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

$$\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i) \quad \text{και} \quad \beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$$

και ας συμβολίσουμε E(i,j) την διορθωτική απόσταση αυτού του υποπροβλήματος

Παράδειγμα:

E	X	P	O	N	E	N	T	I	A	L

α[1..11]

P	O	L	Y	N	O	M	I	A	L

β[1..10]

ΤΕΛΙΚΟΣ ΣΤΟΧΟΣ: **E(11, 10)**

- Ας πάρουμε δύο προθέματα (prefix) των α και β, μήκους $i \le n$ και $j \le m$:

$$\alpha[1..i]=(\alpha_1, \alpha_2, \ldots, \alpha_i) \quad \text{και} \quad \beta[1..j]= (\beta_1, \beta_2, \ldots, \beta_j)$$

και ας συμβολίσουμε $E(i,j)$ την διορθωτική απόσταση αυτού του υποπροβλήματος

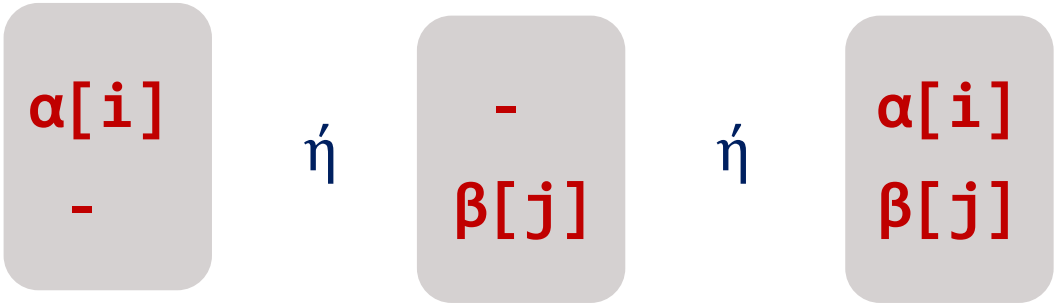
Λύση:

E	X	P	O	N	E	N	-	T	I	A	L
-	-	P	O	L	Y	N	O	M	I	A	L

ΤΕΛΙΚΟΣ ΣΤΟΧΟΣ: **$E(11, 10) = 6$**

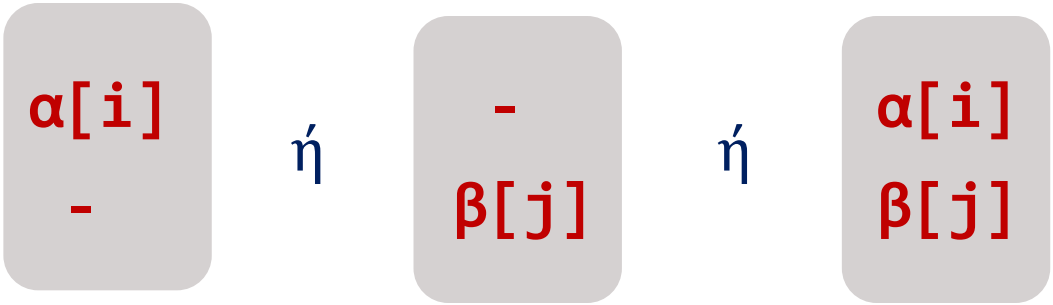
Διορθωτική Απόσταση

- Θα πρέπει να εκφράσουμε το υποπρόβλημα E(i,j) συναρτήσει μικρότερων υποπροβλημάτων !!!
- Τι γνωρίζουμε για την βέλτιστη διόρθωση (στοίχιση) ανάμεσα στις συμβολοσειρές α[1..i] και β[1..j] ?
- Γνωρίζουμε ότι για την **δεξιότερη στήλη** ισχύει μία από τις ακόλουθες τρεις περιπτώσεις :



Διορθωτική Απόσταση

- Το *κόστος* σε κάθε περίπτωση:



Διορθωτική Απόσταση

- Το *κόστος* σε κάθε περίπτωση:

α[i]	ή	**-**	ή	**α[i]**
-		**β[j]**		**β[j]**

- **1η** Περίπτωση:

Επιφέρει κόστος **1**
Και απομένει να διορθώσουμε (στοιχίσουμε) τις συμβολοσειρές

α[1..i-1] και β[1..j]

Όμως, αυτό είναι ακριβώς το υποπρόβλημα **E(i-1, j)**

- Το *κόστος* σε κάθε περίπτωση:

α[i]		**-**		**α[i]**
-	ή	**β[j]**	ή	**β[j]**

- **2η** Περίπτωση:

Επιφέρει κόστος **1**
Και απομένει να διορθώσουμε (στοιχίσουμε) τις συμβολοσειρές

α[1..i] και β[1..j-1]

Όμως, αυτό είναι ακριβώς το υποπρόβλημα **E(i, j-1)**

- Το *κόστος* σε κάθε περίπτωση:

α[i] **-**	ή	**-** **β[j]**	ή	**α[i]** **β[j]**

- **3η** Περίπτωση:

Επιφέρει κόστος **1** (εάν α[i]≠β[j])
ή κόστος **0** (εάν α[i]=β[j])
Και απομένει να διορθώσουμε τις

α[1..i-1] και β[1..j-1]

Τώρα, το υποπρόβλημα είναι ακριβώς το **E(i-1, j-1)**

Διορθωτική Απόσταση

- Επομένως, έχουμε εκφράσει το **E(i, j)** συναρτήσει **τριών *μικροτέρων*** υποπροβλημάτων:

  E(i-1, j) **E(i, j-1)** **E(i-1, j-1)**

- Όμως, δεν γνωρίζουμε ποιο από αυτά είναι σωστό!!!

  Οπότε θα τα δοκιμάσουμε όλα και θα επιλέξουμε το καλύτερο:

$$E(i,j) = \min\{1+E(i-1,j),\ 1+E(i,j-1),\ \{0|1\}+E(i-1,j-1)\}$$

- **Αλγόριθμος**

```
Edit-distance(a[1..n],b[1..m])
1. Initialize()
2. for i ← 1 to n
       for j ← 1 to m
3. return E(n,m)
```

```
for i ← 1 to n
        E(i,0) = i
for j ← 1 to m
        E(0,j) = j
```

```
E(i,j) ← min{1+E(i-1,j), 1+E(i,j-1),
                    {0|1} +E(i-1,j-1)}
```

- **Αλγόριθμος**

```
Edit-distance(a[1..n],b[1..m])
1. Initialize()
2. for i ← 1 to n
       for j ← 1 to m
           E(i,j) ← min{1+E(i-1,j), 1+E(i,j-1),
                        {0|1}+E(i-1,j-1)}
3. return E(n,m)
```

Συνολική Πολυπλοκότητα

O(nm)

Παράδειγμα

Δύο συμβολοσειρές

E X P O N E N T I A L

P O L Y N O M I A L

Υποπρόβλημα

E(4, 3)

- Το **E(4,3)** αντιστοιχεί στις συμβολοσειρές **EXPO** και **POL**

- **Εδώ, θέσαμε το Ερώτημα !!!**

  Τι ισχύει για την δεξιότερη στήλη σε μια βέλτιστη λύση του υποπροβλήματος E(4,3) ?

Παράδειγμα

Δύο συμβολοσειρές

E X P O N E N T I A L

P O L Y N O M I A L

Υποπρόβλημα

E(4, 3)

- Τρεις περιπτώσεις μπορούν να ισχύουν:

O		-		O
-	ή	L	ή	L

Άρα, $E(4,3) = \min\{1+E(3,3), 1+E(4,2), 1+E(3,2)\}$

Διορθωτική Απόσταση

Παράδειγμα

Οι λύσεις όλων των υποπροβλημάτων **E(i,j)** σχηματίζουν ένα **11×10** διδιάστατο πίνακα

		P	O	L	Y	N	O	M	I	A	L
	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	6	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

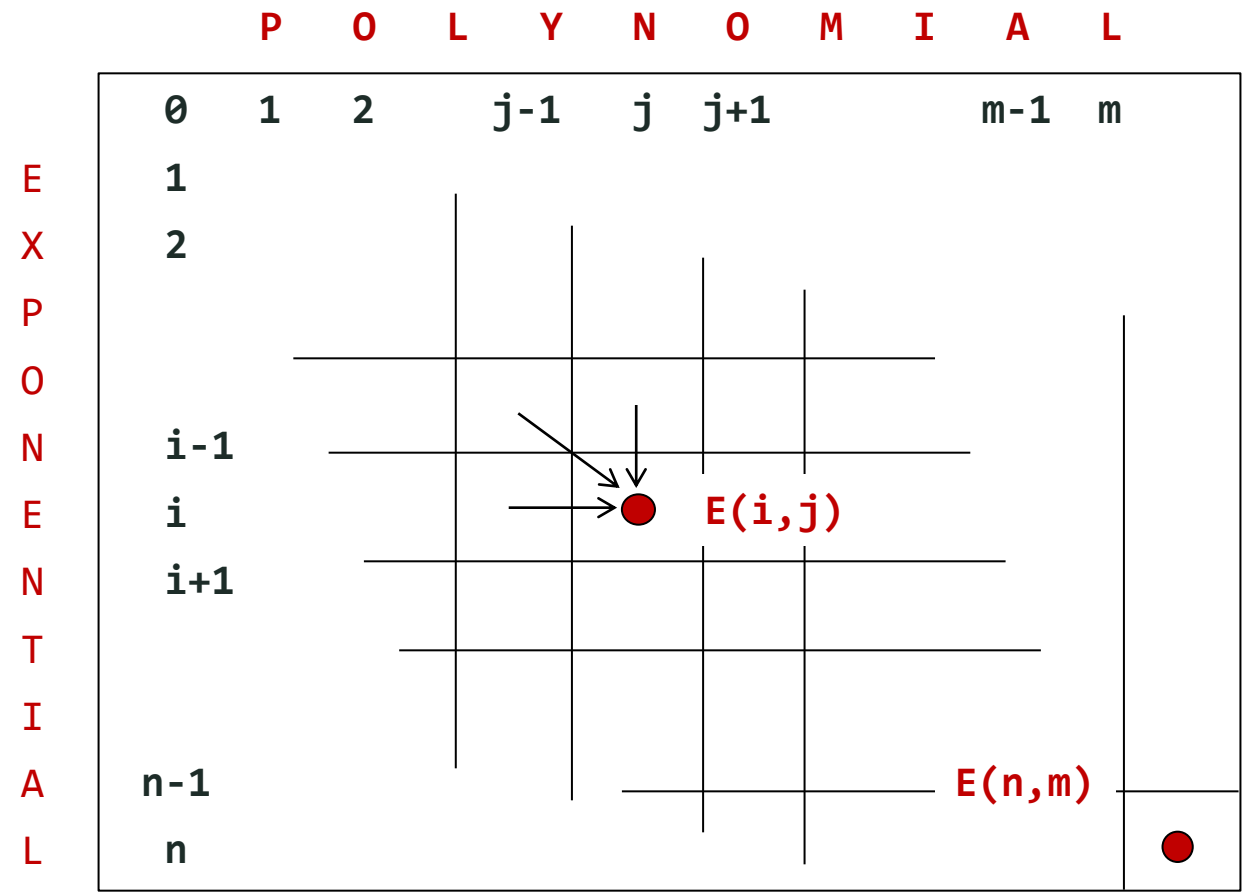
ΛΥΣΗ

E(11,10) = 6

Διορθωτική Απόσταση

Παράδειγμα

Γενικά, όλα τα υποπροβλήματα **E(i,j)** σχηματίζουν έναν **n×m** διδιάστατο πίνακα



ΛΥΣΗ

Διορθωτική απόσταση = E(n,m)

Παράδειγμα

Αλγόριθμος

```
Edit-distance(a[1..11],b[1..10])
   a[1..11] =  EXPONENTIAL
   b[1..10] =  POLYNOMIAL
return E(11,10) = 6
```

E	X	P	O	N	E	N	-	T	I	A	L
-	-	P	O	L	Y	N	O	M	I	A	L

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Έστω ότι θέλουμε να πολλαπλασιάσουμε 4 πίνακες

$$A \times B \times C \times D$$

διαστάσεων

$$50 \times 20, \quad 20 \times 1, \quad 1 \times 10 \quad \text{και} \quad 10 \times 100$$

- Ο πολλαπλασιασμός πινάκων δεν είναι αντιμεταθετός ($A \times B \neq B \times A$), αλλά είναι προσεταιριστηκός, που σημαίνει: $(A \times B) \times C = A \times (B \times C)$

Μπορούμε να υπολογίσουμε το γινόμενο των 4 πινάκων μας με πολλούς διαφορετικούς τρόπους !

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Έστω ότι θέλουμε να πολλαπλασιάσουμε 4 πίνακες

$$A \times B \times C \times D$$

διαστάσεων

$$50 \times 20, \quad 20 \times 1, \quad 1 \times 10 \quad \text{και} \quad 10 \times 100$$

Ερώτημα !!! **Υπάρχει διαφορά ?**

Εάν ΝΑΙ !!! **Ποιος τρόπος είναι ο καλύτερος ?**

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Πολλαπλασιασμός

$$A \times B \times C \times D$$

Ο πολλαπλασιασμός δύο πινάκων $A \times B$ διαστάσεων $n \times m$ και $m \times p$ απαιτεί $n \cdot m \cdot p$ πολλαπλασιασμούς

A	50 × 20
B	20 × 1
C	1 × 10
D	10 × 100

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120.200
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60.200
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7.000

3 Πολλαπλασιασμός Ακολουθίας Πινάκων

Πρόβλημα

Μας δίδεται μια ακολουθία n πινάκων (συμβατών διαστάσεων)

$$A_1, A_2, \ldots, A_n$$

και μας ζητείται να υπολογίσουμε το γινόμενο

$$A_1 \times A_2 \times \ldots \times A_n$$

κάνοντας το ελάχιστο πλήθος πολλαπλασιασμών.

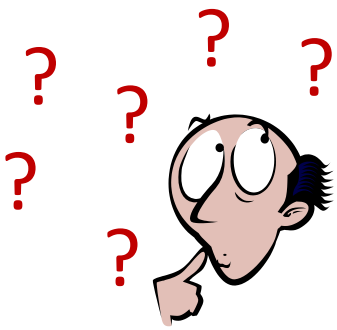
Πολλαπλασιασμός Ακολουθίας Πινάκων

- Από το εισαγωγικό παράδειγμα είδαμε ότι η σειρά των πολλαπλασιασμών επηρεάζει πολύ τον χρόνο επίλυσης του προβλήματος !!!

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120.200
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60.200
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7.000

Ερώτημα !!!

Να ακολουθήσουμε άπληστη προσέγγιση ?



Άπληστη Τεχνική ?

Κανόνας 1: Επίλεξε κάθε φορά τον *φθηνότερο* διαθέσιμο πολλαπλασιασμό !!!

A	50	×	20
B	20	×	1
C	1	×	10
D	10	×	100

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
$A \times ((B \times C) \times D)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120.200
$(A \times (B \times C)) \times D$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60.200
$(A \times B) \times (C \times D)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7.000

Φθηνότερος πολλαπλασιασμός : $\mathbf{B \times C}$ με κόστος $20 \cdot 1 \cdot 10$

Αποτυχία !!!

$\mathbf{A \times B \times C \times D} \quad \Rightarrow \quad \mathbf{A \times (B \times C) \times D}$

Άπληστη Τεχνική ?

Κανόνας 2: Επίλεξε κάθε φορά τον *ακριβότερο* διαθέσιμο πολλαπλασιασμό !!!

A	1	×	10
B	10	×	1
C	1	×	4

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
(A × B) × C	1·10·1 + 1·1·4	14
A × (B × C)	10·1·4 + 1·10·4	80

Ακριβότερος πολλαπλασιασμός : **B × C** με κόστος 10·1·4

Αποτυχία !!!

A × B × C ⇒ A × (B × C)

Άπληστη Τεχνική ?

Κανόνας 3: Επίλεξε τον πολλαπλασιασμό που *απαλείφει την μεγαλύτερη διάσταση* !!!

A	3	×	10
B	10	×	5
C	5	×	1

Διάταξη Παρενθέσεων	Υπολογισμός Κόστους	Κόστος
(A × B) × C	3·10·5 + 3·5·1	165
A × (B × C)	10·5·1 + 3·10·1	80

Πολλαπλασιασμός που *απαλείφει μεγαλύτερης διάστασης* : **A × B** **Αποτυχία !!!**

A × B × C ⇒ (A × B) × C

Πολλαπλασιασμός Ακολουθίας Πινάκων

Δυναμικός Προγραμματισμός !!!

Πως θα βρούμε τη **βέλτιστη σειρά** των πολλαπλασιασμών όταν θέλουμε να υπολογίσουμε το γινόμενο n πινάκων

$$A_1 \times A_2 \times A_3 \times \ldots \times A_{n-1} \times A_n$$

διαστάσεων $m_0 \times m_1$, $m_1 \times m_2$, $m_2 \times m_3$, ..., $m_{n-2} \times m_{n-1}$, $m_{n-1} \times m_n$.

Παρατήρηση

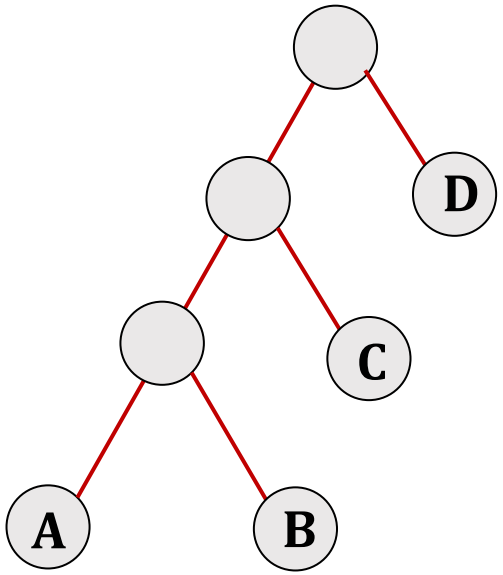
Μια διάταξη παρενθέσεων μπορεί να αναπαρασταθεί πολύ φυσικά με ένα δυαδικό δένδρο:

- ✓ οι πίνακες αντιστοιχούν στα φύλλα
- ✓ οι ενδιάμεσοι κόμβοι είναι τα ενδιάμεσα γινόμενα
- ✓ η ρίζα είναι το τελικό αποτέλεσμα

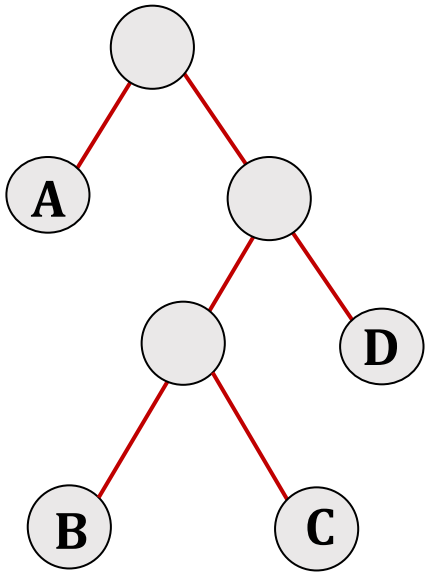
Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

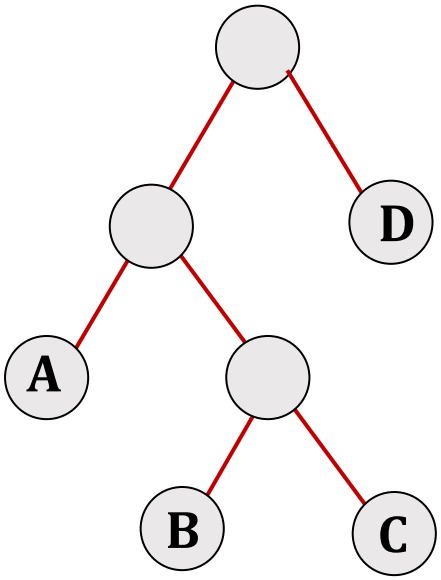
$((A \times B) \times C) \times D)$



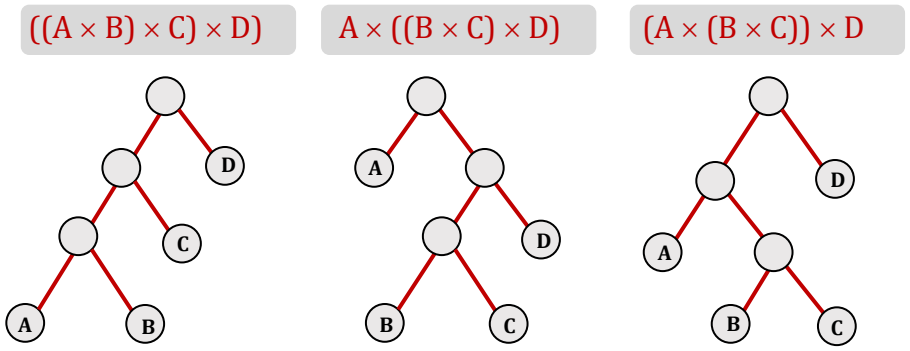
$A \times ((B \times C) \times D)$



$(A \times (B \times C)) \times D$



Δένδρα - Διατάξεις

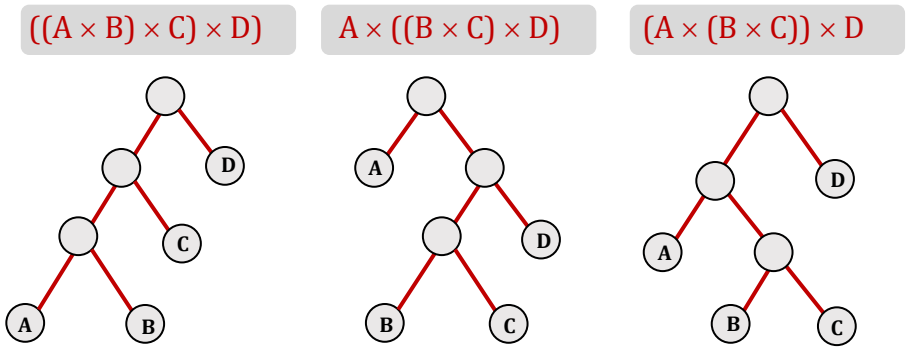


- Οι δυνατές διατάξεις με τις οποίες μπορεί να εκτελεστεί ο πολλαπλασιασμός **n** πινάκων **αντιστοιχούν** στα διαφορετικά πλήρη δυαδικά δένδρα **n** κόμβων

- **Πόσα είναι τα δένδρα αυτά ?** $\Omega(2^n)$

Δυστυχώς, το πλήθος τους είναι εκθετικό !!!

