Σχεδίαση και Ανάλυση Αλγορίθμων

Ενότητα 4.0

Επιλογή

Αλγόριθμοι Επιλογής Select και Quick-Select

Σταύρος Δ. Νικολόπουλος | 2017-18
Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Ιωαννίνων
Webpage: www.cs.uoi.gr/~stavros

Επιλογή… Γρήγορη Επιλογή

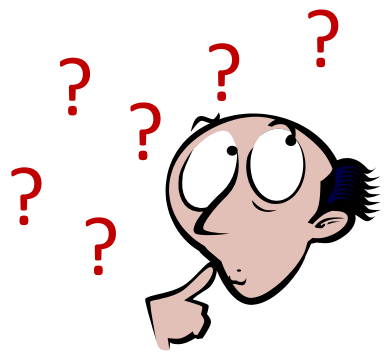
Ας Ξεκινήσουμε με ένα «Απλό» Ερώτημα !

Γνωρίζουμε αλγορίθμους επιλογής ?

- Έχουμε διδαχθεί, έως τώρα, κάποιον αλγόριθμο επιλογής?

ΝΑΙ !!!

- ✓ Αλγόριθμο υπολογισμού του Min ή Max στοιχείου ενός συνόλου… n-1 συγκρίσεις !!!
- ✓ Αλγόριθμο υπολογισμού του 2-ου Min ή 2-ου Max στοιχείου ενός συνόλου… 2n-3 συγκρίσεις !!!



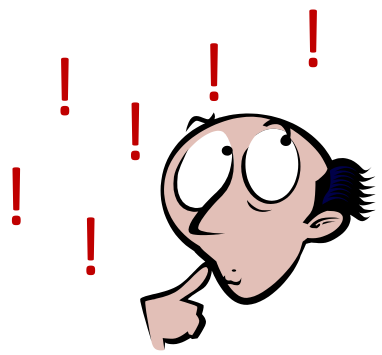
Πρώτο «Εύλογο» Ερώτημα !!

- Μπορούμε με να βρούμε το 2ο Min (ή 2ο Max) με λιγότερες από 2n-3 συγκρίσεις (2 φορές εφαρμογή του findMin) ?

? < 2n-3

Για να απαντήσουμε ΝΑΙ θα πρέπει:

- ✓ Να *μελετήσουμε* το πρόβλημα και να βρούμε *ιδιότητες* που ισχύουν !!!
- ✓ Να επινοήσουμε μια *καλή αλγοριθμική τεχνική* ή ακόμα συνδυασμό τεχνικών !!!
- ✓ Και φυσικά να έχουμε την *«καλή» Ιδέα* !!!



Μια «Καλή» Ιδέα !!

- Κατά τον υπολογισμό του 1ου Min (ή 1ου Max), τι μπορούμε να παρατηρήσουμε ότι ισχύει?

L(1), L(2), L(3), L(4), L(5)

Αρχικά: Min (ή Max) = L(1)

Συγκρίσεις		Νικητής
L(1), L(2)	→	L(1)
L(1), L(3)	→	L(1)
L(1), L(4)	→	L(4)
L(4), L(5)	→	L(4)

Min (ή Max) = L(4)

To 2ο Min (ή 2ο Max) = L(5) ή L(1)

L(5) και L(1)
είναι τα στοιχεία που «έχασαν»
από τον νικητή !!!

Μια «Καλή» Ιδέα !!

- Κατά τον υπολογισμό του 1ου Min (ή 1ου Max), τι μπορούμε να παρατηρήσουμε ότι ισχύει?

L(1), L(2), L(3), L(4), L(5)

Αρχικά: Min (ή Max) = L(1)

Συγκρίσεις		Νικητής
L(1), L(2)	→	L(1)
L(1), L(3)	→	L(1)
L(1), L(4)	→	L(1)
L(1), L(5)	→	L(1)

Min (ή Max) = L(1)

Το 2ο Min (ή 2ο Max) = ?

Εάν Min (ή Max) = L(1) **?**

Μια «Καλή» Ιδέα !!

- Κατά τον υπολογισμό του 1ου Min (ή 1^{ου} Max), τι μπορούμε να παρατηρήσουμε ότι ισχύει?

L(1), L(2), L(3), L(4), L(5)

Αρχικά: Min (ή Max) = L(1)

Συγκρίσεις		Νικητής
L(1), L(2)	→	L(1)
L(1), L(3)	→	L(1)
L(1), L(4)	→	L(1)
L(1), L(5)	→	L(1)

?

Min (ή Max) = L(1)

Συνολικά **2n-3** συγκρίσεις !!!

Δεν κερδίζω κάτι !!!

Το 2ο Min (ή 2ο Max) = L(2) ή L(3) ή L(4) ή L(5)

Μια «Καλή» Ιδέα !!

- Κατά τον υπολογισμό του 1ου Min (ή 1^{ου} Max), τι μπορούμε να παρατηρήσουμε ότι ισχύει?

L(1), L(2), L(3), L(4), L(5)

Αρχικά: Min (ή Max) = L(1)

Δεν κερδίσω συγκρίσεις…

κερδίζω όμως την καλή Ιδέα !!!

Συγκρίσεις	Νικητής
L(1), L(2)	L(1)
L(1), L(3)	L(1)
L(1), L(4)	L(1)
L(1), L(5)	L(1)

?