Δένδρα - Διατάξεις



- Ένα από τα δένδρα αυτά είναι **βέλτιστο**!!
- Εάν το δένδρο **T** είναι βέλτιστο, τότε είναι βέλτιστα και τα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ της ρίζας του.

Ερώτηση! Ποια είναι τα υποπροβλήματα που αντιστοιχούν στα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ του βέλτιστου δένδρου **T ?**

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Τα υποπροβλήματα που αντιστοιχούν στα υποδένδρα $\mathbf{T_1}$ και $\mathbf{T_2}$ της ρίζας ενός βέλτιστου δένδρου **T**

  είναι γινόμενα της μορφής: $\mathbf{A_1 \times A_2 \times \ldots \times A_k}$ (T_1)

  $\mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_n}$ (T_2)

- Γενικά, το υποπρόβλημα που αντιστοιχεί σε ένα υποδένδρο ενός βέλτιστου δένδρου **T**

  είναι γινόμενο της μορφής: $\mathbf{A_i \times A_{i+1} \times \ldots \times A_j = A_{i..j}}$

Πολλαπλασιασμός Ακολουθίας Πινάκων

- **Ορίζουμε**

C[i,j] = min κόστος του πολλαπλασιασμού $A_i \times A_{i+1} \times \ldots \times A_j$

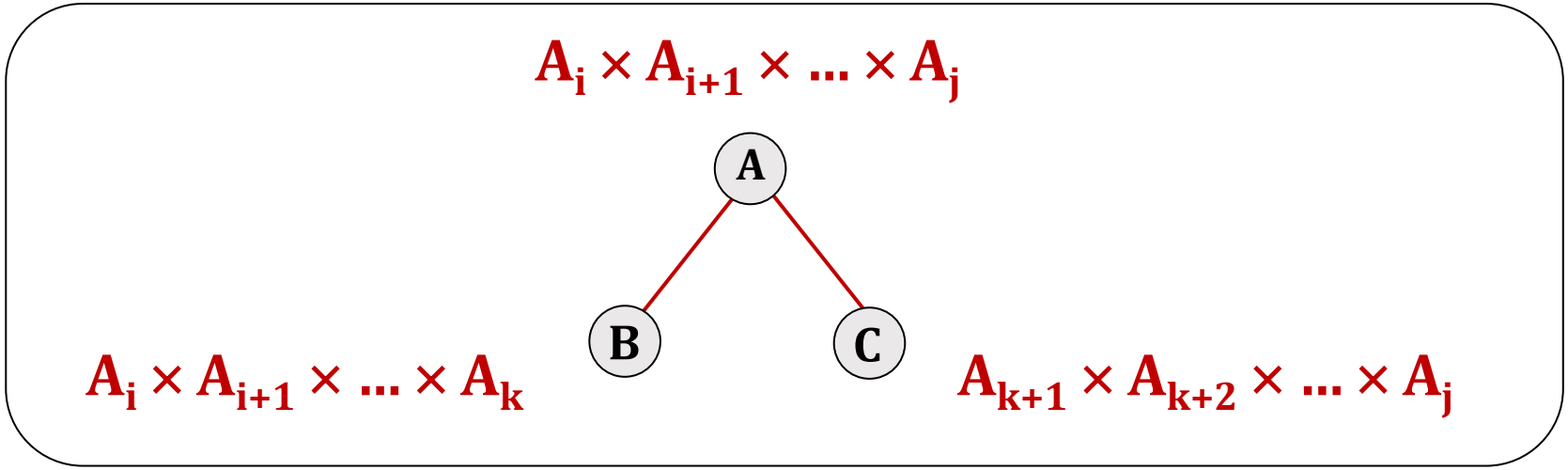
όπου κόστος είναι το πλήθος των πολλαπλασιασμών αυτού του υποπροβλήματος

- Το μικρότερο υποπρόβλημα το παίρνουμε όταν **i = j**, όπου σε αυτή την περίπτωση δεν έχουμε κάτι να πολλαπλασιάσουμε, οπότε C[i,j]=0

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Θεωρούμε το **βέλτιστο υποδένδρο T_{ij}** , $j > i$, για το γινόμενο $A_i \times A_{i+1} \times \ldots \times A_j$

- Η ρίζα του υποδένδρου T_{ij} χωρίσει το γινόμενο



σε δύο τμήματα, για κάποιο k, $i \leq k \leq j$.

Πολλαπλασιασμός Ακολουθίας Πινάκων

- Τότε, το κόστος C[i,j] του υποδένδρου $\mathbf{T_{ij}}$ θα είναι:

το κόστος υπολογισμού των μερικών γινομένων

$$\mathbf{A_i \times A_{i+1} \times \ldots \times A_k} \quad \text{και} \quad \mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_j}$$

συν το κόστος του πολλαπλασιασμού των πινάκων

$$\mathbf{A_{i\,..k}} \text{ και } \mathbf{A_{k+1\,..j}}$$

που είναι $m_{i-1} \cdot m_k \cdot m_j$.

Διάσταση $\mathbf{A_i}$ $m_{i-1} \times m_i$

Πρέπει απλώς να βρούμε το σημείο χωρισμού **k** για το οποίο το κόστος C[i,j] είναι το μικρότερο!!!

- Τότε, το κόστος C[i,j] του υποδένδρου $\mathbf{T_{ij}}$ θα είναι:

το κόστος υπολογισμού των μερικών γινομένων

$$\mathbf{A_i \times A_{i+1} \times \ldots \times A_k} \quad \text{και} \quad \mathbf{A_{k+1} \times A_{k+2} \times \ldots \times A_j}$$

συν το κόστος του πολλαπλασιασμού των πινάκων

$$\mathbf{A_{i\,..k}} \text{ και } \mathbf{A_{k+1\,..j}}$$

που είναι $m_{i-1} \cdot m_k \cdot m_j$.

Διάσταση $\mathbf{A_i}$ $m_{i-1} \times m_i$

$$C[i,j] = \min_{i \le k \le j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

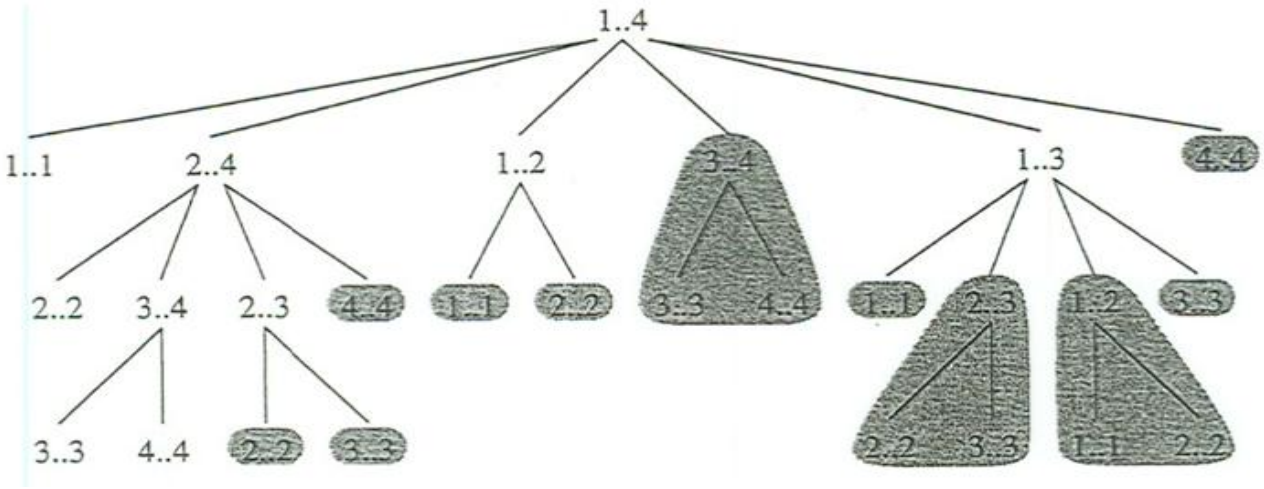
- Σχέση που δίδει τη λύση ενός υποπροβλήματος από «μικρότερα» υποπροβλήματα :

$$C[i,j] = \min_{i \le k \le j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}$$

- **Αναδρομή?**

ΟΧΙ, ευχαριστώ!!!

$\Omega(2^n)$



Πολλαπλασιασμός Ακολουθίας Πινάκων

- Σχέση που δίδει τη λύση ενός υποπροβλήματος από «μικρότερα» υποπροβλήματα

$$C[i,j] = \min_{i \le k \le j} \{C[i,k] + C[k+1,j] + m_{i-1} \cdot m_k \cdot m_j\}$$

- **Δυναμικός Προγραμματισμός ?**

ΝΑΙ, θα ήθελα !!!

Ερώτηση: Πόσα υποπροβλήματα έχω ?

Ένα για κάθε ζεύγος (i, j) $\Rightarrow$ $\mathbf{O(n^2)}$

- **Αλγόριθμος**

```
Matrix-Chain()
1. for i ← 1 to n: C(i,i)=0
2. for s ← 1 to n-1
       for i ← 1 to n-s
           j ← i + s
           for k ← i to j
               C[i,j]←min{C[i,k]+C[k+1,j]+m_{i-1}·m_k·m_j}
3. return C[1,n]
```

Συνολική Πολυπλοκότητα

$O(n^3)$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

$\mathbf{A_1}$	30 × 35
$\mathbf{A_2}$	35 × 15
$\mathbf{A_3}$	15 × 5
$\mathbf{A_4}$	5 × 10
$\mathbf{A_5}$	10 × 20
$\mathbf{A_6}$	20 × 25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

```
C[1,1] = 0
C[2,2] = 0
   ...
C[6,6] = 0
```

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$C[1,2] = 30 \cdot 35 \cdot 15 = 15,750$

$C[2,3] = 35 \cdot 15 \cdot 5 = 2,625$

...

$C[5,6] = 10 \cdot 20 \cdot 25 = 5,000$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

$\mathbf{A_1}$	30 × 35
$\mathbf{A_2}$	35 × 15
$\mathbf{A_3}$	15 × 5
$\mathbf{A_4}$	5 × 10
$\mathbf{A_5}$	10 × 20
$\mathbf{A_6}$	20 × 25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13,000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7,125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11,375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30 × 35
A_2	35 × 15
A_3	15 × 5
A_4	5 × 10
A_5	10 × 20
A_6	20 × 25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Τελικό Αποτέλεσμα: C[1,6] = 15,125

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

Ποια είναι η βέλτιστη λύση της $A_1 \times A_2 \times \ldots \times A_6$ **?**

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	7,125	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13,000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7,125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11,375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

$\mathbf{A_1}$	30 × 35
$\mathbf{A_2}$	35 × 15
$\mathbf{A_3}$	15 × 5
$\mathbf{A_4}$	5 × 10
$\mathbf{A_5}$	10 × 20
$\mathbf{A_6}$	20 × 25

	1	2	3	4	5	6
1	0	15,750	7,875	9,375	11,875	15,125
2		0	2,625	4,375	3	10,500
3			0	750	2,500	5,375
4				0	1,000	3,500
5					0	5,000
6						0

$$C[2,5] = \min \begin{cases} C[2,2]+C[3,5]+m_1 \cdot m_2 \cdot m_5 = 13,000 \\ C[2,3]+C[4,5]+m_1 \cdot m_3 \cdot m_5 = 7,125 \\ C[2,4]+C[5,5]+m_1 \cdot m_4 \cdot m_5 = 11,375 \end{cases}$$

Πολλαπλασιασμός Ακολουθίας Πινάκων

Παράδειγμα

A_1	30 × 35
A_2	35 × 15
A_3	15 × 5
A_4	5 × 10
A_5	10 × 20
A_6	20 × 25

	1	2	3	4	5	6
1	0	1	1	3	3	3
2		0	2	3	3	3
3			0	3	3	3
4				0	4	5
5					0	5
6						0

C[1,6] = 15,125

$$A_1 \times A_2 \times A_3 \times A_4 \times A_5 \times A_6$$

$$(A_1 \times A_2 \times A_3) \times (A_4 \times A_5 \times A_6)$$

$$(A_1 \times (A_2 \times A_3)) \times ((A_4 \times A_5) \times A_6)$$

4 Σακίδιο

Κατά την διάρκεια μιας κλοπής, ένας διαρρήκτης βρίσκει περισσότερα λάφυρα απ΄ όσα περίμενε και πρέπει να αποφασίσει τι θα πάρει !!!

Το «σακίδιό» του (knap-sack) μπορεί να μεταφέρει **W** κιλά !!!

Υπάρχουν **n** είδη

με βάρη $w_1, w_2, \ldots, w_n$

και αξία $v_1, v_2, \ldots, v_n$

4 Σακίδιο

Το ερώτημα που τον απασχολεί !!!
Ποιος είναι ο πλέον πολύτιμος συνδυασμός ειδών που μπορεί να βάλει στο σακίδιό του ?

Το «σακίδιό» του (knap-sack) μπορεί να μεταφέρει **W** κιλά !!!

Υπάρχουν **n** είδη

με βάρη $w_1, w_2, \ldots, w_n$

και αξία $v_1, v_2, \ldots, v_n$

4 Σακίδιο

Παράδειγμα

Έστω ότι το σακίδιο χωράει **W = 10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

4 Σακίδιο

Παράδειγμα

Έστω **W = 10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

Υπάρχουν απεριόριστες ποσότητες από κάθε είδος

Υπάρχει μία μόνο ποσότητα από κάθε είδος

4 Σακίδιο

Παράδειγμα

Έστω **W = 10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

Υπάρχουν απεριόριστες ποσότητες από κάθε είδος

Βέλτιστη Επιλογή

1 από είδος Α w=6 30€
2 από είδος Δ w=4 18€

Συνολική Τιμή **48€**

4 Σακίδιο

Παράδειγμα

Έστω **W = 10** κιλά, και

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Δύο Εκδοχές

Βέλτιστη Επιλογή

1 από είδος Α w=6 30€
1 από είδος Γ w=4 16€
Συνολική Τιμή **46€**

Υπάρχει μία μόνο ποσότητα από κάθε είδος

4 Σακίδιο (knap-sack)

Πρόβλημα

Μας δίδεται ένα σύνολο n αντικειμένων

$$S = \{a_1, a_2, \ldots, a_n\}$$

με ακέραια βάρη w_i και αξίες v_i $(1 \le i \le n)$, και μία τιμή W,

και μας ζητείται να υπολογίσουμε ένα υποσύνολο αντικειμένων $S' \subseteq S$ με μέγιστη συνολική αξία C και με συνολικό βάρος $\le \mathbf{W}$, δηλαδή,

$$\max C = \sum_{a_i \in S'} v_i \quad \text{και} \quad \sum_{a_i \in S'} w_i \le W$$

Σακίδιο : Με επανάληψη

- **Ας μελετήσουμε την πρώτη εκδοχή του προβλήματος !!!**
  και ας δούμε τι συμβαίνει όταν **επιτρέπονται οι επαναλήψεις** !

  Το κρίσιμο ερώτημα :

  Ποια είναι τα υποπροβλήματα ?

- Μπορούμε να **«συρρικνώσουμε»** το αρχικό πρόβλημα με δύο τρόπους:

(1) Μπορούμε να εξετάσουμε την περίπτωση σακιδίων μικρότερης χωρητικότητας $\mathbf{w \leq W}$

ή

(2) Την περίπτωση λιγότερων ειδών $\mathbf{1, 2, \ldots, i \leq n}$

Σακίδιο : Με επανάληψη

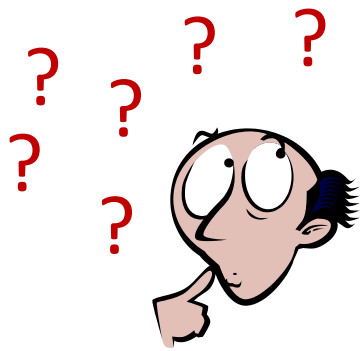
- Ο πρώτος περιορισμός επιβάλει να έχουμε μικρότερες χωρητικότητες !

  Ορίζουμε:

  C(w) = Max τιμή που παίρνουμε με σακίδιο χωρητικότητας w

- **Ερώτημα !!!**

  Μπορούμε να εκφράσουμε το υποπρόβλημα **C(w)** συναρτήσει μικρότερων υποπροβλημάτων ?



Σακίδιο : Με επανάληψη

- Έστω μια βέλτιστη λύση του υποπροβλήματος **C(w),** η οποία περιλαμβάνει :
  - ✓ τα είδη $p, \ldots, i, \ldots, q$
  - ✓ τέτοια ώστε $w_p + \cdots + w_i + \cdots + w_q \leq w$
  - ✓ με max τιμή $C(w) = v_p + \cdots + v_i + \cdots + v_q$
- Εάν η βέλτιστη λύση του υποπροβλήματος **C(w)** περιλαμβάνει το είδος **i** βάρους w_i, τότε η αφαίρεση του **i** από το σακίδιο δίδει μια βέλτιστη λύση για το υποπρόβλημα **$C(w-w_i)$!!!**

Επομένως :

> Μπορούμε να πάρουμε το υποπρόβλημα **C(w)** από το μικρότερο **$C(w-w_i)$** προσθέτοντας την αξία v_i του είδους w_i, για κάποιο **i**

Σακίδιο : Με επανάληψη

- Με άλλα λόγια:

> Το υποπρόβλημα **C(w)** είναι απλώς το **C(w- w_i) + v_i**, για κάποιο **i**.

Ποιο είναι αυτό το **i** ? ... δεν γνωρίζουμε !!!

Τότε, δεν έχουμε παρά να πάρουμε όλες τις δυνατότητες!

- Ορίστε πως !!!

$$C(w) = \max_{i\,:\,w_i \le w} \{C(w-w_i) + v_i\}$$

όπου, ως συνήθως, η σύμβασή μας είναι ότι η μέγιστη τιμή ενός κενού συνόλου είναι 0 !!!

Σακίδιο : Με επανάληψη

- Τελειώσαμε ! … ο αλγόριθμος τώρα γράφετε μόνος του και είναι ιδιαιτέρα κομψός !!!