Min (ή Max) = L(1)

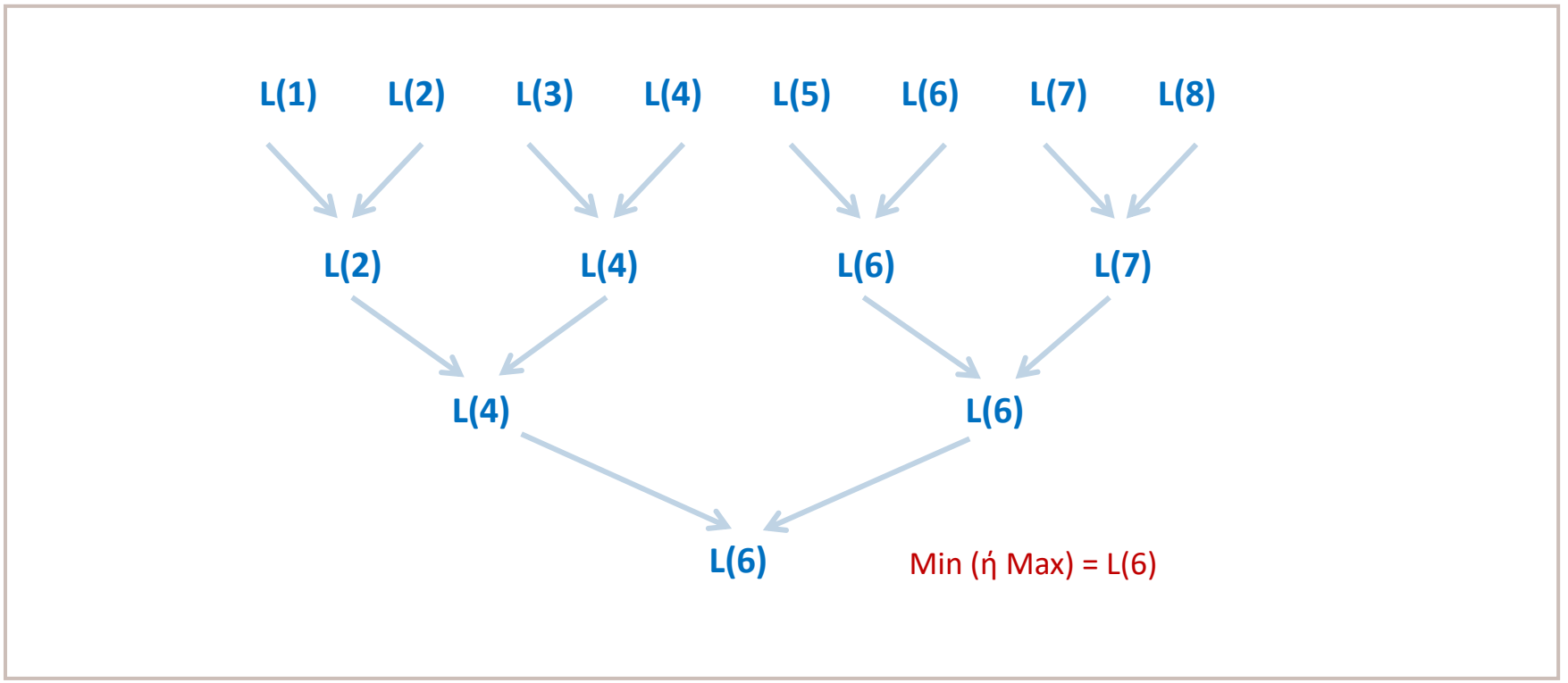
Συνολικά **2n-3** συγκρίσεις !!!

Δεν κερδίζω κάτι !!!

Το 2ο Min (ή 2ο Max) = L(2) ή L(3) ή L(4) ή L(5)