Αλγόριθμος

```
Knap-sack1(w[1..n], v[1..n], W)
1. C(0)=0
2. for w ← 1 to W
```

$$C(w) \leftarrow \max\{C(w-w(i)) + v(i) : w(i) \leq w\}$$

```
3. return C(W)
```

Συνολική Πολυπλοκότητα

O(nW)

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

0 1 2 3 4 5 6 7 8 9 10

Χωρητικότητες Σακιδίου **w = 0, 1, 2, ..., i, ..., n**

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2, 3, 4\} \le 5$

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18						
C(0)	C(1)	C(2)	C(3)	C(4)						

w = 5

$C(5) = C(5-2) + 9 = 23$

$C(5) = C(5-3) + 14 = 23$

$C(5) = C(5-4) + 16 = 16$

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2, 3, 4\} \le 5$

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23					
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)					

$C(5) = C(5-2) + 9 = 23$

$C(5) = C(5-3) + 14 = 23$

$C(5) = C(5-4) + 16 = 16$

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2, 3, 4, 6\} \le 6$

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23					
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)					

w = 6

$C(6) = C(6-2) + 9 = 27$

$C(6) = C(6-3) + 14 = 28$

$C(6) = C(6-4) + 16 = 25$

$C(6) = C(6-6) + 30 = 30$

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

$\{2, 3, 4, 6\} \le 6$

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23	30				
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)				

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23	30	30			
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)			

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W=10**

$$C(w)=\max\{C(w-w(i))+v(i) : w(i)\le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23	30	30	39		
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)		

Σακίδιο : Με επανάληψη

Παράδειγμα

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα Σακιδίου **W=10**

$$C(w)=\max\{C(w-w(i))+v(i): w(i)\le w\}$$

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23	30	30	39	39	
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)	C(9)	

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W=10**

$$C(w)=\max\{C(w-w(i))+v(i): w(i)\le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

0	1	2	3	4	5	6	7	8	9	10
0	0	9	14	18	23	30	30	39	39	48
C(0)	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)	C(9)	C(10)

Σακίδιο : Με επανάληψη

Παράδειγμα

Χωρητικότητα Σακιδίου **W=10**

$$C(w)=\max\{C(w-w(i))+v(i): w(i)\le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

C(10) = 48 1 είδος Α αξίας 30
2 είδη Δ συνολικής αξίας 18

10

48

C(10)

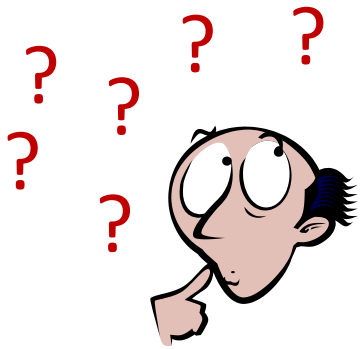
Σακίδιο : Με επανάληψη

- **Ας μείνουμε λίγο ακόμα** σε αυτή την εκδοχή του προβλήματος !

  - ✓ Γνωρίζουμε ότι ένα πρόβλημα ΔΠ εκφράζεται με ένα γράφημα **G** τύπου DAG !!!

Εύλογο ερώτημα !!!

Ποιο είναι το **DAG** του προβλήματος **του σακιδίου με επανάληψη ?**



Σακίδιο : Με επανάληψη

- Το DAG του Προβλήματος

Χωρητικότητα **W = 10**

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Κόμβοι DAG ⇔ Χωρητικότητα Σακιδίου **W**

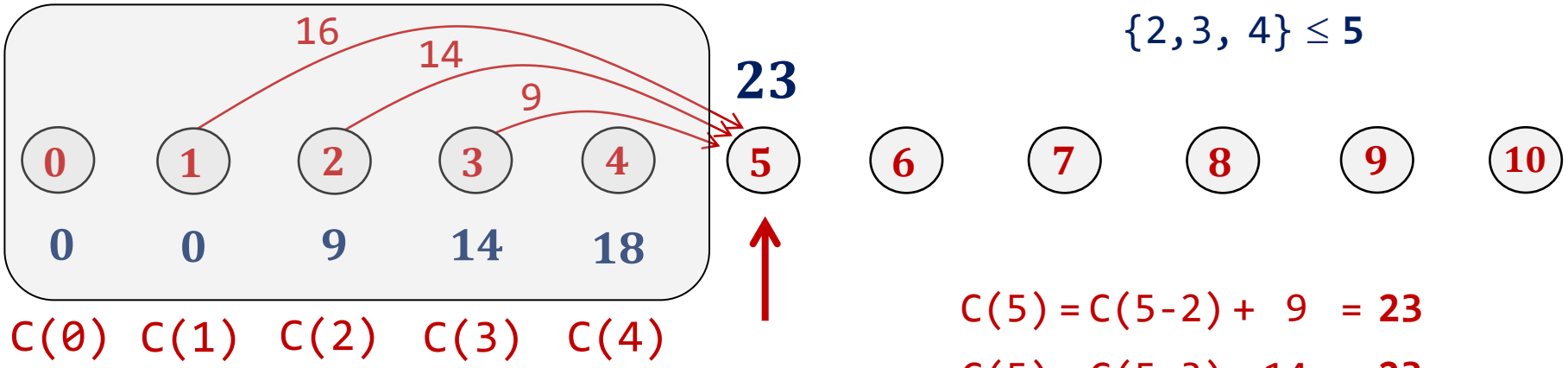
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Χωρητικότητα **W=10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €



$\{2,3,4\} \le 5$

w = 5

$C(5) = C(5-2) + 9 = 23$

$C(5) = C(5-3) + 14 = 23$

$C(5) = C(5-4) + 16 = 16$

Σακίδιο : Με επανάληψη

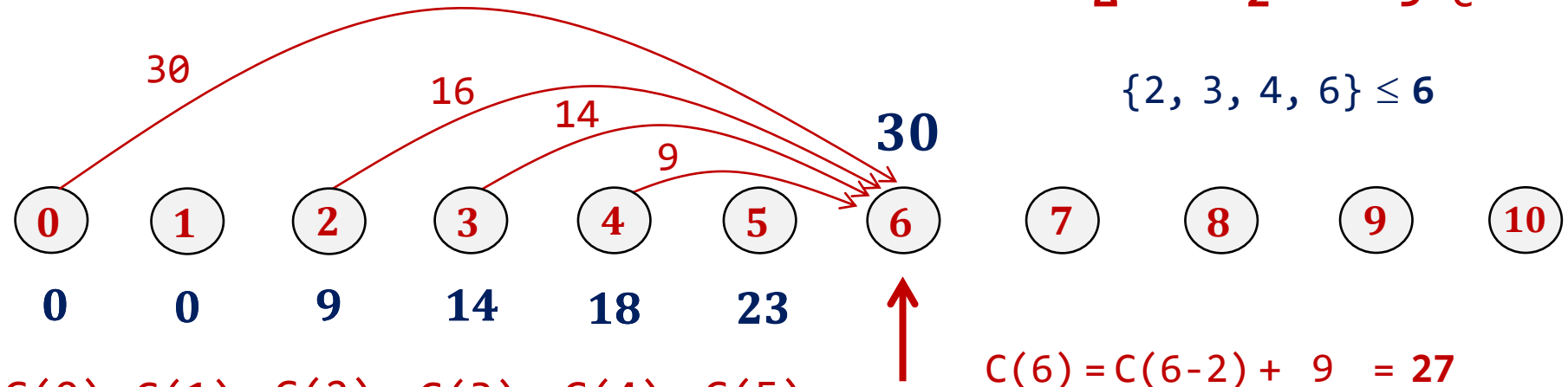
Το DAG του Προβλήματος

Χωρητικότητα **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

Είδος	Βάρος	Τιμή
Α	**6**	**30 €**
Β	**3**	**14 €**
Γ	**4**	**16 €**
Δ	**2**	**9 €**

$\{2, 3, 4, 6\} \le 6$



$C(6) = C(6-2) + 9 = \mathbf{27}$

$C(6) = C(6-3) + 14 = \mathbf{28}$

$C(6) = C(6-4) + 16 = \mathbf{25}$

$C(6) = C(6-6) + 30 = \mathbf{30}$

Σακίδιο : Με επανάληψη

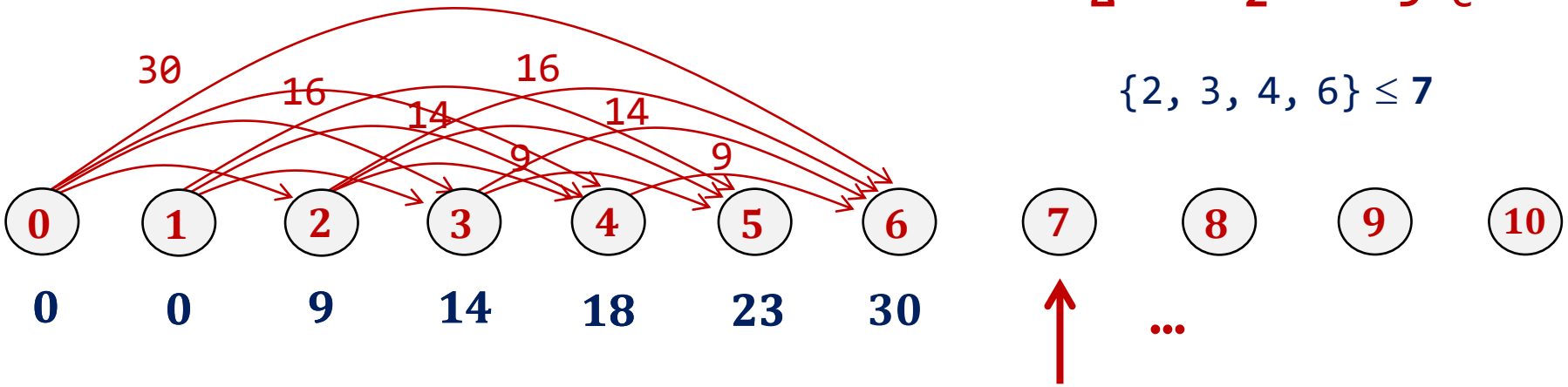
Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

$$C(w) = \max\{C(w-w(i)) + v(i) : w(i) \le w\}$$

$\{2, 3, 4, 6\} \le 7$



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

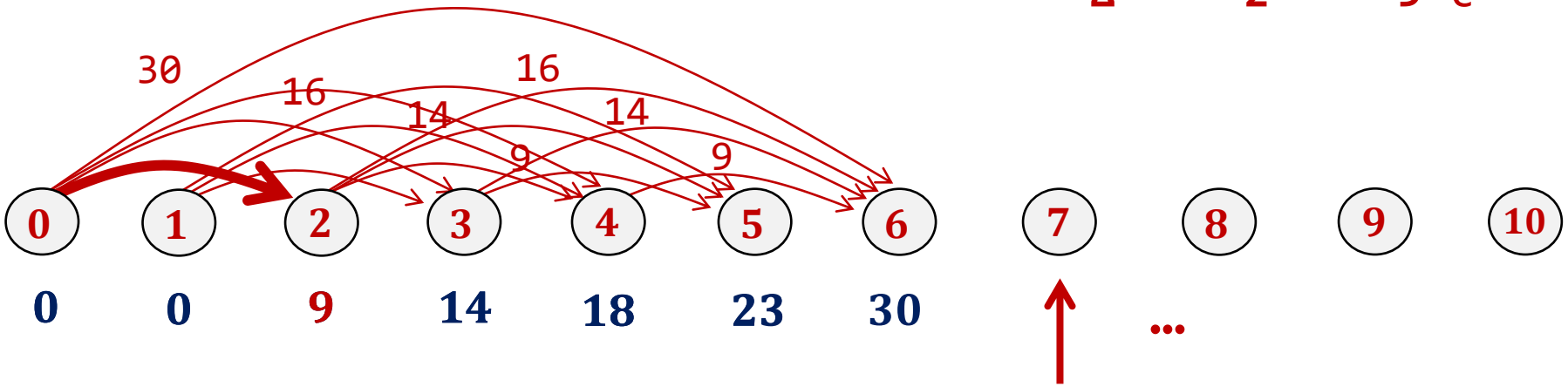
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

{Δ} λύση για w = 2 με C(w) = 9



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

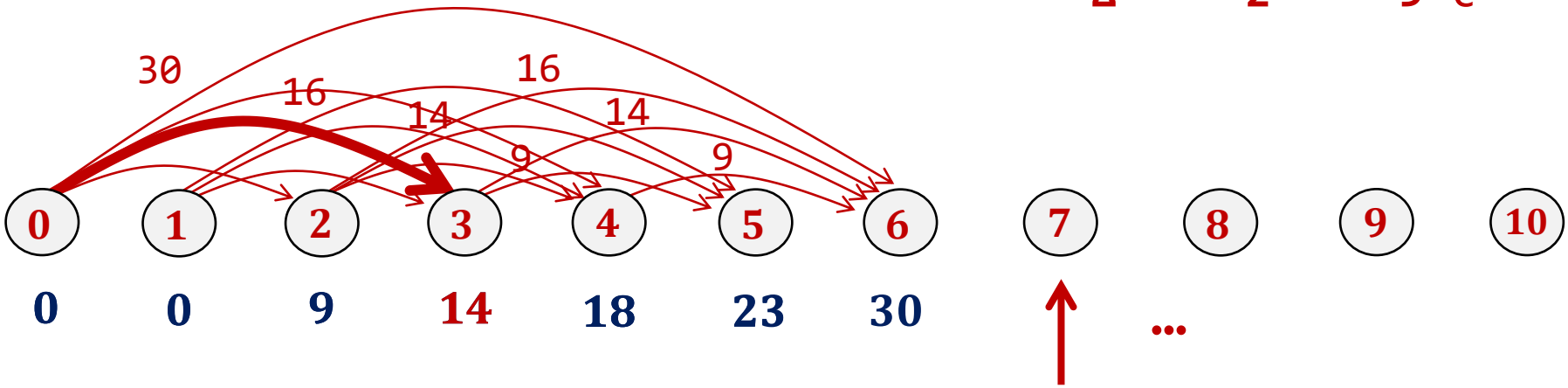
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W=10**

{Β} λύση για w = 3 με C(w) = 14



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

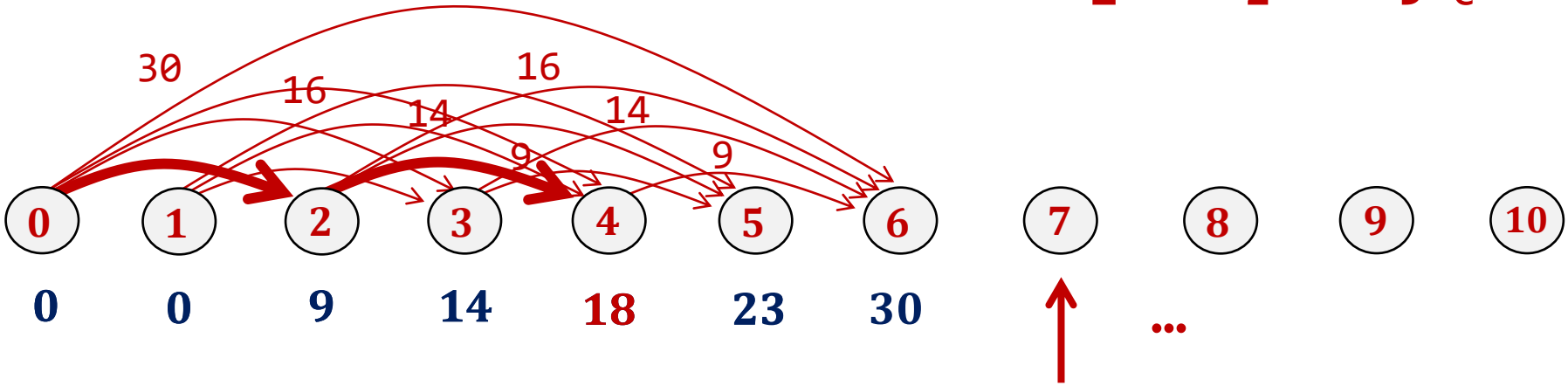
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

{Δ, Δ} λύση για w = 4 με C(w) = 18



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

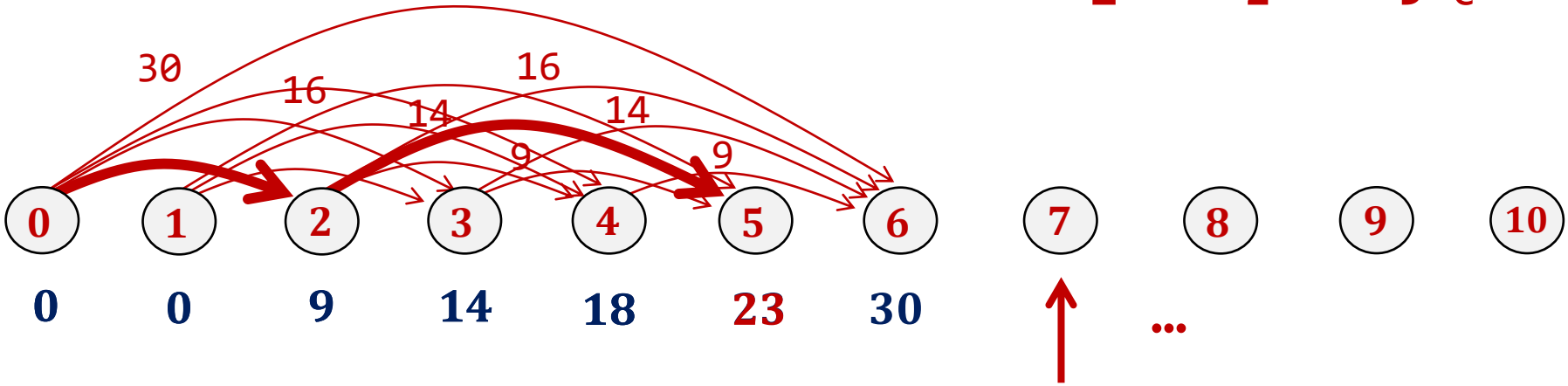
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

{Δ, Β} λύση για w = 5 με C(w) = 23



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

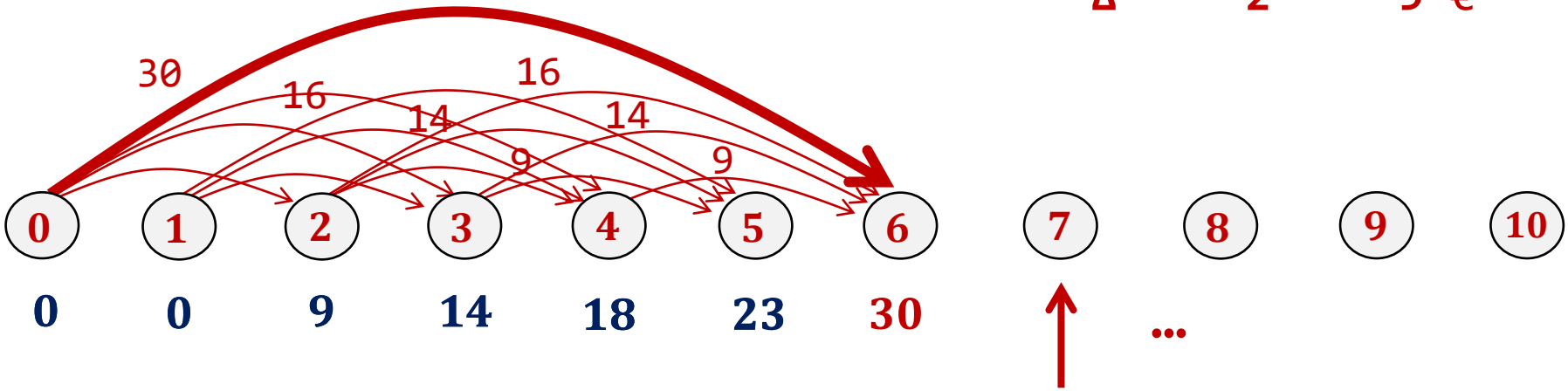
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

{A} λύση για w = 6 με C(w) = 30



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

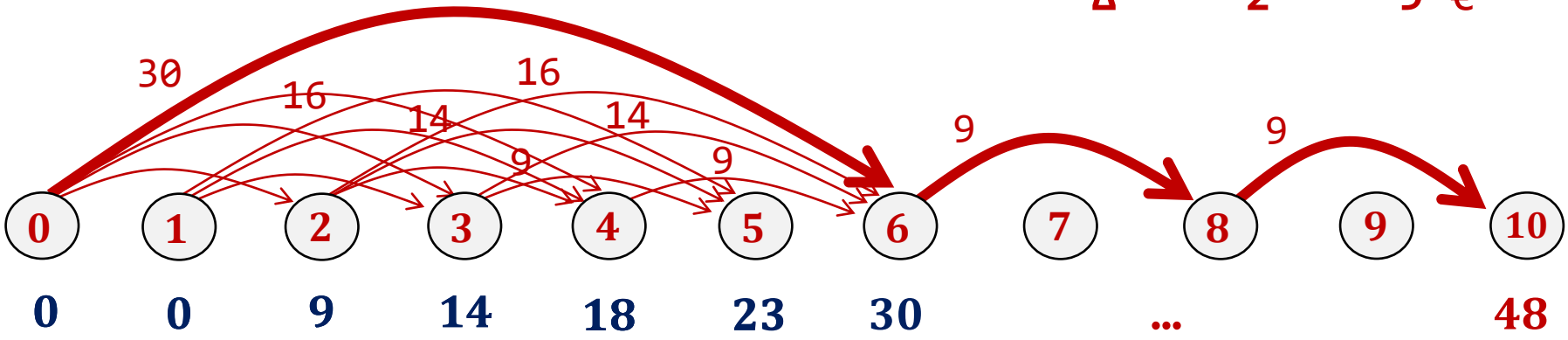
Σακίδιο : Με επανάληψη

Το DAG του Προβλήματος

Είδος	Βάρος	Τιμή
Α	6	30 €
Β	3	14 €
Γ	4	16 €
Δ	2	9 €

Χωρητικότητα **W = 10**

{Α,Δ,Δ} τελική λύση, C(w) = 48



Σακίδιο με επανάληψη ⇔ Max διαδρομή σε DAG

Σακίδιο : Χωρίς επανάληψη

- **Ερχόμαστε τώρα στην δεύτερη εκδοχή του προβλήματος !!!**

και ας δούμε τι συμβαίνει όταν…

δεν επιτρέπονται οι επαναλήψεις !