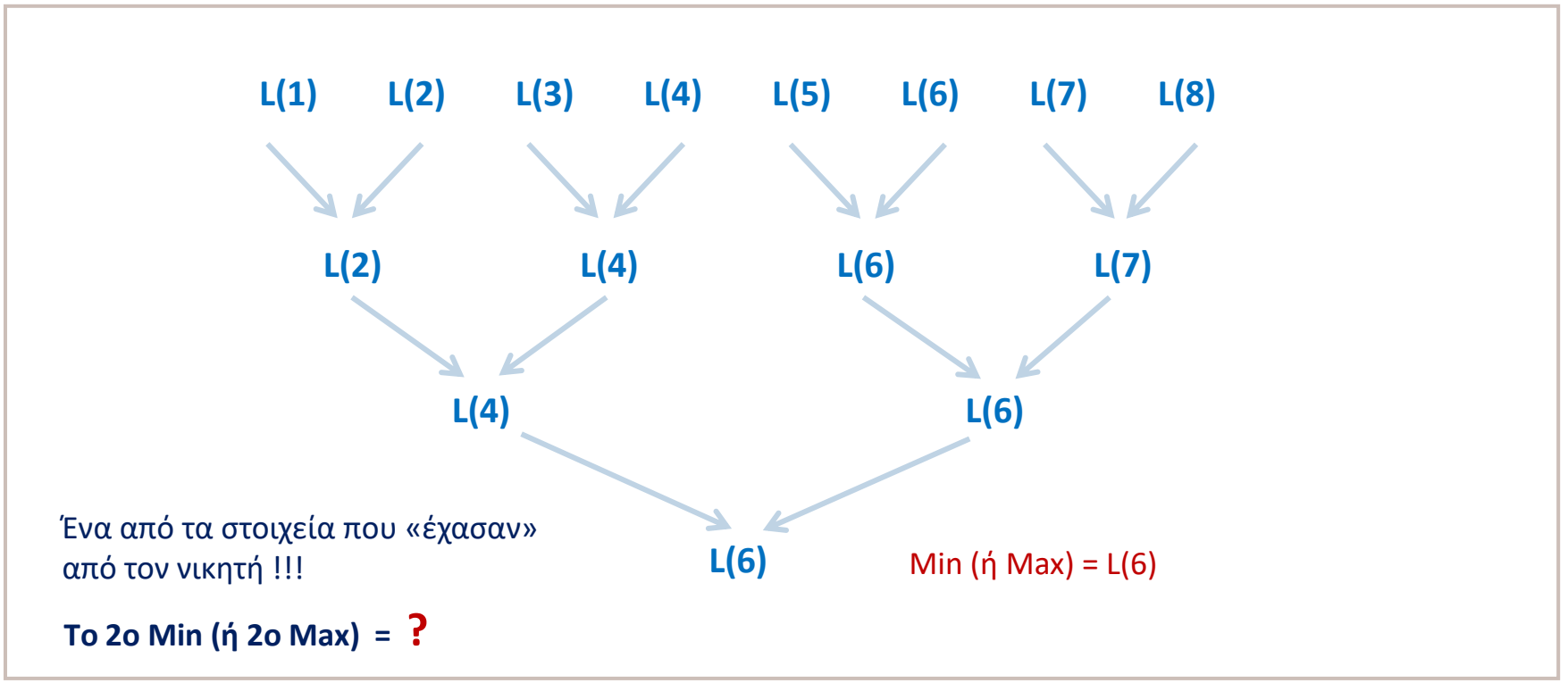
Μια «Καλή» Ιδέα !!

- Επινόηση Τεχνικής… Τουρνουά



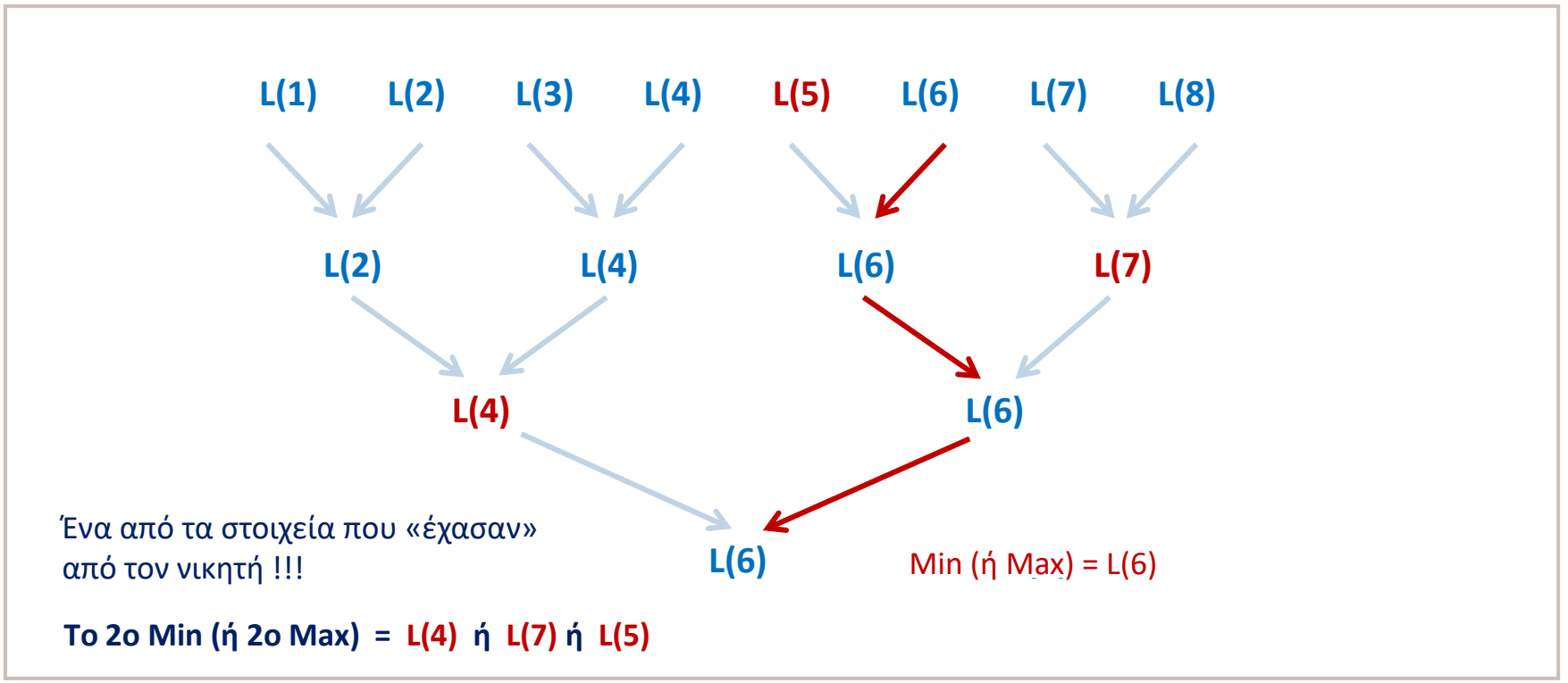
Μια «Καλή» Ιδέα !!