- Η γνώση της τιμής του υποπροβλήματος

$$\mathbf{C(w- w_i)}$$

δεν μας βοηθάει τώρα !!! καθώς δεν γνωρίζουμε

εάν το είδος **i** έχει ήδη χρησιμοποιηθεί ή όχι σε αυτό το υποπρόβλημα !!!

Σακίδιο : Χωρίς επανάληψη

- **Ερχόμαστε τώρα στην δεύτερη εκδοχή του προβλήματος !!!**

και ας δούμε τι συμβαίνει όταν…

δεν επιτρέπονται οι επαναλήψεις !

- Επομένως, πρέπει να βελτιώσουμε την έννοια του υποπροβλήματος

$$\mathbf{C(w- w_i)}$$

ώστε

να μεταφέρει πληροφορίες σχετικά με τα είδη που χρησιμοποιούνται σε αυτό !!!

Πως μπορούμε να το κάνουμε ?

Να προσθέσουμε μια δεύτερη παράμετρο, ας την πούμε **j**, $1 \leq j \leq n$, η οποία θα προσδιορίζει το σύνολο $\{1, 2, \ldots, j\}$ των j πρώτων ειδών από ένα σύνολο **n** συνολικά ειδών, οπότε:

Ορίζουμε

C(w, j) = Max τιμή που παίρνουμε με σακίδιο χωρητικότητας **w** και με είδη **1, 2, …, j**

Και τότε,

- Η απάντηση που ψάχνουμε είναι η **C(W, n)** !!!

- **Πως μπορούμε** να εκφράσουμε το υποπρόβλημα **C(w, j)** συναρτήσει μικροτέρων ?

Πολύ απλά... εάν έχουμε μια βέλτιστη λύση με είδη από το σύνολο {1, 2, …, j-1} , τότε το επόμενο είδος **j** ($1 \leq j \leq n$) :

- είτε είναι απαραίτητο να πετύχουμε βέλτιστη λύση
- είτε όχι !!!

Επομένως,

$$C(w,j) = \max \{C(w, j-1), C(w-w_j, j-1) + v_j\}$$

Η δεύτερη περίπτωση καλείται μόνο όταν $w_j \leq w$

- **Και έτσι**

$$C(w,j) = \max\{C(w, j-1), C(w-w_j, j-1) + v_i\}$$

μπορέσαμε να εκφράσουμε το υποπρόβλημα **C(w, j)** συναρτήσει των μικροτέρων **C(•, j-1)**

- Τελειώσαμε ! ...
και σε αυτή την περίπτωση ο αλγόριθμος γράφετε πολύ εύκολα !!!

Σακίδιο : Χωρίς επανάληψη

Αλγόριθμος

```
Knap-sack2(w[1..n], v[1..n], W)
1.  Initialize()
2.  for j ← 1 to n
        for w ← 1 to W
            if w_j > w then
                C(w,j) ← C(w,j-1)
            else
                C(w,j) ← max{C(w, j-1), C(w-w_j, j-1) + v_j}
3.  return C(W,n)
```

```
for w ← 0 to W
        C(w,0) = 0
for j ← 0 to n
        C(0,j) = 0
```

Παράδειγμα

Είδος	Βάρος	Τιμή
1 : Α	6	30 €
2 : Β	3	14 €
3 : Γ	4	16 €
4 : Δ	2	9 €

Στο παράδειγμά μας όπου έχουμε **W = 10** και **4** είδη **{Α, Β, Γ, Δ}**, οι λύσεις όλων των υποπροβλημάτων **C(i,j)** σχηματίζουν έναν **11×5** πίνακα.

ΛΥΣΗ : E(10,4) = 46

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	9
3	0	0	14	14	14
4	0	0	14	16	16
5	0	0	14	16	16
6	0	30	30	30	30
7	0	30	30	30	30
8	0	30	30	30	30
9	0	30	30	30	30
10	0	30	30	30	46

Σακίδιο : Χωρίς επανάληψη

Αλγόριθμος

```
Knap-sack2(w[1..n], v[1..n], W)
```

- Ο αλγόριθμος συμπληρώνει ένα πίνακα διαστάσεων $(W+1) \times (n+1)$

- Κάθε καταχώρηση στον πίνακα απαιτεί $O(1)$ χρόνο

Συνολική Πολυπλοκότητα

O(nW)

Παρατήρηση 1

Παρόλο που ο πίνακας αποθήκευσης των τιμών των υποπροβλημάτων είναι πολύ μεγαλύτερος από αυτόν του Αλγόριθμου `Knap-sack1`, ο χρόνος εκτέλεσης παραμένει ο ίδιος !!!

Σακίδιο : Χωρίς επανάληψη

- **Αλγόριθμος**

```
Knap-sack2(w[1..n], v[1..n], W)
```

Παρατήρηση 2

Η πολυπλοκότητα χρόνου O(nW) είναι *ψευδοπολυωνυμική*, καθώς εξαρτάται από την τιμή W, η οποία μπορεί να είναι εκθετικά εξαρτώμενη από το n !!!

Συνολική Πολυπλοκότητα

O(nW)

Το πρόβλημα Knap-sack ανήκει στην κλάση NPC – ήτοι, δεν είναι γνωστό (και το πιθανότερο είναι να παραμείνει έτσι) πως μπορεί να λυθεί σε πολυωνυμικό χρόνο !!!

5 Ελάχιστες Αξιόπιστες Διαδρομές

Κίνητρο

Σε εφαρμογές συμβαίνει συχνά τα βάρη των ακμών και οι ελάχιστες διαδρομές να **μην** απεικονίζουν **ολόκληρη την αλήθεια** !!!

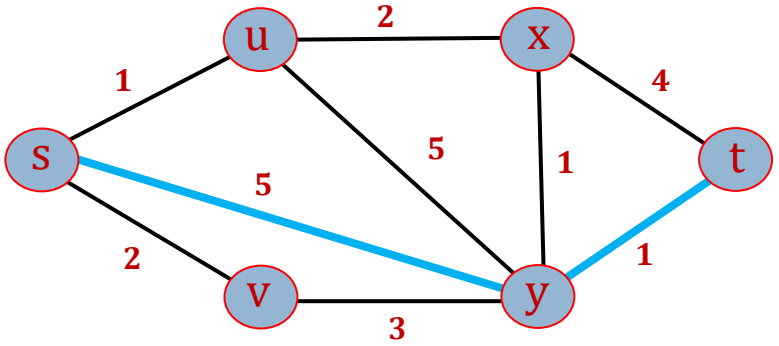
Για παράδειγμα, σε ένα δίκτυο επικοινωνιών, ακόμα και αν τα βάρη των ακμών απεικονίζουν πιστά τις καθυστερήσεις μετάδοσης, μπορεί να υπάρχουν άλλοι παράγοντες που εμπλέκονται σε μια ε.δ !!!

5 Ελάχιστες Αξιόπιστες Διαδρομές

Κίνητρο

Θα μπορούσε, για παράδειγμα, κάθε επιπλέον ακμή στην ε.δ να είναι ένας παράγοντας «αβεβαιότητας» που ελλοχεύει κίνδυνους απώλειας πακέτων!

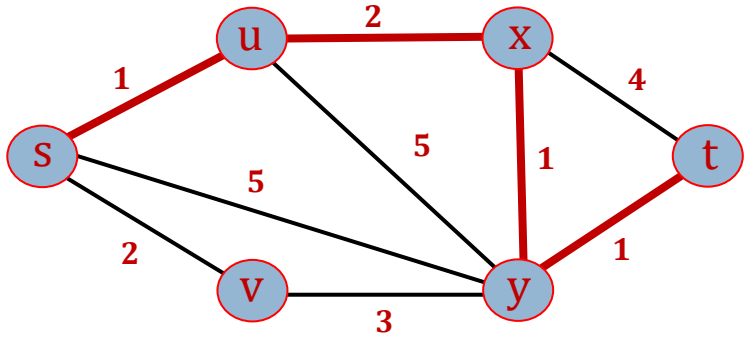
Τότε, θα θέλαμε να αποφύγουμε διαδρομές με πάρα πολλές ακμές!



5 Ελάχιστες Αξιόπιστες Διαδρομές

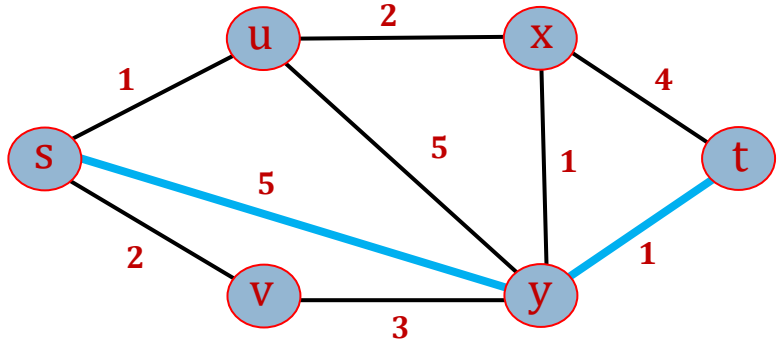
d(s,t)=(s,u,x,y,t)

Κόστος 5
Πλήθος ακμών 4



δ(s,t)=(s,y,t)

Κόστος 6
Πλήθος ακμών 2



5 Ελάχιστες Αξιόπιστες Διαδρομές

Πρόβλημα

Έστω ένα έμβαρο γράφημα G, δύο κόμβοι του **s** και **t**, και ένας ακέραιος ***k***.

Θέλουμε να υπολογίσουμε μια ε.δ στο G

s ~> t

*η οποία να χρησιμοποιεί το πολύ **k** ακμές* !!!

Ελάχιστες Αξιόπιστες Διαδρομές

- Υπάρχει κάποιος αποτελεσματικός τρόπος να προσαρμόσουμε τον αλγόριθμο του Dijkstra στο νέο μας πρόβλημα ?

- Μάλλον **Όχι** !!!

  Ο αλγόριθμος του Dijkstra εστιάζει στο μήκος κάθε ε.δ χωρίς να «θυμάται» το πλήθος των ακμών στη διαδρομή !

  Ακόμα και να το θυμόταν, δεν θα μπορούσε να επιλέξει την ελάχιστη με το πολύ k ακμές !!!

Ελάχιστες Αξιόπιστες Διαδρομές

- Μήπως ο Δυναμικός Προγραμματισμός μας λύνει το πρόβλημα ?
- Ας ορίσουμε για κάθε κόμβο **v** και κάθε ακέραιο $\mathbf{i \leq k}$

$$\mathbf{d(s,v,i) = d(v,i)}$$

να είναι το κόστος της ελάχιστης διαδρομής **s ~> v** η οποία να *χρησιμοποιεί το πολύ* **i** *ακμές*

Προφανώς, ισχύει:

$$\mathbf{d(s,0) = 0}$$
$$\mathbf{d(v,0) = \infty \quad \forall\ v \in V}$$

Ελάχιστες Αξιόπιστες Διαδρομές

- Πολύ φυσικά, η γενική εξίσωση ενημέρωσης, είναι:

$$\mathbf{d(v,i)} = \min\{\mathbf{d(v,i-1)}, \min_{(u,v)\in E}\{\mathbf{d(u,i-1)} + \mathbf{w(u,v)}\}\}$$

- **Αυτό ήταν… Τελειώσαμε !!!**

  Ο αλγόριθμος πλέον γράφετε πολύ εύκολα, σχεδόν μόνος του !!!

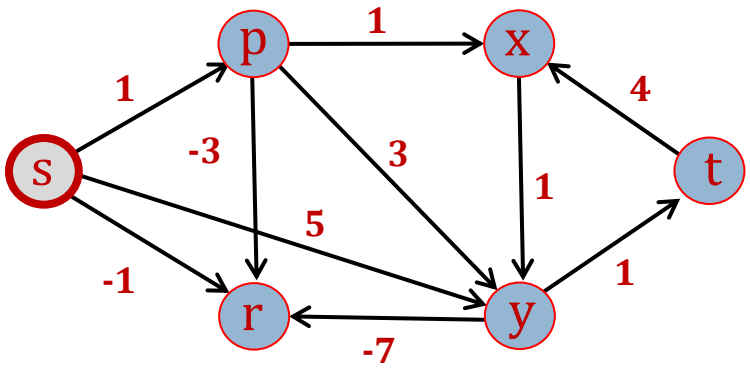
Αλγόριθμος

```
K-shortest-path(G,w, k)
1. Initialize()
2. for i ← 1 to k
       for each v ∈ V-{s}
           for each u ∈ Adj(v)
               d(v,i) ← min{C(v, i-1),C(u, i-1)+w(u,v)}
3. return C(k)
```

```
for i ← 1 to n
    C(i,0)=∞
C(s,0)=0
```

Ελάχιστες Αξιόπιστες Διαδρομές

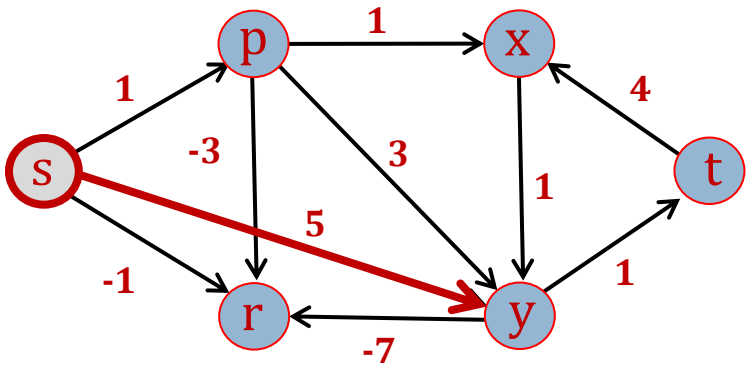
Παράδειγμα



	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞

Ελάχιστες Αξιόπιστες Διαδρομές

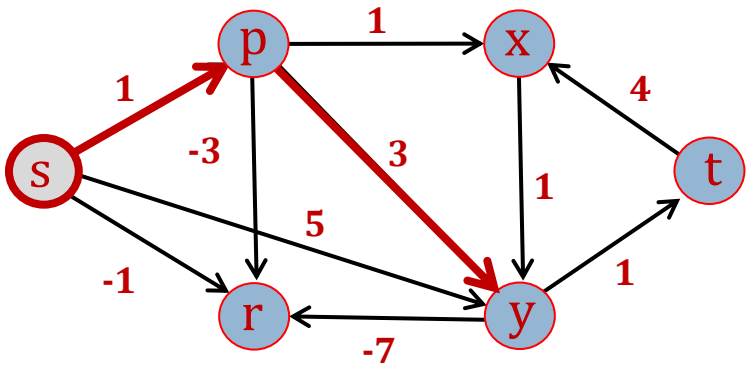
Παράδειγμα



	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞

Ελάχιστες Αξιόπιστες Διαδρομές

Παράδειγμα



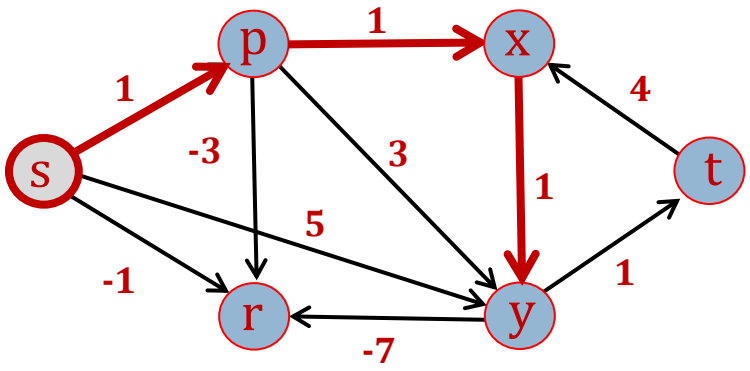
	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞
d(v,2) =	0	1	-2	2	4	6

$$d(y,2) = \min\{d(y,1), \min_{(u,y)\in E}\{d(u,1)+w(u,y)\}\}$$

$$= \min\{5, \min\{0+5,\ 1+3,\ \infty+1\}\} = \mathbf{4}$$

Ελάχιστες Αξιόπιστες Διαδρομές

Παράδειγμα

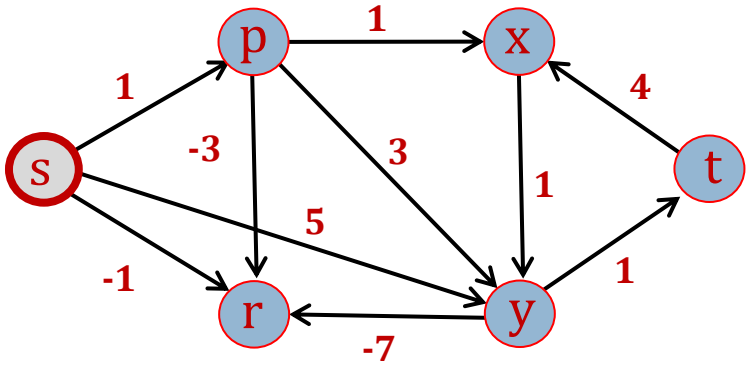


	s	p	r	x	y	t
d(v,0) =	0	∞	∞	∞	∞	∞
d(v,1) =	0	1	-1	∞	5	∞
d(v,2) =	0	1	-2	2	4	6
d(v,3) =	0	1	-3	2	3	6

d(y,3) = min{4, min{0+5, 1+3, 2+1}} = **3**

Ελάχιστες Αξιόπιστες Διαδρομές

Παράδειγμα



	s	p	r	x	y	t
d(s,v,0)	0	∞	∞	∞	∞	∞
d(s,v,1)	0	1	-1	∞	5	∞
d(s,v,2)	0	1	-2	2	4	6
d(s,v,3)	0	1	-3	2	3	6
d(s,v,4)	0	1	-4	2	3	4

6 Πανζευκτικές Ελάχιστες Διαδρομές

Κίνητρο

Σε πολλές εφαρμογές, χρειαζόμαστε να έχουμε όχι μόνο την ε.δ μεταξύ **s ~> t** ενός G τάξης n, αλλά μεταξύ κάθε ζεύγους κόμβων του!!!

Μπορούμε να λύσουμε το πρόβλημα ?

Φυσικά ΝΑΙ !!!
Εφαρμόζοντας n φορές τον αλγόριθμο των Bellman-Ford (επιτρέπονται αρνητικά βάρη)

6 Πανζευκτικές Ελάχιστες Διαδρομές

Σε πόσο χρόνο ?

Ο Bellman-Ford εκτελείται σε O(nm) χρόνο, και επομένως:

$$\mathbf{O(n^2m)}$$

Μπορούμε καλύτερα ?

ΝΑΙ !!! με τον αλγόριθμο των **Floyd-Watshall**

ο οποίος βασίζεται σε ΔΠ, σε χρόνο: $\mathbf{O(n^3)}$

6 Πανζευκτικές Ελάχιστες Διαδρομές

Πρόβλημα

Έστω ένα έμβαρο γράφημα G (με αρνητικά ακμικά βάρη) τάξης n και μεγέθους m.