- Επινόηση Τεχνικής… Τουρνουά



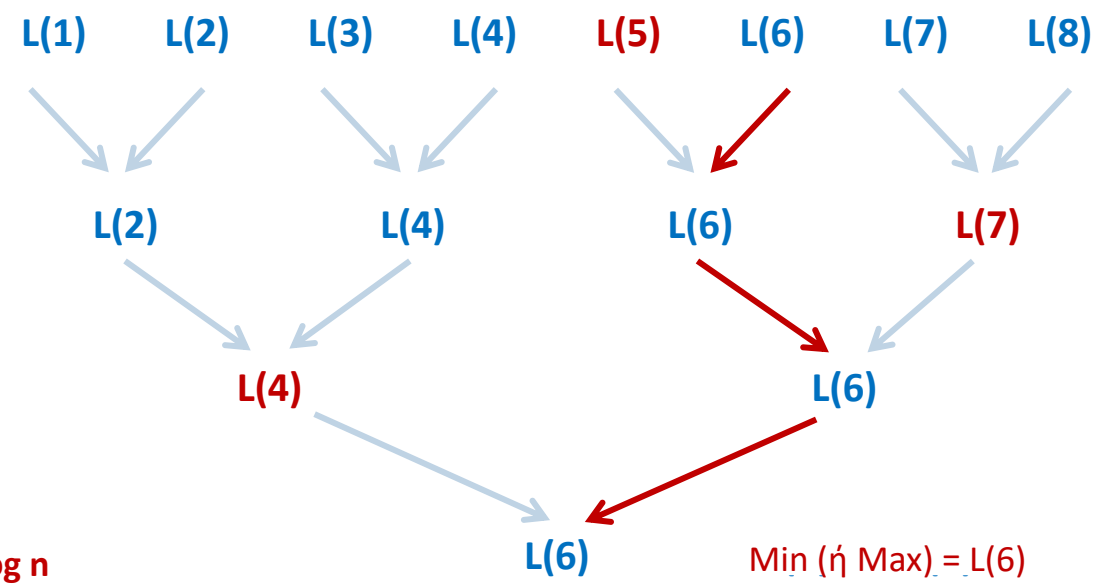
Μια «Καλή» Ιδέα !!

- Επινόηση Τεχνικής… Τουρνουά



Μια «Καλή» Ιδέα !!

- Επινόηση Τεχνικής… Τουρνουά



Πλήθος = log n

Το 2ο Min (ή 2ο Max) = L(4) ή L(7) ή L(5)

Min (ή Max) = L(6)

Συνολικά απαιτούνται

$n + \lceil \log n \rceil - 2$

συγκρίσεις !!!

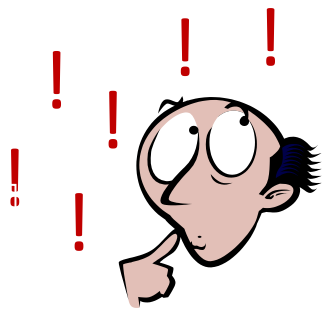
Απάντηση στο Πρώτο «Εύλογο» Ερώτημα !!

- Στο ερώτημα: Μπορούμε με να βρούμε το 2ο Min με λιγότερες από 2n-3 συγκρίσεις (2 φορές εφαρμογή του findMin) **?**

  Η απάντηση είναι ΝΑΙ !!!

$$n + \lceil \log n \rceil - 2 \; < \; 2n-3$$

Μπορούμε με λιγότερες συγκρίσεις ?



Θεώρημα: Κάθε αλγόριθμος (που εργάζεται με συγκρίσεις) για την εύρεση του 2ου Min (ή $2^{ου}$ Max) από n στοιχεία απαιτεί τουλάχιστον $n + \lceil \log n \rceil - 2$ συγκρίσεις στην χειρότερη περίπτωση !!!

Επόμενο «Εύλογο» Ερώτημα !!!

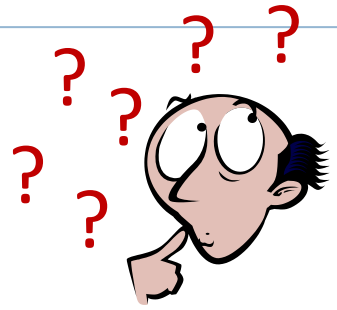
- Μπορούμε να υπολογίσουμε το 3ο Min ενός συνόλου ;
  το 4ο, το 5ο Min στοιχείο, …, το $\lceil n/2 \rceil$-στο Min (median) ενός συνόλου ;

  Γενικά, το k-στο Min στοιχείο ενός συνόλου ;

Αποτελεσματικά !!!

Το γενικό ερώτημα ορίζει το πρόβλημα της **ΕΠΙΛΟΓΗΣ** !!!

Και η απάντηση στο ερώτημα για «**Αποτελεσματικό Υπολογισμό**» δεν είναι προφανής !!!



Επόμενο «Εύλογο» Ερώτημα !!!

- Μπορούμε να υπολογίσουμε το 3ο Min ενός συνόλου ?
  το 4ο, το 5ο Min στοιχείο, …, το $\lceil n/2 \rceil$-στο Min (median) ενός συνόλου ?

  Γενικά, το k-στο Min στοιχείο ενός συνόλου ?

Αποτελεσματικά !!!

Το γενικό ερώτημα ορίζει το πρόβλημα της **ΕΠΙΛΟΓΗΣ** !!!