Θέλουμε να υπολογίσουμε τις ε.δ στο G

$$\mathbf{x} \rightsquigarrow \mathbf{y}$$

για κάθε ζεύγος κόμβων (x, y), $x, y \in V(G)$!!!

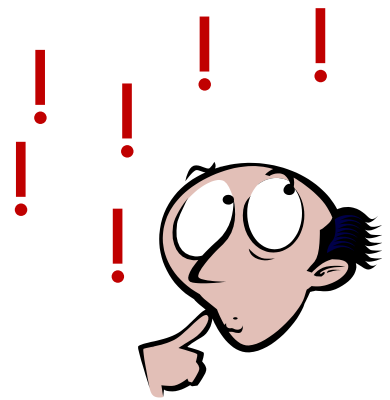
Πανζευκτικές Ελάχιστες Διαδρομές

- Το να λύσουμε το πρόβλημα υπολογίζοντας όλο και περισσότερα ζεύγη κόμβων δε βοηθά, καθώς οδηγεί πίσω στον αλγόριθμο **$O(n^2m)$** !!!

- **Κρίσιμο Ερώτημα ΔΠ**

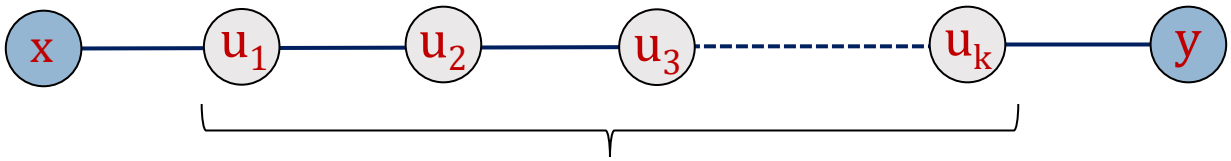
  Υπάρχει κάποιο καλό υποπρόβλημα για τον υπολογισμό των αποστάσεων μεταξύ κάθε ζεύγους κόμβων ?

Πρέπει
να επινοήσουμε κάτι έξυπνο !
να βρούμε μια νέα ιδέα !



- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ $x \rightsquigarrow y$



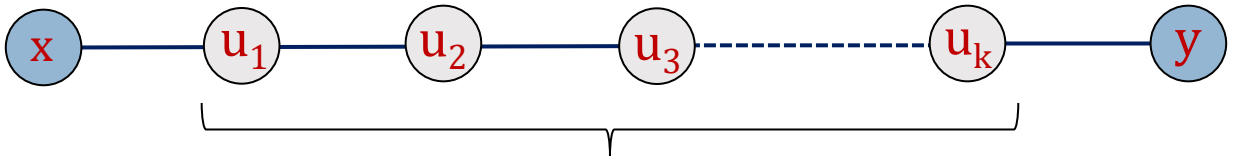
- Υποθέστε ότι δεν επιτρέπουμε ενδιάμεσους κόμβους !!!

  Τότε, μπορούμε να λύσουμε το πρόβλημα των π.ε.δ, καθώς για κάθε ζεύγος κόμβων x, y του G:

  η ε.δ $x \rightsquigarrow y$ είναι η ακμή (x, y), εάν υπάρχει !!!

- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ $x \rightsquigarrow y$

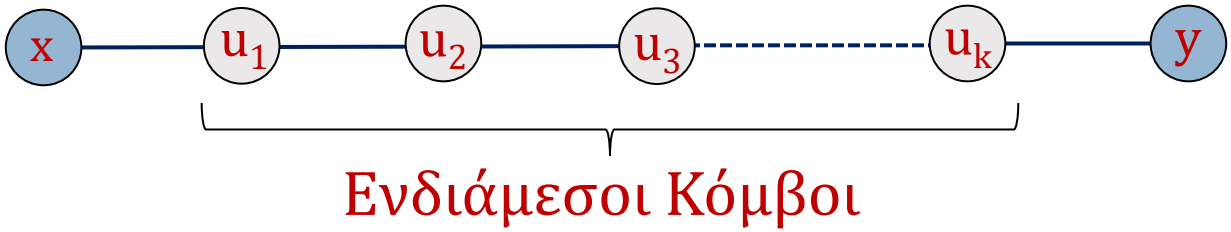


- Εάν διευρύνουμε σταδιακά το **σύνολο των ενδιάμεσων κόμβων**, **έναν κάθε φορά**, **ενημερώνοντας τα μήκη των ε.δ** σε κάθε διεύρυνση,

  τι μπορούμε να πετύχουμε !!!

- **Να μια ιδέα !!!**

  Ας πάρουμε μια ε.δ x ~> y



- Μπορούμε να διευρύνουμε το σύνολο των ενδιάμεσων κόμβων **έως να γίνει όλο το σύνολο κόμβων V !!!**

Τότε όμως έχουμε λύσει το πρόβλημα των π.ε.δ !!!

- **Τυποποίηση της ιδέας !!!**
  - Αριθμούμε τους κόμβους του **V** ως **{1,2, …,n}**
  - d(i,j,k) συμβολίζουμε το μήκος της ε.δ i ~> j

    με ενδιάμεσους κόμβους **MONO** από το σύνολο **{1,2, …,k}**
  - Επομένως,

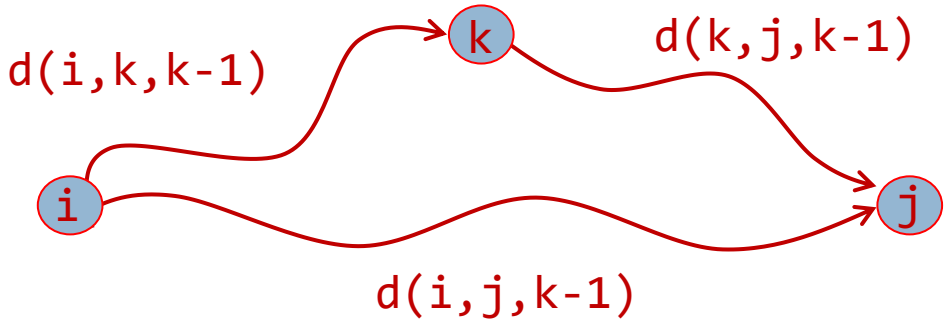
    d(i,j,0) = w(i,j) εάν υπάρχει η ακμή (i,j)

Πανζευκτικές Ελάχιστες Διαδρομές

- Τι πρέπει να ελέγξουμε όταν επεκτείνουμε το ενδιάμεσο σύνολο κατά ένα κόμβο ?

$$\{1,2,\ldots,k-1\} \rightarrow \{1,2,\ldots,k\}$$

  - Πρέπει να **επανεξετάσουμε** όλα τα ζεύγη κόμβων **i, j** και να **ελέγξουμε** εάν η χρήση του **k** ως ενδιάμεσου κόμβου μας δίνει μια συντομότερη διαδρομή **i ~> j**

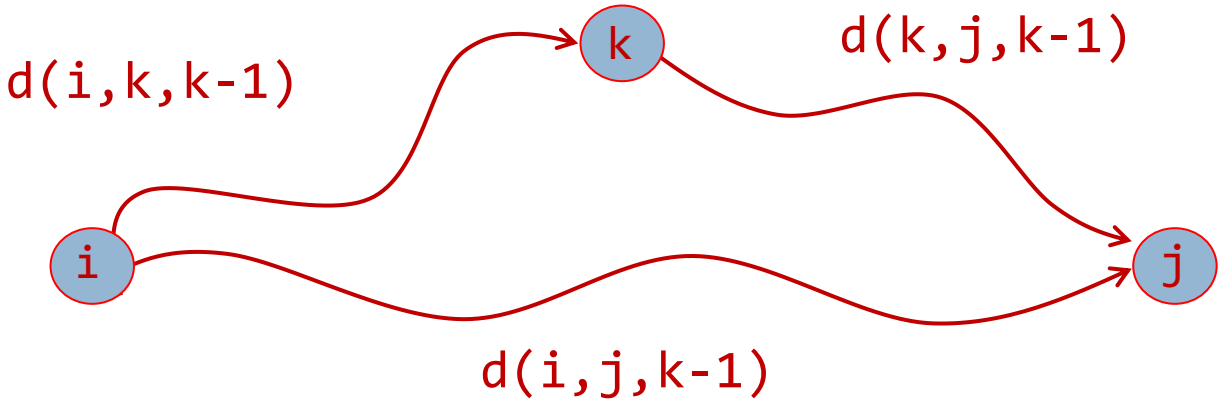


αυτό είναι πολύ εύκολο !!!

Πανζευκτικές Ελάχιστες Διαδρομές

- Μια ε.δ `i ~> j` η οποία χρησιμοποιεί τον `k` μαζί με άλλους κόμβους χαμηλότερης αρίθμησης `{1,2, …,k-1}`, περνάει από τον `k` **μία μόνο φορά** !!!

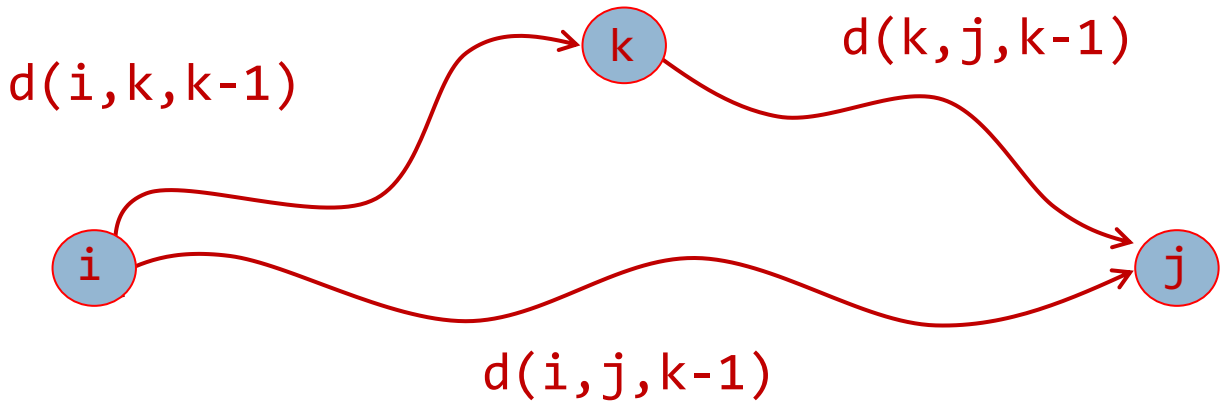
γιατί δεν υπάρχουν αρνητικοί κύκλοι



Πανζευκτικές Ελάχιστες Διαδρομές

- Έτσι, η χρήση του **k** ως ενδιάμεσου κόμβου μας δίνει μια συντομότερη διαδρομή **i ~> j** εάν και μόνο εάν:

```
d(i,k,k-1) + d(k,j,k-1) < d(i,j,k-1)
```



οπότε η τιμή:

`d(i,j,k)` θα πρέπει να ενημερωθεί ανάλογα !!!

Σχέση Επίλυσης Υποπροβλημάτων

- `d(i,j,0) = w(i,j)` εάν υπάρχει η ακμή `(i,j)`
- Εξίσωση ενημέρωσης (ή αναδρομική σχέση) :

$$d(i,j,k) = \begin{cases} w(i,j) & \text{if } k=0 \\ \min\{d(i,j,k-1),\ d(i,k,k-1) + d(k,j,k-1)\} & \text{if } k \geq 1 \end{cases}$$

Αλγόριθμος Floyd-Warshall

```
All-pair-Shortest-Paths(G,w)
1. Initialize(G)
2. for k ← 1 to n
       for i ← 1 to n
           for j ← 1 to k
               d(i,j,k) ← min{d(i,j,k-1),
                             d(i,k,k-1) + d(k,j,k-1)}
3. return d(i,j,n)
```

```
for i ← 1 to n
        for j ← 1 to n
                C(i,i)=0
for all (i,j)∈E
        d(i,j,0)=w(i,j)
```

Πολυπλοκότητα

Το πιο χρονοβόρο βήμα:

```
2. for k ← 1 to n
       for i ← 1 to n
           for j ← 1 to k
               d(i,j,k) ← min {d(i,j,k-1),
                               d(i,k,k-1) + d(k,j,k-1)}
```

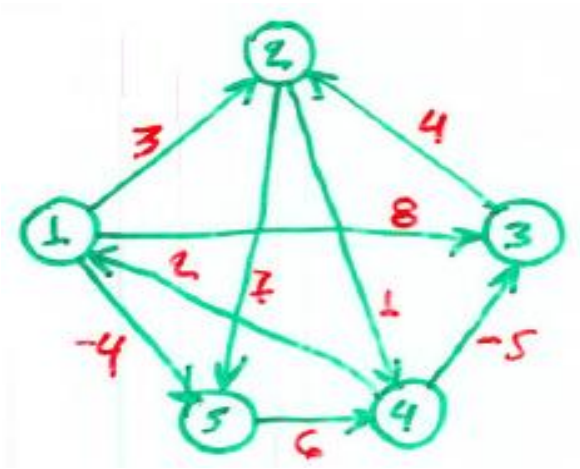
Συνολική Πολυπλοκότητα Αλγόριθμου: $O(n^3)$

Πανζευκτικές Ελάχιστες Διαδρομές

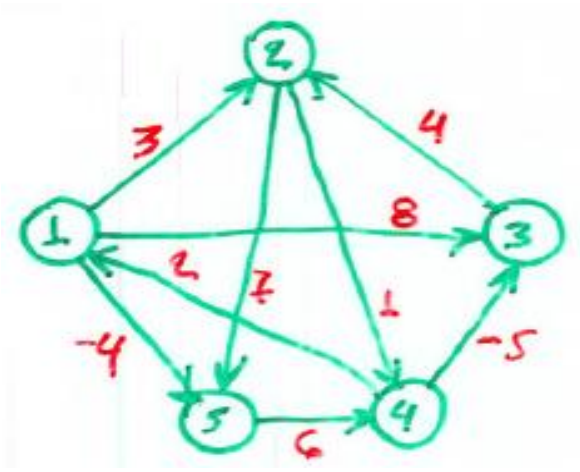
Παράδειγμα

d(i,j,0)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



Παράδειγμα



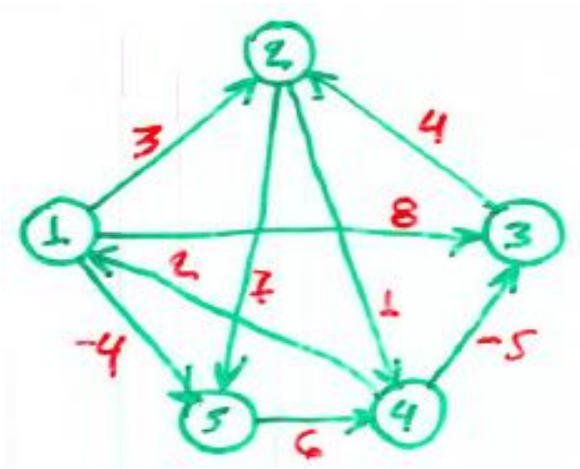
d(i,j,1)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$$d(4,2,1) = \min\{d(4,2,0), d(4,1,0)+d(1,2,0)\}$$
$$= \min\{\infty, 2+3\} = \mathbf{5}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

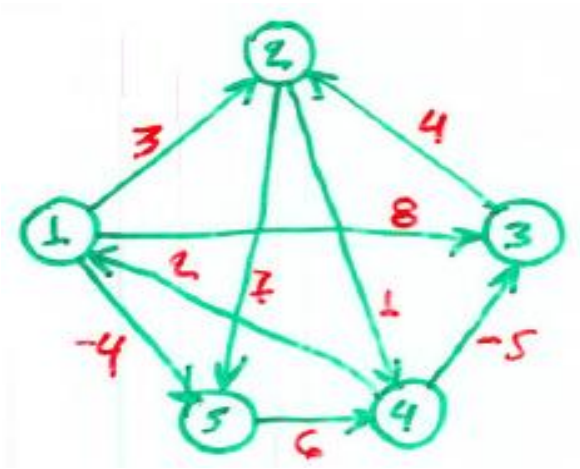


$d(i,j,2)$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$$d(4,2,2) = \min\{d(4,2,1), d(4,2,1)+d(2,2,1)\}$$
$$= \min\{5, 5+0\} = \mathbf{5}$$

Παράδειγμα



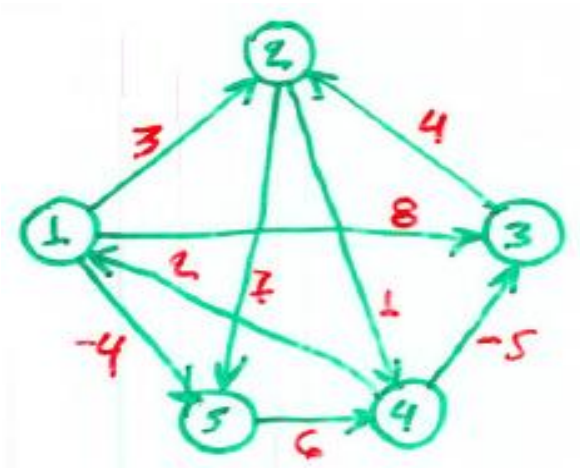
d(i,j,2)

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$$d(3,4,2) = \min\{d(3,4,1), d(3,2,1)+d(2,4,1)\}$$
$$= \min\{\infty, 4+1\} = \mathbf{5}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα



$d(i,j,2)$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

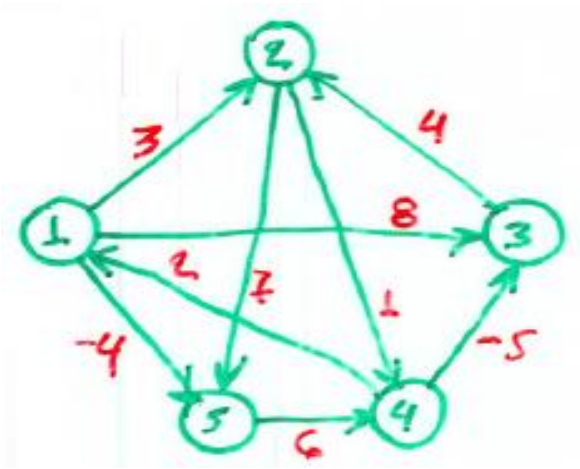
$$d(3,5,2) = \min\{d(3,5,1), d(3,2,1)+d(2,5,1)\}$$
$$= \min\{\infty, 4+7\} = \mathbf{11}$$

Πανζευκτικές Ελάχιστες Διαδρομές

Παράδειγμα

d(i,j,5)

	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

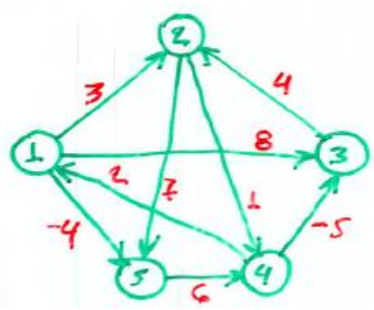


Ποιες Αλγοριθμικές Τεχνικές έχουμε δει έως τώρα ?

- Αλγοριθμικές Τεχνικές, όπως:
  - ✓ Διαίρει-και-Βασίλευε
  - ✓ Άπληστη Επιλογή
- Πρόσφατα, προσθέσαμε και την τεχνική
  - ✓ **Δυναμικός Προγραμματισμός**

S	U	N	N	-	Y
S	–	N	O	W	Y

A	50 × 20
B	20 × 1
C	1 × 10
D	10 ×100



Αλγοριθμικές Τεχνικές & Προβλήματα

Τι είδους Προβλήματα έχουμε δει έως τώρα ?

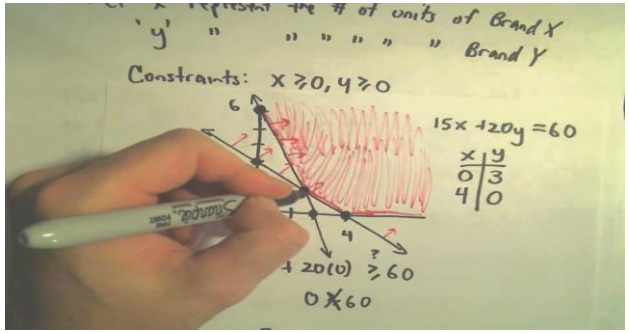
- Προβλήματα των οποίων η λύση απαιτεί Αλγορίθμους για εργασίες βελτιστοποίησης, όπως:
  - ✓ Ελάχιστες διαδρομές
  - ✓ Ελάχιστα συνδετικά (επικαλυπτικά) δένδρα
  - ✓ Μέγιστες αύξουσες υπακολουθίες, κλπ

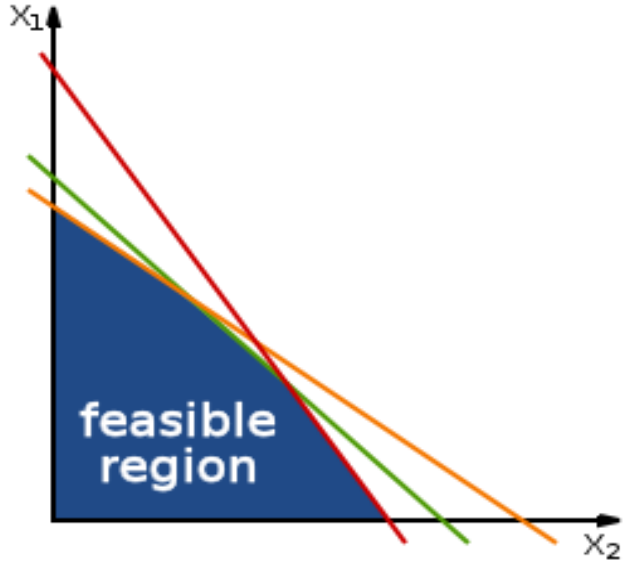
Ποιο το χαρακτηριστικό της λύσης του ?

- Αναζητούμε λύση η οποία:
  - (α) **Ικανοποιεί** ορισμένους **περιορισμούς**, και
  - (β) **Είναι η καλύτερη δυνατή** ως προς κάποιο **κριτήριο** !!!

- **Σημαντικό Εργαλείο Τέχνης για τέτοια προβλήματα Βελτιστοποίησης**

Γραμμικός Προγραμματισμός - LP





- **Σημαντικό Εργαλείο Τέχνης για τέτοια προβλήματα Βελτιστοποίησης**

Γραμμικός Προγραμματισμός - LP

- **Κύριο χαρακτηριστικό του LP**
  - **Περιγράφει εργασίες βελτιστοποίησης στις οποίες:**
    - ✓ Οι **Περιορισμοί** και το **Κριτήριο βελτιστοποίησης** είναι **Γραμμικές Συναρτήσεις** !!!

Παράδειγμα

- Βρες τις τιμές των μεταβλητών x_1 και x_2 οι οποίες

max $x_1 + x_2$

με τους περιορισμούς

$x_1 + 2x_2 \leq 4$

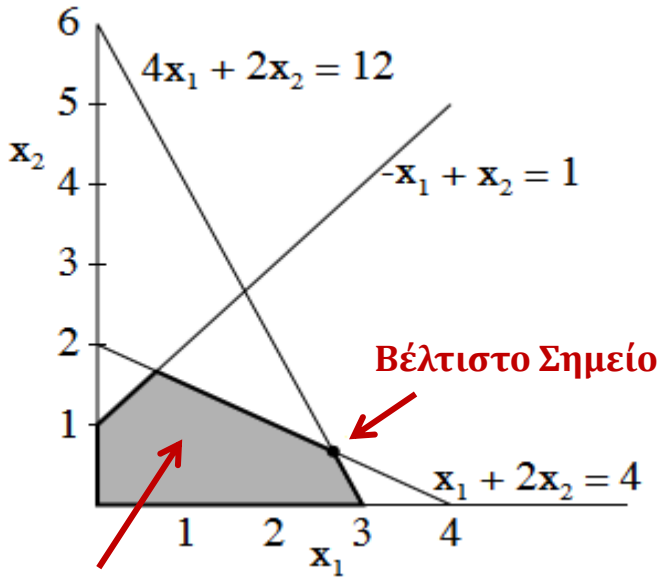
$4x_1 + 2x_2 \leq 12$

$-x_1 + x_2 \leq 1$

$x_1, x_2 \geq 0$

Περιορισμοί

Αντικειμενική Συνάρτηση



Σύνολο Περιορισμών ή Εφικτών Λύσεων

Γενικά, επειδή η ΑΣ είναι γραμμική, παίρνει max (ή min) σε γωνιακό σημείο του ΣΠ, εάν αυτό είναι φραγμένο !!!

1 Μεγιστοποίηση Κέρδους

Ένα εργοστάσιο παράγει δύο προϊόντα: **A** και **B**

Ανά Μονάδα πώλησης: από το **A** κερδίζει **1** €, και
από το **B** κερδίζει **6** €

Οι ημερήσιες απαιτήσεις: για το **A** είναι **200** Μ, και
για το **B** είναι **300** Μ το πολύ

Επίσης, το εργοστάσιο μπορεί να παράγει συνολικά **400** Μ/ημέρα !

Πόση πρέπει να είναι η ημερήσια παραγωγή από κάθε προϊόν A και B, ώστε να επιτευχθεί το max κέρδος ?

Περιγράφουμε το πρόβλημά μας με χρήση LP

- Εάν x_1 και x_2 είναι οι Μονάδες Ημερήσιας Παραγωγής των προϊόντων A και B, αντίστοιχα, τότε

  Βρες τις τιμές των x_1 και x_2 οι οποίες

$$\max\ x_1 + 6x_2$$

  με τους περιορισμούς

$$x_1 \le 200$$

$$x_2 \le 300$$

$$x_1 + x_2 \le 400$$

$$x_1,\ x_2 \ge 0$$

από **A** κέρδος **1** €
από **B** κέρδος **6** €

για **A** απαιτήσεις ≤ **200** Μ/ημ.
για **B** απαιτήσεις ≤ **300** Μ/ημ.

δυνατότητα παραγωγής
συνολικά **400** Μ/ημέρα !

Περιγράφουμε το πρόβλημά μας με χρήση LP

- Εάν x_1 και x_2 είναι οι Μονάδες Ημερήσιας Παραγωγής των προϊόντων A και B, αντίστοιχα, τότε

  Βρες τις τιμές των x_1 και x_2 οι οποίες

$$\max \; x_1 + 6x_2$$

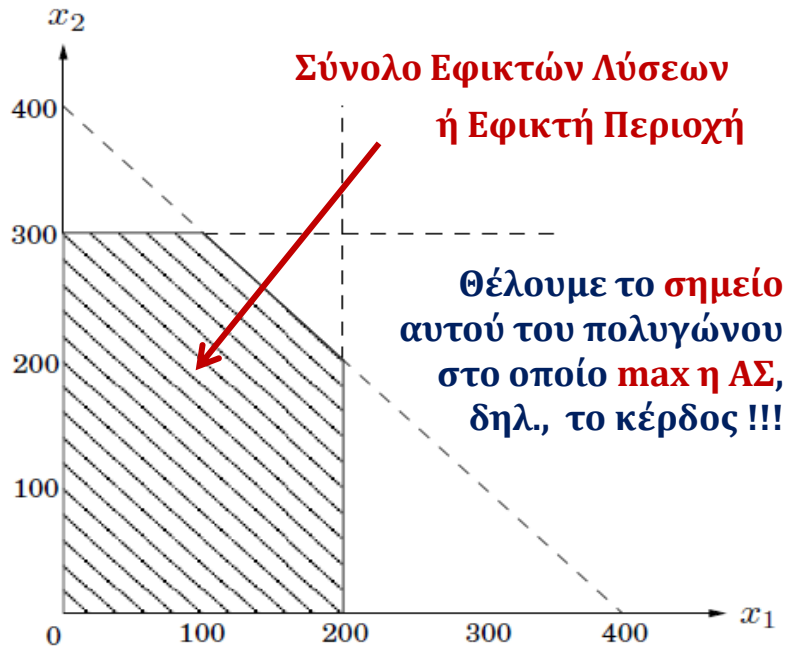
  με τους περιορισμούς

$$x_1 \leq 200$$

$$x_2 \leq 300$$

$$x_1 + x_2 \leq 400$$

$$x_1, \; x_2 \geq 0$$



Περιγράφουμε το πρόβλημά μας με χρήση LP

- Εάν x_1 και x_2 είναι οι Μονάδες Ημερήσιας Παραγωγής των προϊόντων A και B, αντίστοιχα, τότε

  Βρες τις τιμές των x_1 και x_2 οι οποίες

$$\max\ x_1 + 6x_2$$

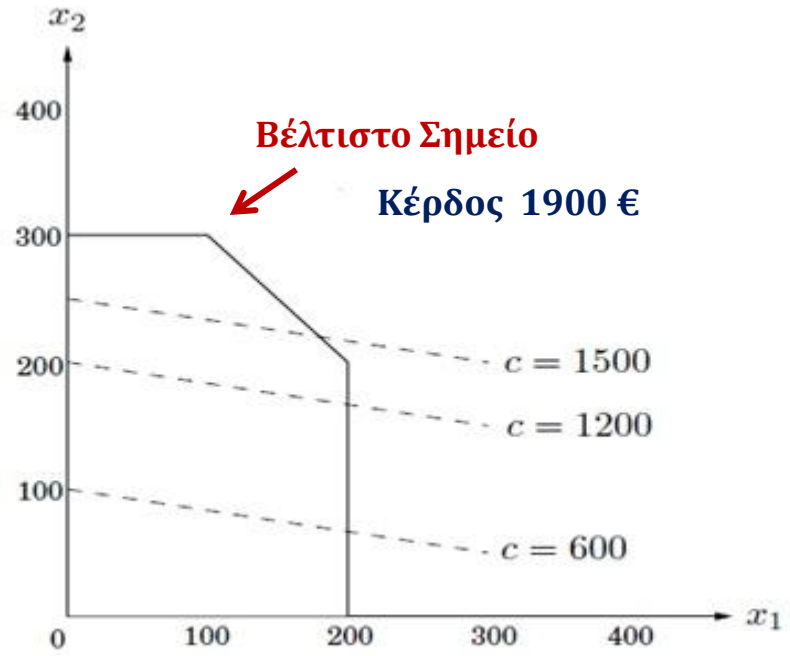
με τους περιορισμούς

$$x_1 \leq 200$$

$$x_2 \leq 300$$

$$x_1 + x_2 \leq 400$$

$$x_1,\ x_2 \geq 0$$



Χαρακτηριστικά της λύσης

- Αποτελεί γενικό κανόνα των Γραμμικών Προγραμμάτων ότι το βέλτιστο επιτυγχάνεται σε μια κορυφή της εφικτής περιοχής !!!

Εκτός, εάν δεν υπάρχει βέλτιστο !!!

Αυτό μπορεί να συμβεί με 2 τρόπους:

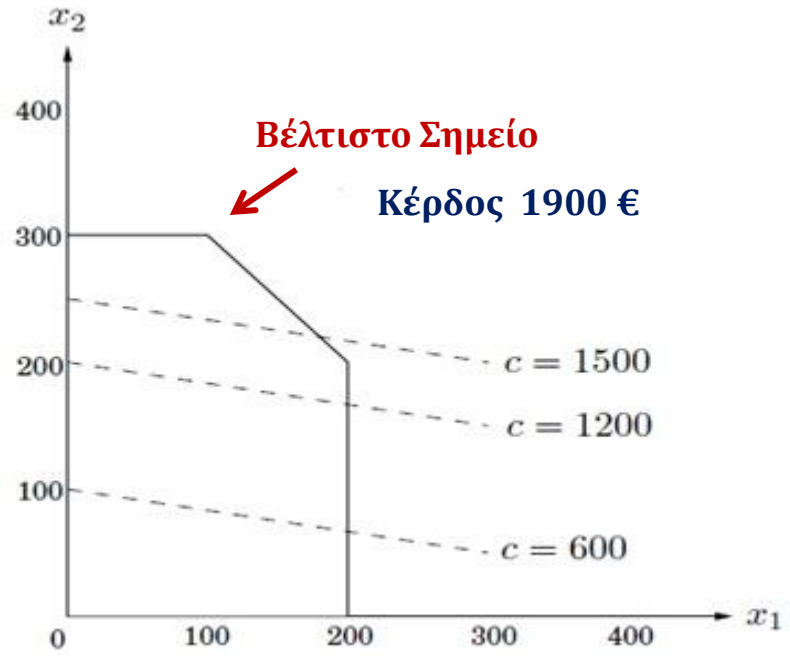
(1) Το ΓΠ είναι μη εφικτό (infeasible)!
Για παράδειγμα:

$$x \leq 1, \quad x \geq 2$$

(2) Οι περιορισμοί είναι τέτοιοι ώστε η εφικτή περιοχή να είναι μη φραγμένη!
Για παράδειγμα:

$$\max \; x_1 + x_2$$

$$x_1 + x_2 \geq 0$$

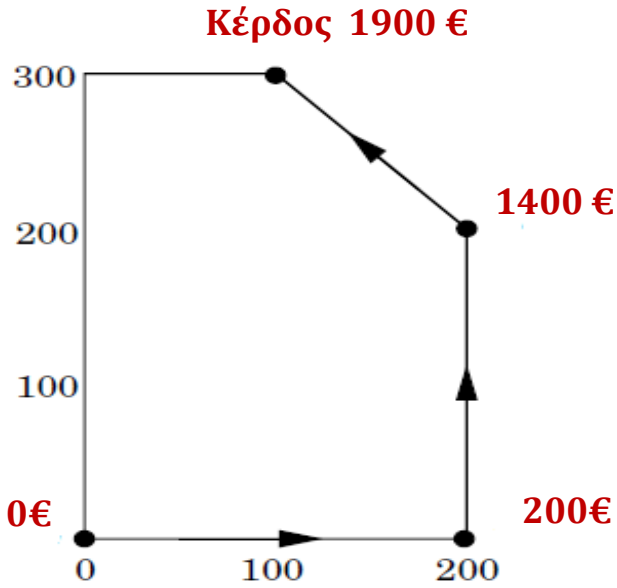


Επίλυση Γραμμικών Προγραμμάτων

- Με την μέθοδο simplex (George Dantzig, 1947) !!!

Βασική Ιδέα Αλγόριθμου !!!

- ✓ Ξεκινά από μία κορυφή, στην περίπτωσή μας ίσως από την (0,0)
- ✓ Επαναληπτικά αναζητά μια γειτονική κορυφή με καλύτερη αντικειμενική τιμή
- ✓ Όταν φθάσει σε μια κορυφή η οποία δεν έχει καλύτερο γείτονα, ο αλγόριθμος simplex την δηλώνει ως βέλτιστη και σταματά !



Περισσότερα Προϊόντα

- Η εταιρεία, βλέποντας την ανταπόκριση των καταναλωτών, αποφασίζει να εισάγει τρίτο προϊόν **Γ**, από το οποίο κερδίζει **13** € ανά Μονάδα !!!
- Οι δυνατότητες παραγωγής της εταιρίας παραμένουν ίδιες, δηλ., συνολικά **400** Μ/ημέρα !
- Τα προϊόντα **Γ** και **Β** θα παράγονται στο ίδιο μηχάνημα ικανότητας λειτουργίας **10** ώρες/ημ. και το **Γ** απαιτεί 3-πλάσιο χρόνο από το **Β** !

από **Α** κέρδος **1** €
από **Β** κέρδος **6** €

για **Α** απαιτήσεις ≤ **200** Μ/ημ.
για **Β** απαιτήσεις ≤ **300** Μ/ημ.

δυνατότητα παραγωγής
συνολικά **400** Μ/ημέρα !

Ποια είναι τα καλύτερα δυνατά επίπεδα παραγωγής, ώστε να επιτευχθεί το max κέρδος ?

Το νέο Γραμμικό Πρόγραμμα είναι το ακόλουθο:

- Εάν x_1, x_2 και x_3 είναι οι Μονάδες Ημερήσιας Παραγωγής των προϊόντων **Α**, **Β** και **Γ**, αντίστοιχα, τότε:

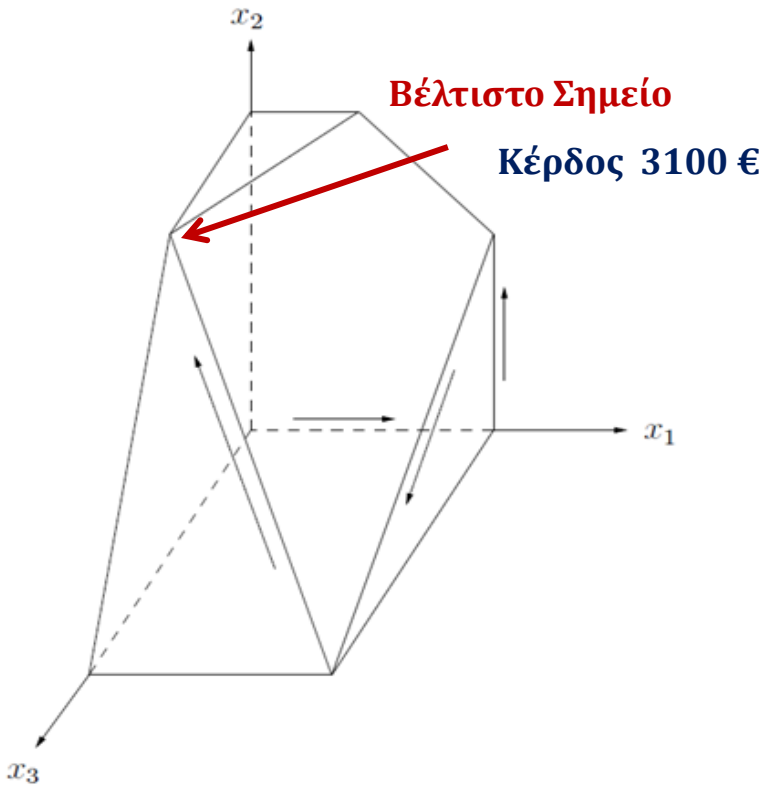
  Βρες τις τιμές των x_1, x_2 και x_3 οι οποίες

  $$\max\ x_1 + 6x_2 + 13x_3$$

  με τους περιορισμούς

  $$x_1 \le 200$$
  $$x_2 \le 300$$
  $$x_1 + x_2 + x_3 \le 400$$
  $$x_2 + 3x_3 \le 600$$
  $$x_1,\ x_2,\ x_3 \ge 0$$

Η μηχανή των Β και Γ εργάζεται 10 ώρες/ημ. (600 λεπτά) και έστω ότι το Β απαιτεί 1 λεπτό και το Γ 3 λεπτά !!!