Και η απάντηση στο ερώτημα για «**Αποτελεσματικό Υπολογισμό**»
δεν είναι προφανής !!!... χρειάζεται η «καλή» Ιδέα !!!

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Προφανής Αλγόριθμος!

Ταξινόμηση του συνόλου S σε χρόνο O(nlogn) και εύρεση του στοιχείου x σε σταθερό χρόνο!!!

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Ερώτημα !

Μπορούμε να βρούμε το k-στο μικρότερο στοιχείο σε χρόνο O(n) ?

δηλαδή, υπάρχει κάποιος καλύτερος αλγόριθμος από αυτόν που χρησιμοποιεί ταξινόμηση ?

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Παρατήρηση !

4	2	9	7	12	14	8	1	13	6	3	10	11

1	2	3	4	6	7	8	9	10	11	12	13	14

4	2	1	7	6	3	8	13	11	9	14	10	12

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Στο Ερώτημα !

Μπορούμε να βρούμε το k-στο Min στοιχείο του S σε χρόνο O(n) ?

Η απάντηση είναι **ΝΑΙ** !!!

Divide & Conquer

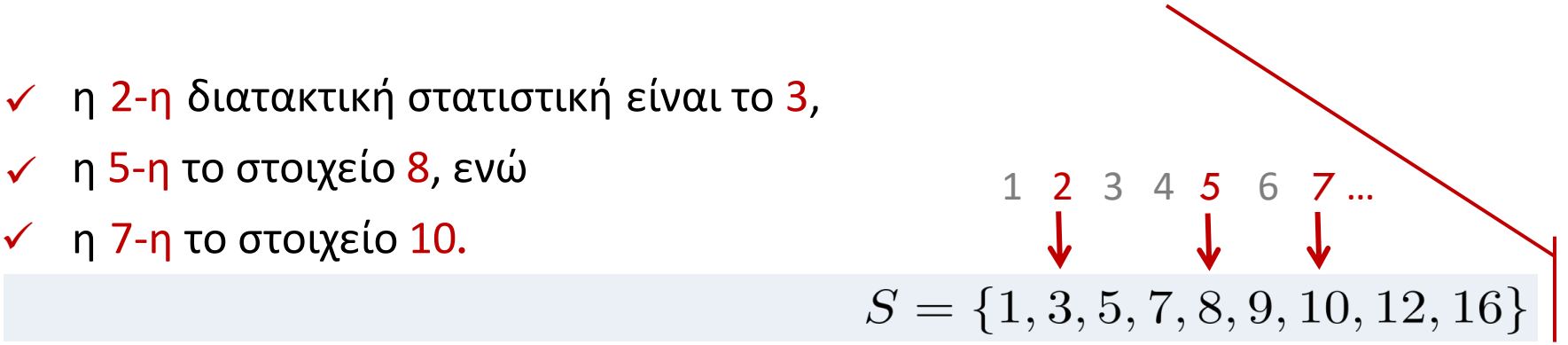
Διατακτική Στατιστική Τάξης k ≡ k-στο Min

- Έστω S ένα σύνολο n στοιχείων (για ευκολία θεωρούμε ότι είναι ακέραιοι).

- Η **k-οστή διατακτική στατιστική** του συνόλου S είναι το στοιχείο **x ∈ S** που είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S.

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9\}$$

- ✓ η 2-η διατακτική στατιστική είναι το 3,
- ✓ η 5-η το στοιχείο 8, ενώ
- ✓ η 7-η το στοιχείο 10.



Διατακτική Στατιστική Τάξης k ≡ k-στο Min

- Έστω S ένα σύνολο n στοιχείων (για ευκολία θεωρούμε ότι είναι ακέραιοι).

- Η **k-οστή διατακτική στατιστική** του συνόλου S είναι το στοιχείο **x ∈ S** που είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S.

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9\}$$

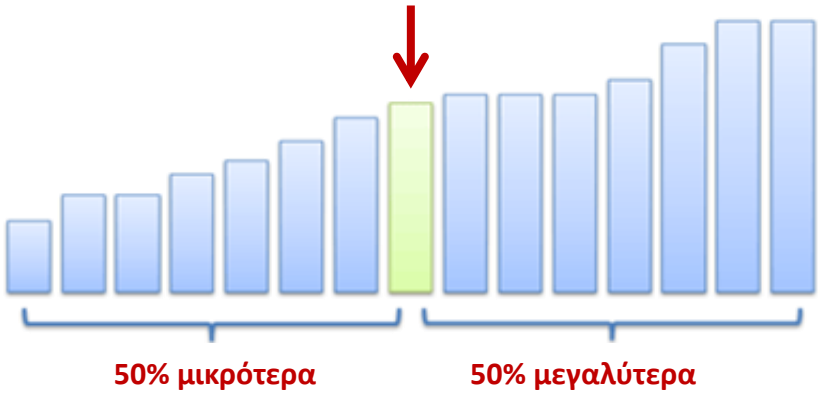
✓ η 1-η διατακτική στατιστική (i = 1) είναι το **Min**

✓ η n-οστή διατακτική (i = n) είναι το μέγιστο **Max**

$$S = \{1, 3, 5, 7, 8, 9, 10, 12, 16\}$$

Διάμεσος

- Η **διάμεσος** ενός συνόλου S με n στοιχεία, σε περιγραφικό επίπεδο, είναι το «μέσο στοιχείο» του συνόλου…



… το στοιχείο **x** του **S** τέτοιο ώστε:

το 50% των στοιχείων του συνόλου S να είναι μικρότερα από αυτό !!!