Σκιαγράφηση της λύσης του νέου Προγράμματος:

- Συμπεριφορά του Αλγόριθμου Simplex

Βρες τις τιμές των x_1, x_2 και x_3 οι οποίες

$$\max\ x_1 + 6x_2 + 13x_3$$

με τους περιορισμούς

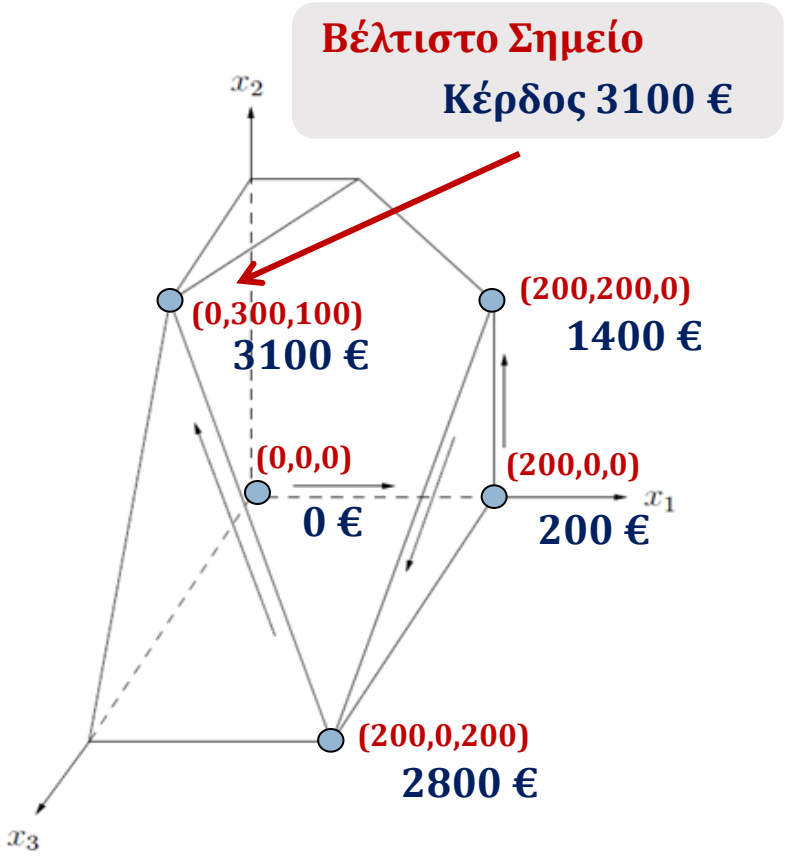
$$x_1 \le 200$$

$$x_2 \le 300$$

$$x_1 + x_2 + x_3 \le 400$$

$$x_2 + 3x_3 \le 600$$

$$x_1,\ x_2,\ x_3 \ge 0$$

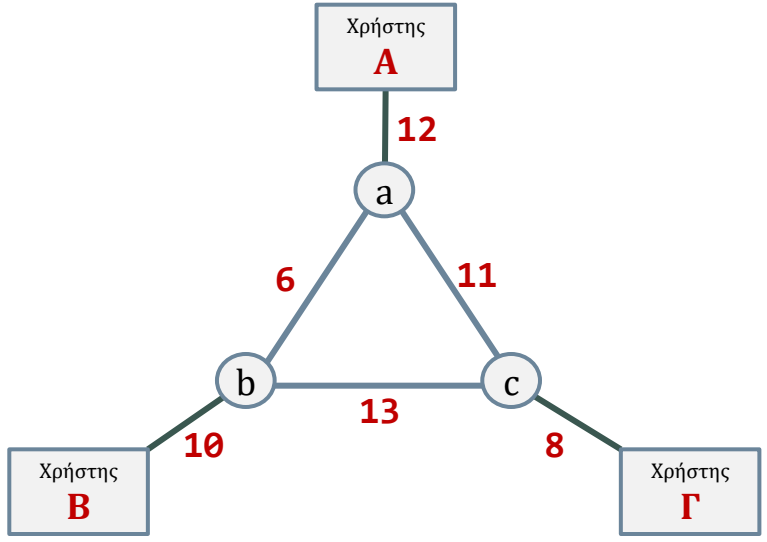


2 Βέλτιστη Κατανομή Εύρους Ζώνης

Έστω ένας πάροχος δικτυακών υπηρεσιών, ο οποίος διαχειρίζεται ένα δίκτυο, θέλει να δημιουργήσει τρεις συνδέσεις μεταξύ των χρηστών :

Α-Β, **Β-Γ** και **Α-Γ**

Οι γραμμές του δικτύου έχουν εύρος ζώνης ακεραίους αριθμούς που φαίνονται διπλανό στο σχήμα

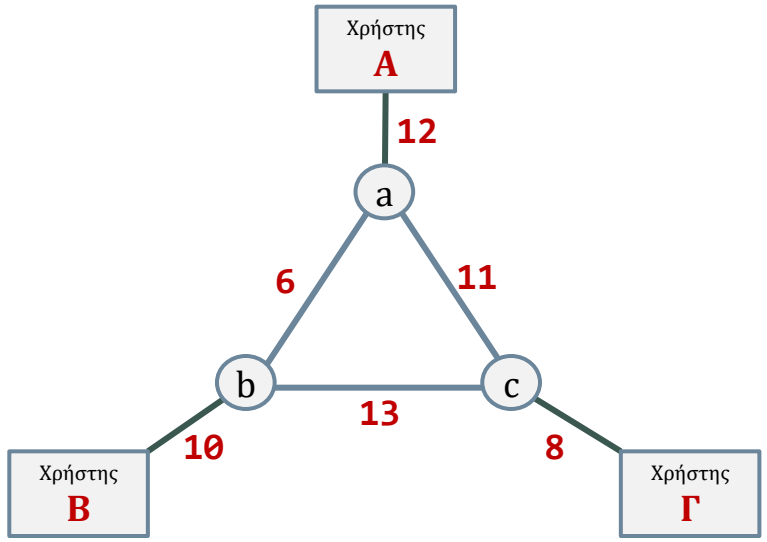


2 Βέλτιστη Κατανομή Εύρους Ζώνης

Κάθε σύνδεση απαιτεί τουλάχιστον **2 μονάδες** εύρους ζώνης στο δίκτυο !!!

Ανά μονάδα εύρους ζώνης, η σύνδεση:

- ✓ **Α-Β** πληρώνει **3 €**
- ✓ **Β-Γ** πληρώνει **2 €**
- ✓ **Α-Γ** πληρώνει **4 €**



2 Βέλτιστη Κατανομή Εύρους Ζώνης

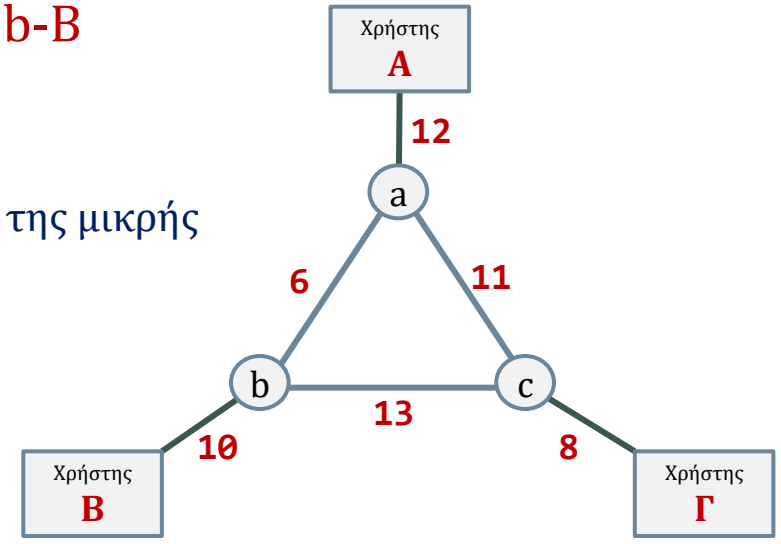
Κάθε σύνδεση μπορεί να δρομολογηθεί με 2 τρόπους:

- ✓ Με **μικρή** διαδρομή A-a-b-B
- ✓ Με **μεγάλη** διαδρομή A-a-c-b-B

Ή με κάποιο συνδυασμό !!!

Για παράδειγμα, 2 μονάδες εύρους ζώνης μέσω της μικρής και 1 μέσω της μεγάλης διαδρομής

Πως θα δρομολογηθούν αυτές οι συνδέσεις για να μεγισ/θούν (max) τα έσοδα του παρόχου !!!



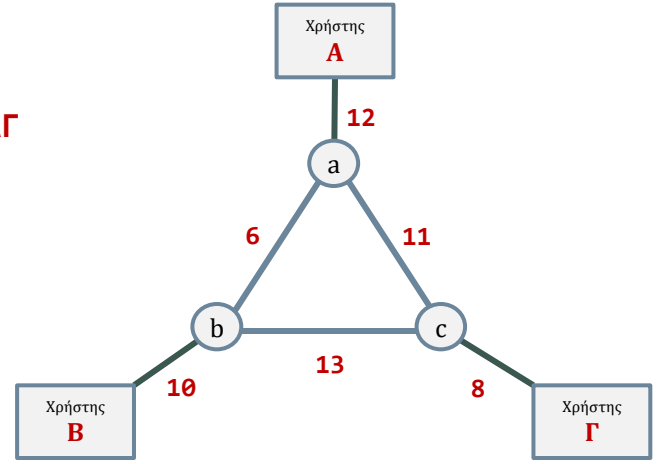
Αυτό είναι ένα LP… ας το περιγράψουμε

- Έστω x_{AB} και x'_{AB} είναι το εύρος ζώνης της μικρής και της μεγάλης διαδρομής, αντίστοιχα, που έχει κατανεμηθεί στη σύνδεση **A-B** (αντίστοιχα, για τις συνδέσεις **B-Γ** και **A-Γ**).

- $\max\ 3x_{AB} + 3x'_{AB} + 2x_{B\Gamma} + 2x'_{B\Gamma} + 4x_{A\Gamma} + 4x'_{A\Gamma}$

μ.τ.π

$$
\begin{aligned}
x_{AB} + x'_{AB} + x_{B\Gamma} + x'_{B\Gamma} &\le 10\\
x_{AB} + x'_{AB} + x_{A\Gamma} + x'_{A\Gamma} &\le 12\\
x_{B\Gamma} + x'_{B\Gamma} + x_{A\Gamma} + x'_{A\Gamma} &\le 8\\
x_{AB} + x'_{B\Gamma} + x'_{A\Gamma} &\le 6\\
&\dots\\
x_{A\Gamma} + x'_{A\Gamma} &\ge 2\\
x_{AB},\ x'_{AB},\ x_{B\Gamma},\ x'_{B\Gamma},\ x_{A\Gamma},\ x'_{A\Gamma} &\ge 0
\end{aligned}
$$



A-B πληρώνει **3 €**
B-Γ πληρώνει **2 €**
A-Γ πληρώνει **4 €**

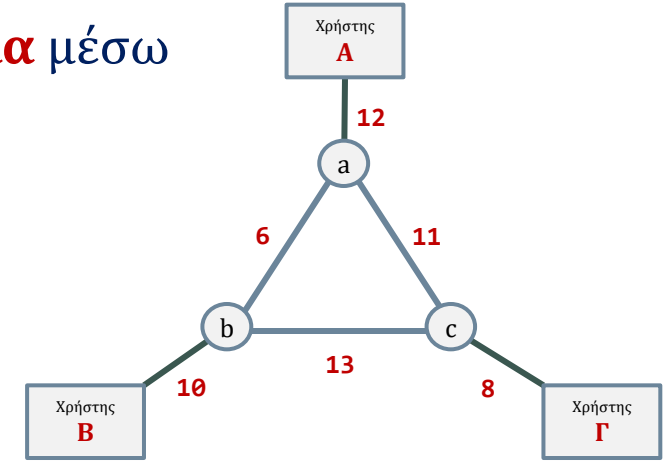
Λύση μέσω του Αλγορίθμου Simplex

- Αν και το πρόβλημά μας είναι μια πολύ μικρή μικρογραφία του πραγματικού, είναι δύσκολο να λυθεί !!!... δοκιμάστε !

- Όμως, η **βέλτιστη λύση** λαμβάνεται **ακαριαία** μέσω του αλγορίθμου simplex:

$$x_{AB} = 0 \qquad x'_{AB} = 7$$
$$x_{B\Gamma} = 1.5 \qquad x'_{B\Gamma} = 1.5$$
$$x_{A\Gamma} = 0.5 \qquad x'_{A\Gamma} = 4.5$$

Η λύση δεν είναι ακέραιη, αλλά για την παρούσα εφαρμογή δεν χρειάζεται να είναι !!!



A-B πληρώνει **3 €**
B-Γ πληρώνει **2 €**
A-Γ πληρώνει **4 €**

- **Παρατήρηση:** Εντός της a-c, κάθε άλλη ακμή χρησιμοποιείται πλήρως !!!

Παρατηρήσεις 1/4

- Η **λύση** του προβλήματος **Κατανομής Εύρους Ζώνης** δεν είναι **ακέραιη** !!!

$$x_{AB} = 0 \qquad x'_{AB} = 7$$
$$x_{BΓ} = 1.5 \qquad x'_{BΓ} = 1.5$$
$$x_{AΓ} = 0.5 \qquad x'_{AΓ} = 4.5$$

- Εάν σε ένα LP απαιτήσουμε, με κατάλληλους περιορισμούς, η λύση μας να είναι **ακέραιη** !!!...

... τότε το πρόβλημα γίνεται

δυσεπίλυτο !!!

Παρατηρήσεις 2/4

- Μια γραμμική συνάρτηση, όπως η $\mathbf{x_1 + 6x_2}$, μπορεί να γραφεί ως εσωτερικό γινόμενο δύο διανυσμάτων:

$$\mathbf{c} = \begin{bmatrix} 1 \\ 6 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

συμβολίζεται ως $\mathbf{c \cdot x}$ ή $\mathbf{c^T x}$

- Όμοια, οι γραμμικοί περιορισμοί μπορούν να γραφούν με μορφή πινάκων-διανυσμάτων:

$$\begin{aligned} x_1 &\le 200 \\ x_2 &\le 300 \\ x_1 + x_2 &\le 400 \end{aligned} \quad \Rightarrow \quad \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\mathbf{x}} \le \underbrace{\begin{bmatrix} 200 \\ 300 \\ 400 \end{bmatrix}}_{\mathbf{b}}$$

Παρατηρήσεις 3/4

- Με αυτό το συμβολισμό, ένα γενικό LP εκφράζεται απλά ως:

$$\max \ \mathbf{c}^T\mathbf{x}$$
$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq \mathbf{0}$$

- Οποιοδήποτε LP μπορούμε να το αναγάγουμε σε ***κανονική μορφή*** (standard form), δηλαδή σε ένα νέο LP όπου:
  - ✓ όλες οι μεταβλητές είναι μη-αρνητικές,
  - ✓ όλοι οι περιορισμοί είναι εξισώσεις, και
  - ✓ η αντικειμενική συνάρτηση ελαχιστοποιείται !!!

Παρατηρήσεις 4/4

Μέθοδοι LP

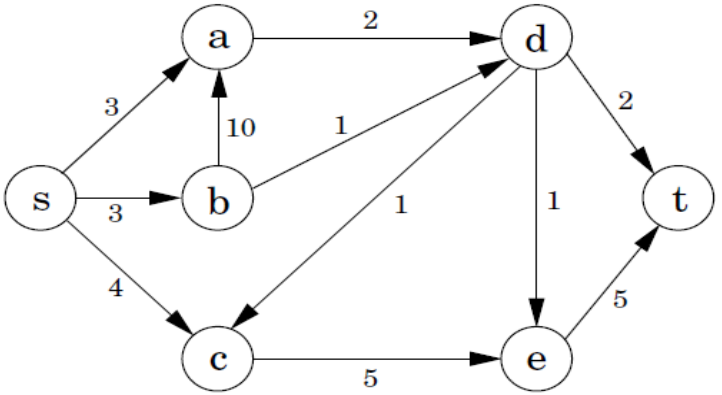
- ✓ **Ο Simplex** δεν είναι πολυωνυμικού χρόνου – πρακτικά πολύ καλός !
- ✓ **Ο Ελλειψοειδής Αλγόριθμος** (Leonid Khachiyan, 1979) είναι πολυωνυμικού χρόνου – πρακτικά κακός !!
- ✓ **Μέθοδο εσωτερικών σημείων** (Narenda Karmarkar, λίγα χρόνια αργότερα), είναι πολυωνυμικού χρόνου – πρακτικά καλός !!!

- Δυϊκότητα
  - ✓ Αποδεικνύεται ότι κάθε max LP έχει ένα *δυϊκό (dual)* min LP !!!

3 Ροές σε Δίκτυα

Έστω ένα *δίκτυο πετρελαιαγωγών* - δίκτυο μεταφοράς πετρελαίου από την περιοχή παραγωγής **s** στην περιοχή κατανάλωσης **t** !!!

Το δίκτυο αναπαρίσταται με ένα κατευθυνόμενο γράφημα $G = (V, E)$ που έχει δύο ειδικούς κόμβους $s, t \in V$, και ακμικές *χωρητικότητες* $c_e > 0$



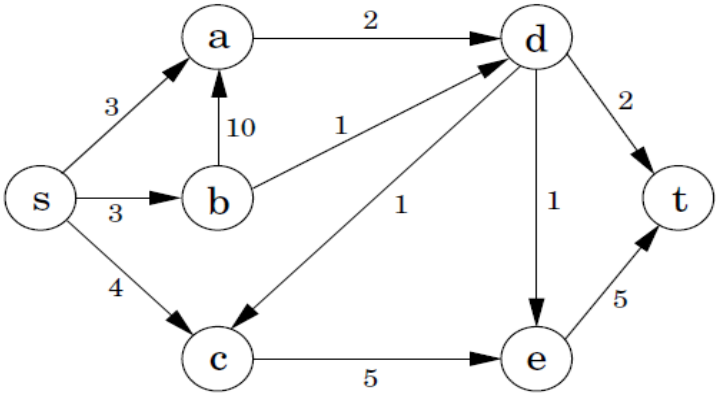
Η αντιστοιχία Δίκτυο ⇔ $G = (V, E)$ είναι προφανής !!!

3 Ροές σε Δίκτυα

Έστω ένα *δίκτυο πετρελαιαγωγών* - δίκτυο μεταφοράς πετρελαίου από την περιοχή παραγωγής **s** στην περιοχή κατανάλωσης **t** !!!

Πρόβλημα

Θέλουμε να μεταφέρουμε όσο το δυνατόν περισσότερο πετρέλαιο από τον **s** στον **t** χωρίς να υπερβούμε τη χωρητικότητα καμιάς ακμής !!!

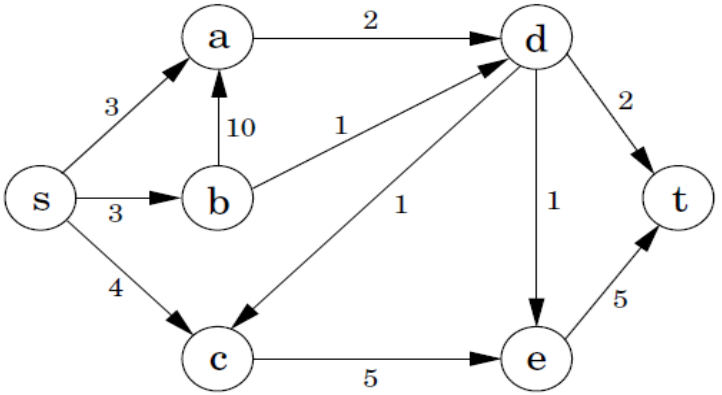


3 Ροές σε Δίκτυα

Ένα συγκεκριμένο σχήμα μεταφοράς *ονομάζεται* **Ροή** (flow), και αποτελείται από μια μεταβλητή $f_e \; \forall \; e \in E$:

(1) $0 \le f_e \le c_e \quad \forall \; e \in E$

(2) $\sum_{xu \in E} f_{xu} = \sum_{uy \in E} f_{uy} \quad \forall \; u \in V\text{-}\{s,t\}$

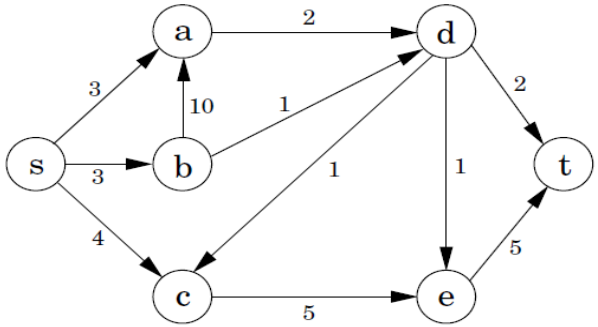


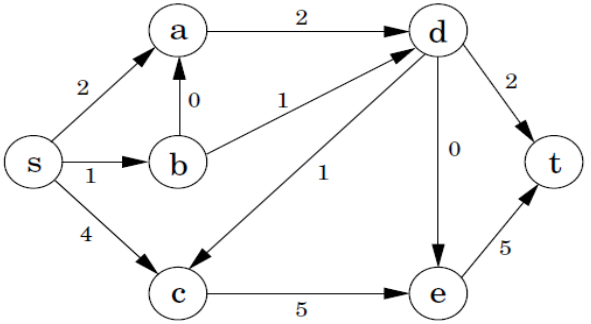
Η ποσότητα ροής που *εισέρχεται* σε ένα κόμβο ***u*** είναι **ίση** με την ποσότητα που *εξέρχεται* από τον ***u*** !!!

Με άλλα λόγια, η ροή διατηρείται !!!

3 Ροές σε Δίκτυα

Το δίκτυο G του παραδείγματος και μια ροή f του G !!!