Διάμεσος

- Η **διάμεσος** ενός συνόλου S με n στοιχεία, σε περιγραφικό επίπεδο, είναι το «μέσο στοιχείο» του συνόλου…

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9\}$$

… το στοιχείο **x** του **S** τέτοιο ώστε:

το 50% των στοιχείων του συνόλου S να είναι μικρότερα από αυτό.

1, 3, 5, 7 < **8** < **9, 10, 12, 16**

Διάμεσος

- Η **διάμεσος** ενός συνόλου S με n στοιχεία, σε περιγραφικό επίπεδο, είναι το «μέσο στοιχείο» του συνόλου…

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9\}$$

- Πληθικότητα του S : **n = περιττός**

Υπάρχουν μόνο ένας διάμεσος:

✓ η υπ' αριθμόν $k = (n+1)/2$ διατακτική στατιστική.

7, 1, 3, 5 < **8** < **12, 16, 10, 9**

Διάμεσος

- Η **διάμεσος** ενός συνόλου S με n στοιχεία, σε περιγραφικό επίπεδο, είναι το «μέσο στοιχείο» του συνόλου…

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9, 11\}$$

- Πληθικότητα του S : **n = άρτιος**

Υπάρχουν δύο διάμεσοι:

- ✓ η υπ' αριθμόν $k = n/2$ διατακτική στατιστική, και
- ✓ η υπ' αριθμόν $k = n/2 + 1$ διατακτική στατιστική.

7, 1, 3, 5 < **8** < **9** < **12, 16, 10, 11**

Διάμεσος

- Η **διάμεσος** ενός συνόλου S με n στοιχεία, σε περιγραφικό επίπεδο, είναι το «μέσο στοιχείο» του συνόλου…

$$S = \{12, 16, 8, 10, 7, 1, 3, 5, 9, 11\}$$

- Για οποιοδήποτε **n**, οι διάμεσοι είναι:
  - ✓ $k = \lfloor (n+1)/2 \rfloor$ διατακτική στατιστική (κάτω διάμεσος)
  - ✓ $k = \lceil (n+1)/2 \rceil$ διατακτική στατιστική (άνω διάμεσος).

Με τον όρο «διάμεσος» θα αναφερόμαστε στον κάτω διάμεσο:

$$k = \lfloor (n+1)/2 \rfloor$$

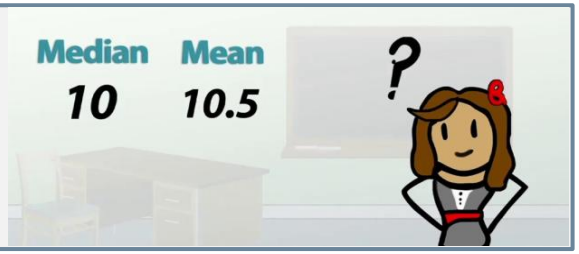
Διάμεσος & Μέση Τιμή

- Η **διάμεσος** (median) ενός συνόλου S έχει σκοπός να «συνοψίσει» ένα σύνολο αριθμών με μία μόνο τυπική τιμή.

- Η **μέση τιμή** (mean) ή **μέσος όρος** (average) πολύ συχνά χρησιμοποιείται για τον ίδιο σκοπό!!!

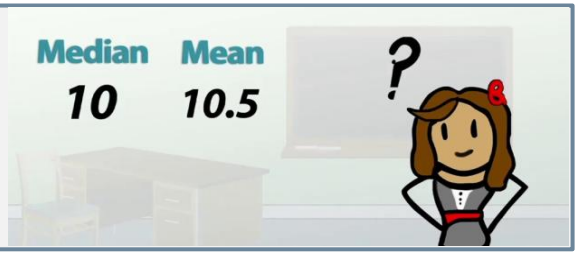
Διάμεσος vs Μέση Τιμή

Η *τιμή της διαμέσου* είναι *πιο αντιπροσωπευτική* για τα δεδομένα απ' αυτή της *μέσης τιμής* !!!



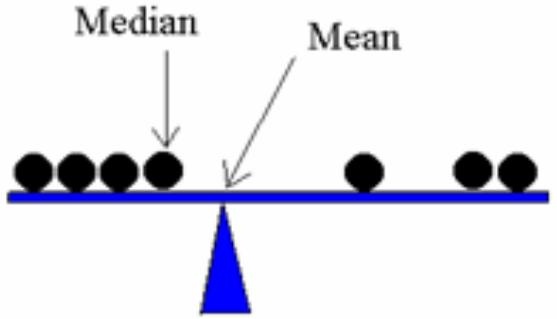
Διάμεσος & Μέση Τιμή

Η *τιμή της διαμέσου* είναι *πιο αντιπροσωπευτική* για τα δεδομένα απ' αυτή της *μέσης τιμής* !!!