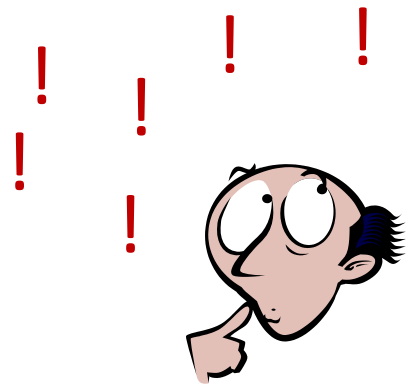


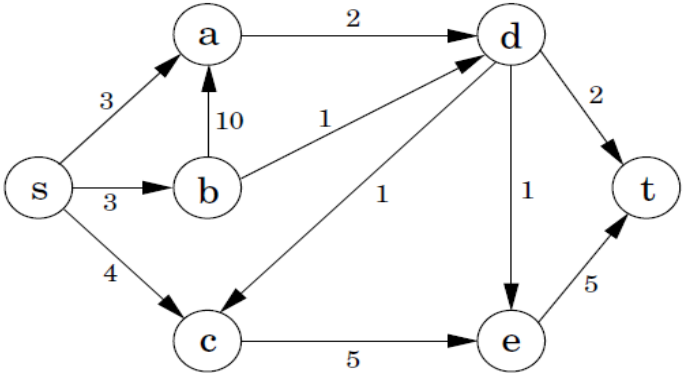
Το **μέγεθος** (size) μιας ροής *f* είναι η συνολική ποσότητα που στέλνεται από το s στον t και, λόγω της αρχής της διατήρησης της ροής, ισούται:

$$\text{Size}(f) = \sum_{su \in E} f_{su}$$

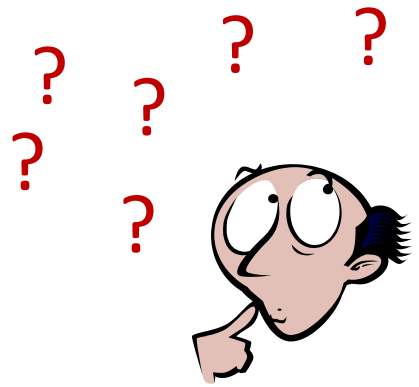
Πρόβλημα

Θέλουμε να μεταφέρουμε όσο το δυνατόν περισσότερο πετρέλαιο από τον **s** στον **t** χωρίς να υπερβούμε τη χωρητικότητα καμιάς ακμής !!!



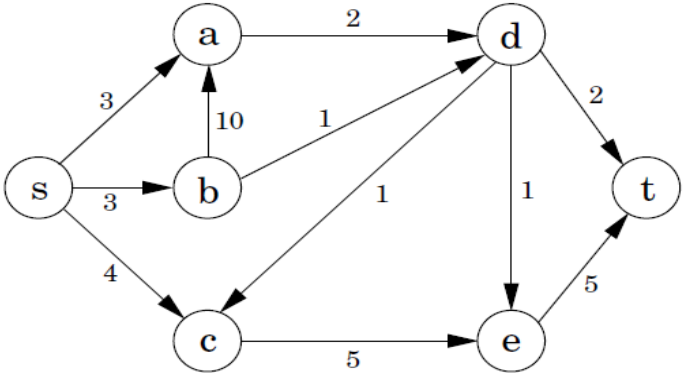


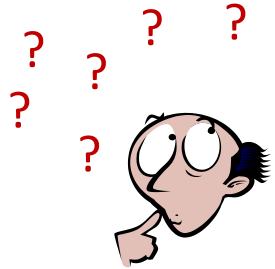
- **Πως θα λύσουμε το πρόβλημα ?**



Πρόβλημα

Θέλουμε να μεταφέρουμε όσο το δυνατόν περισσότερο πετρέλαιο από τον **s** στον **t** χωρίς να υπερβούμε τη χωρητικότητα καμιάς ακμής !!!



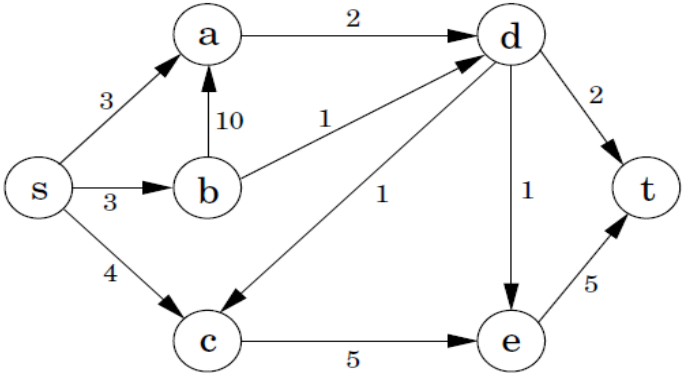


Πως θα λύσουμε το πρόβλημα ?

- Αρκεί να αναθέσουμε τιμές f_e, $\forall\ e \in E$, οι οποίες:
  - ✓ θα *ικανοποιούν* ένα σύνολο γραμμικών περιορισμών, και
  - ✓ θα *μεγιστοποιούν* μια γραμμική αντικειμενική συνάρτηση !!!

Πρόβλημα

Θέλουμε να μεταφέρουμε όσο το δυνατόν περισσότερο πετρέλαιο από τον **s** στον **t** χωρίς να υπερβούμε τη χωρητικότητα καμιάς ακμής !!!

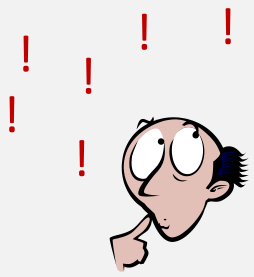


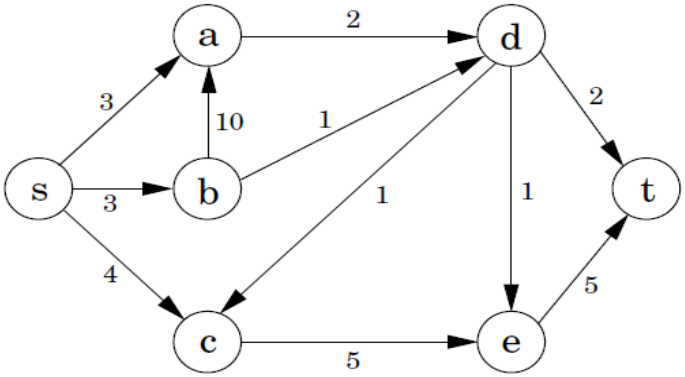
Πως θα λύσουμε το πρόβλημα ?

- Αρκεί να αναθέσουμε τιμές f_e, $\forall\ e \in E$, οι οποίες:
  - ✓ θα *ικανοποιούν* ένα σύνολο γραμμικών περιορισμών, και
  - ✓ θα *μεγιστοποιούν* μια γραμμική αντικειμενική συνάρτηση !!!

Όμως, αυτό είναι ένα

Γραμμικό Πρόγραμμα !!!





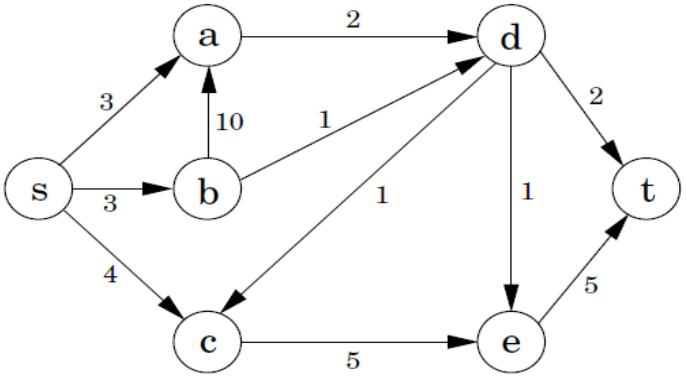
Να και η λύση του !!!

- Για το δίκτυο του σχήματος, το ΓΠ έχει 11 μεταβλητές μία για $\forall$ e $\in$ E, και ζητά να μεγιστοποιήσει την παράσταση:

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

η οποία υπόκειται συνολικά σε 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-{s, t} (όπως $f_{sc} + f_{dc} = f_{ce}$)

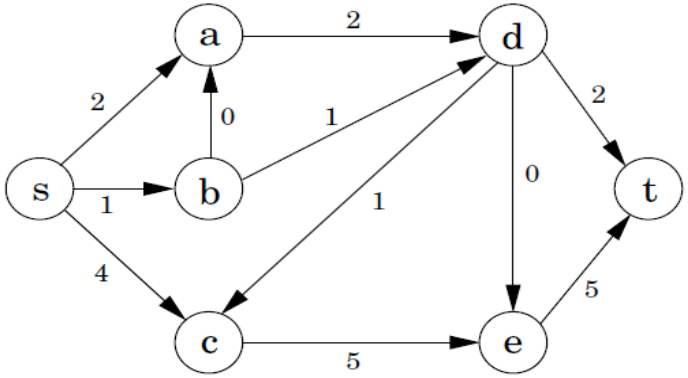


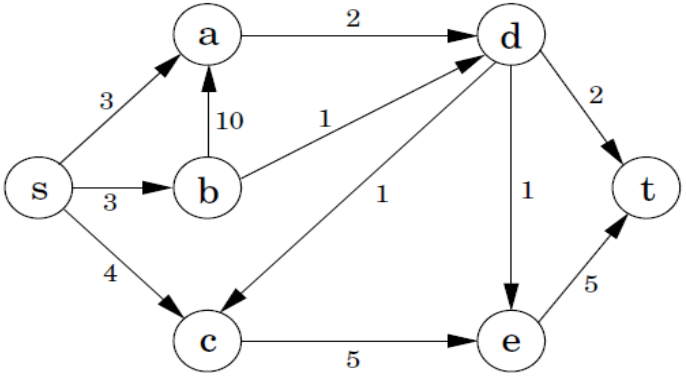
Να και η λύση του !!!

- Για το δίκτυο του σχήματος, το ΓΠ έχει 11 μεταβλητές μία για $\forall$ e $\in$ E, και ζητά να μεγιστοποιήσει την παράσταση:

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

- Ο αλγόριθμος Simplex επιλύει το πρόβλημα σε **ελάχιστο χρόνο**!!!... δίνοντας ροή 7 που είναι πράγματι βέλτιστη !!!





Παρατηρήσεις

- Με τη λύση που μόλις δώσαμε, τι ***πετύχαμε*** ?

  Πετύχαμε να **«ανάγουμε»** το πρόβλημα της μεγίστης ροής σε πρόβλημα Γραμμικού Προγραμματισμού !!!

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

- Ο αλγόριθμος των **Ford-Fulkerson (1956)** είναι ένας **άμεσος αλγόριθμος** για το πρόβλημα της μεγίστης ροής (direct max-flow algorithms) !!!

Υ Υστερόγραφο… Αποτίμηση Κυκλώματος

Η **σημασία** και η **ισχύς** του **Γραμμικού Προγραμματισμού** απορρέει από την μεγάλη ποικιλία και το εύρος των προβλημάτων τα οποία **ανάγονται** σε αυτόν !!!!

Κατά μία έννοια, το επόμενο πρόβλημα που θα μελετήσουμε:

Αποτίμηση Κυκλώματος

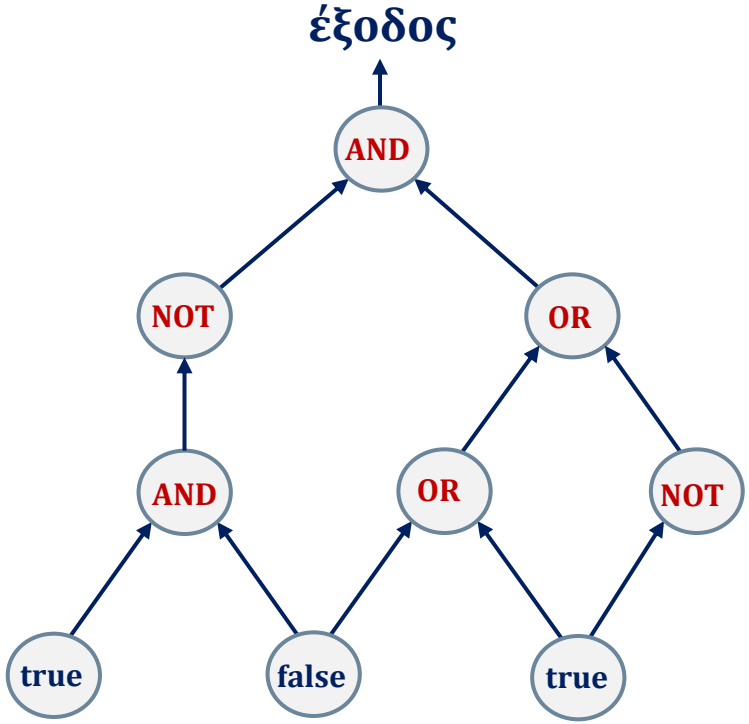
είναι η ***υπέρτατη εφαρμογή*** !!!

5 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**

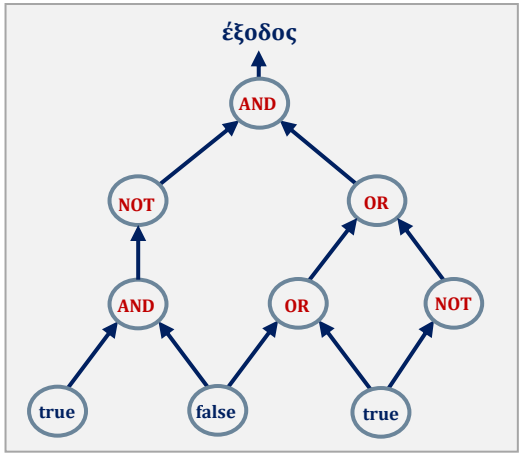
Θέλουμε την αποτίμηση της εξόδου του κυκλώματος σε τιμή true



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 $0 \le x_g \le 1$



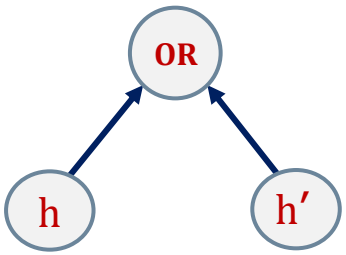
πύλη **g**

$x_g = 1$

g

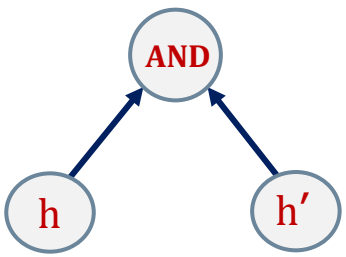
$x_g = 0$

g



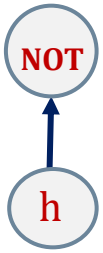
$x_g \ge x_h$
$x_g \ge x_{h'}$
$x_g \le x_h + x_{h'}$

g



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

g

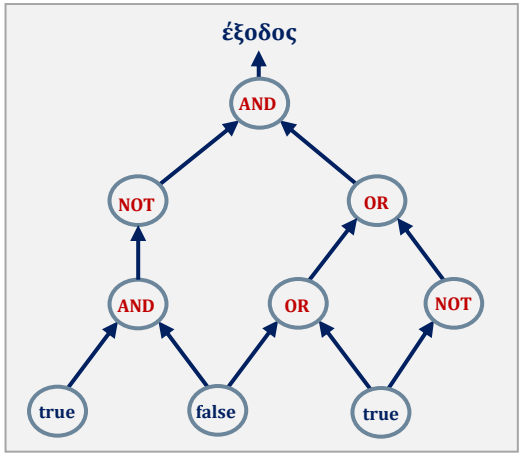


$x_g = 1 - x_h$

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 ελαχιστοποιήσουμε κάτι ... απλώς να πάρουμε την τιμή της μεταβλητής $\mathbf{x_o}$ ⇔ πύλη εξόδου !!!



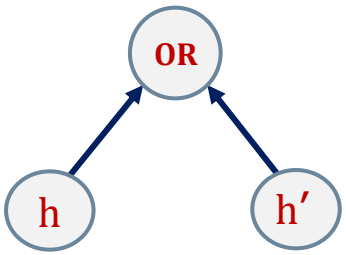
πύλη **g**

$x_g = 1$

g

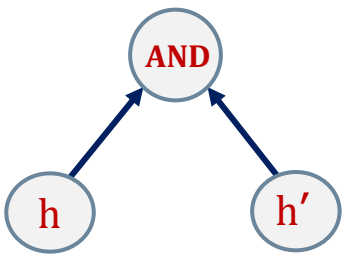
$x_g = 0$

g



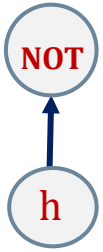
$x_g \geq x_h$
$x_g \geq x_{h'}$
$x_g \leq x_h + x_{h'}$

g



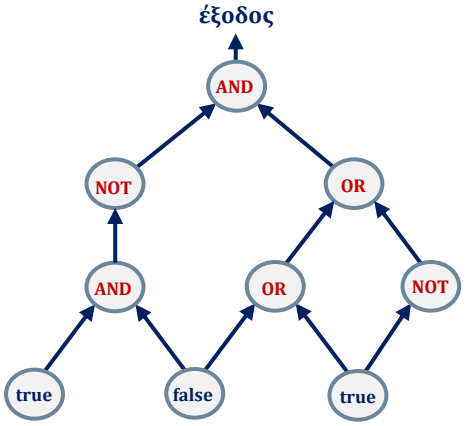
$x_g \leq x_h$
$x_g \leq x_{h'}$
$x_g \geq x_h + x_{h'} - 1$

g



$x_g = 1 - x_h$

Παρατηρήσεις 1/3



- Δώσαμε **άμεση αναγωγή σε LP** από ένα πρόβλημα το οποίο εκ πρώτης όψεως ***δεν φαίνεται ενδιαφέρον*** !!!

  Όμως, κατά μια έννοια, το πρόβλημα

  ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ

  είναι :

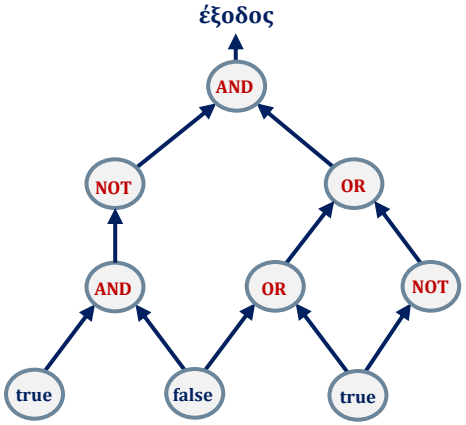
  Το πιο γενικό πρόβλημα επιλύσιμο σε πολυωνυμικό χρόνο !!!

- **Ας δούμε γιατί !!!**

  Που εκτελείται ο αλγόριθμος ?... σε έναν **Η/Υ** !!!

  Τι είναι ένας Η/Υ ?... ένα **λογικό συνδυαστικό κύκλωμα** σε ένα τσιπ !!!

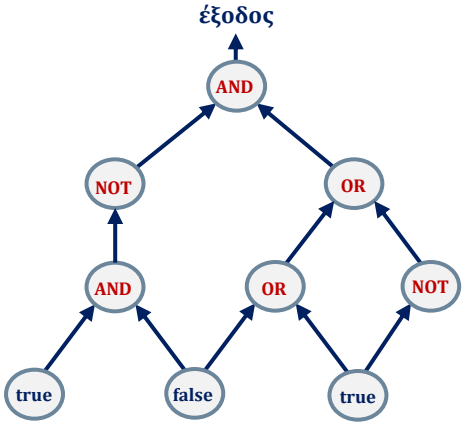
Παρατηρήσεις 2/3



- **Επομένως, ένας πολυωνυμικός αλγόριθμος μπορεί να θεωρηθεί ως:**
  - ✓ ένα ***λογικό κύκλωμα*** αποτελούμενο από πολυωνυμικό πλήθος αντιγράφων του ***λογικού κυκλώματος του Η/Υ*** !!!
  - ✓ ένα ***αντίγραφο ανά μονάδα χρόνου*** !!!... όπου οι τιμές στις πύλες του ενός επιπέδου χρησιμοποιούνται για τον υπολογισμό των τιμών του επομένου !!!
- Άρα, το γεγονός ότι το πρόβλημα ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ **ανάγεται** σε LP σημαίνει ότι:

> ***όλα τα προβλήματα*** *τα οποία μπορούν να λυθούν σε πολυωνυμικό χρόνο* ***μπορούν να αναχθούν*** *σε* ***Γραμμικά Προγράμματα*** *(LP) !!!*

Παρατηρήσεις 3/3



- **Και μια τελευταία σκέψη !!!**
  - ✓ Πως αλλιώς μπορεί να λυθεί το πρόβλημα ΤΙΜΗ-ΚΥΚΛΩΜΑΤΟΣ ?
  - ✓ Ας σκεφτούμε λίγο… το κύκλωμα είναι ένα DAG !!!
  - ✓ Ποια αλγοριθμική τεχνική είναι κατάλληλη για προβλήματα σε DAG ?
  - ✓ Σωστά !!!... ο **Δυναμικός Προγραμματισμός** !!!

Δυναμικός Προγραμματισμός και ***Γραμμικός Προγραμματισμός***
Ίσως οι δύο πιο Γενικές Αλγοριθμικές Τεχνικές
του Κόσμου έως Σήμερα !!!

Ευχαριστώ για τη Συμμετοχή σας !!!

Σταύρος Δ. Νικολόπουλος

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Ιωαννίνων

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