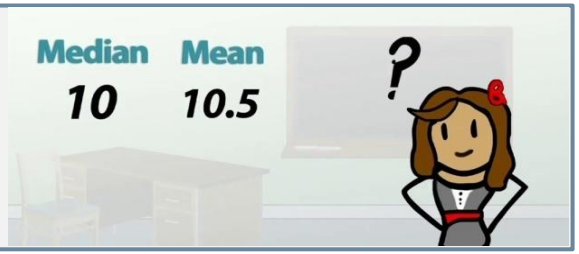
Διάμεσος vs Μέση Τιμή

I. Η διάμεσος είναι πάντοτε μια τιμή από τις τιμές των δεδομένων, σε αντίθεση με την μέση τιμή !



Διάμεσος & Μέση Τιμή

Η *τιμή της διαμέσου* είναι *πιο αντιπροσωπευτική* για τα δεδομένα απ' αυτή της *μέσης τιμής* !!!



Διάμεσος **vs** Μέση Τιμή

II. Η διάμεσος είναι λιγότερο «ευαίσθητη» σε τιμές που απέχουν πολύ από τις υπόλοιπες τιμές των δεδομένων !!

Διάμεσος : $S_1 = 1$ $S_2 = 1$

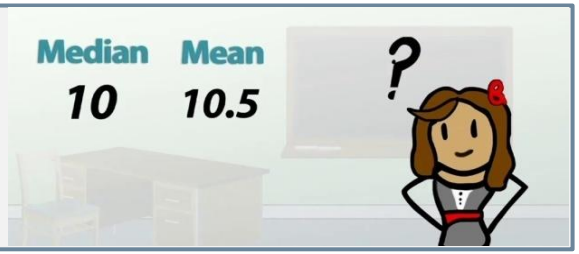
Μέση τιμή : $S_1 = 1$ $S_2 = 100.99$

$$S_1 = (\underbrace{1, 1, \cdots, 1, 1, 1}_{100})$$

$$S_2 = (\underbrace{1, 1, \cdots, 1, 1, 10000}_{100})$$

Διάμεσος & Μέση Τιμή

Η *τιμή της διαμέσου* είναι *πιο αντιπροσωπευτική* για τα δεδομένα απ' αυτή της *μέσης τιμής* !!!

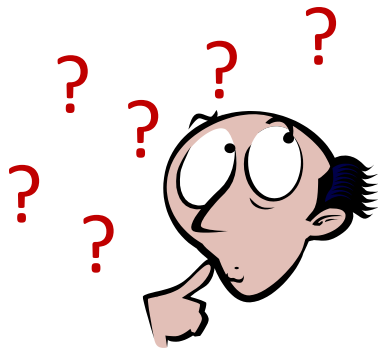


Διάμεσος vs Μέση Τιμή

✓ Τι μπορούμε να υπολογίσουμε γρηγορότερα…

τη *Μέση τιμή* ή τη *Διάμεσο*;

$$W_{\text{mean}} = O(n) \qquad W_{\text{median}} = ?$$



Υπολογισμός Διαμέσου

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$, βρες τη διάμεσο του S, δηλαδή το στοιχείο $x \in S$ το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S, όπου $k = \lfloor (n+1)/2 \rfloor$

Υπολογισμός Διαμέσου $\Longrightarrow$ **Πρόβλημα Επιλογής**

Εάν μπορώ να βρω το k-τάξης στοιχείο ενός συνόλου S σε χρόνο O(n),

τότε μπορώ να βρω και την διάμεσο του S σε χρόνο O(n) !!!

Διάμεσος = k-τάξης στοιχείο ενός συνόλου S !!!

$k = \lfloor (n+1)/2 \rfloor$

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Στο Ερώτημα !

Μπορούμε να βρούμε το k-τάξης στοιχείο ενός συνόλου S σε χρόνο O(n) ?

Η απάντηση είναι **ΝΑΙ** !!!

Divide & Conquer

Το Πρόβλημα της Επιλογής

- Δοθέντος ενός συνόλου S με n στοιχεία και ενός ακεραίου k, $1 \leq k \leq n$,
  βρες το k-στο Min ή k-τάξης στοιχείο $x \in S$
  (το στοιχείο το οποίο είναι μεγαλύτερο από ακριβώς k-1 στοιχεία του S).

Στο Ερώτημα !

Μπορούμε να βρούμε το k-τάξης στοιχείο ενός συνόλου S σε χρόνο O(n) ?

Η απάντηση είναι **ΝΑΙ** !!!

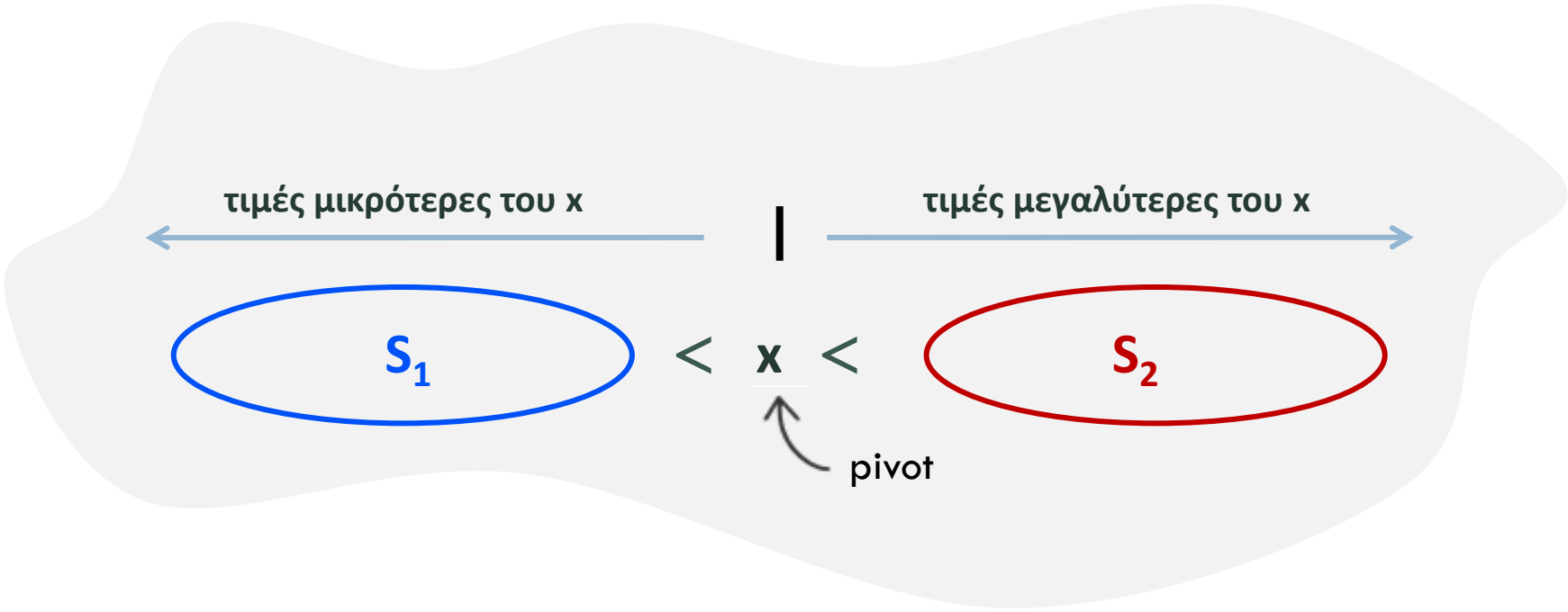
Select(S, k)

Quick-Select(S, k)

- Αλγόριθμοι Select και Quick-select

Αλγόριθμος Select(S,k)

Divide & Conquer



Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

Input: ένα σύνολο S με n στοιχεία
Output: το k-τάξης στοιχείο του συνόλου S

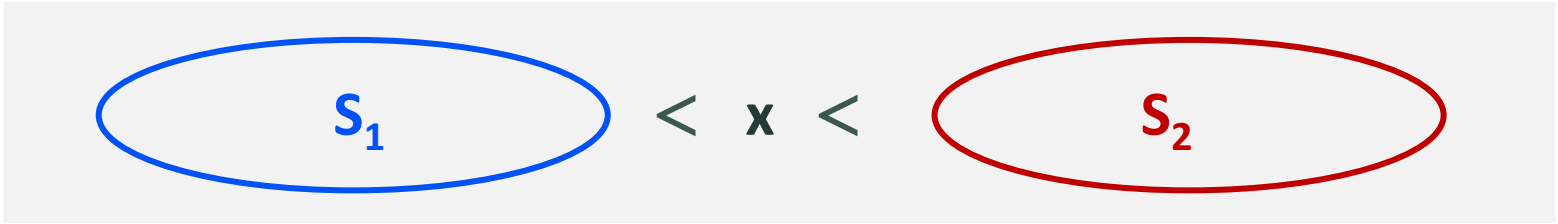
	Διαίρεσε 1	(divide)
	Κατέκτησε 2	(conquer)
Καμία ενέργεια !!!	Συνδύασε 3	(combine)

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

  Επιλογή ενός στοιχείου x του S (pivot) και διαχωρισμός του S σε S_1 και S_2 με βάση το pivot x




```
pivot = S[first], sp = first
for i = first+1 to last
  if S[i] < pivot then
    sp++, swap(S[sp],S[i])
  end
swap(S[first],S[sp]
```

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

Έστω ότι αναζητώ την **διάμεσο** του S, δηλ. $\mathbf{k} = \lfloor (n+1)/2 \rfloor$

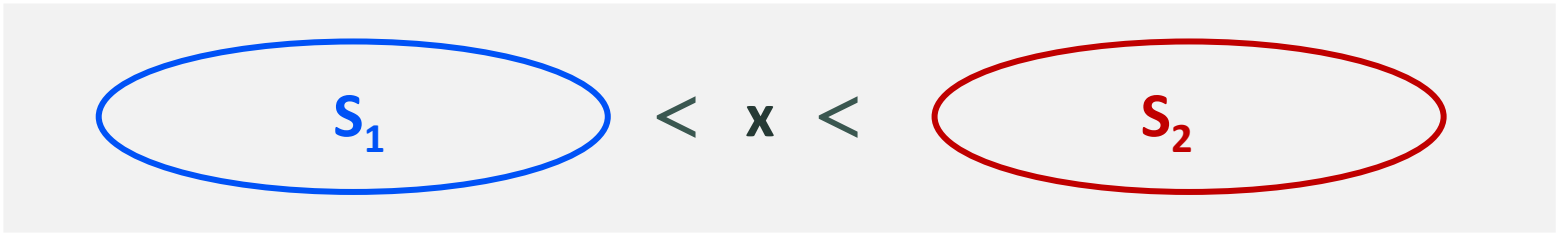
και έστω ότι η διάμεσος δεν είναι το στοιχείο **x**

Εύλογη ερώτηση !!!

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)



- Κατάκτηση (conquer)

Σε ποιο σύνολο από τα S_1 και S_2 βρίσκεται η διάμεσος ?

$$k = \lfloor (n+1)/2 \rfloor$$

Εύλογο ερώτημα !!!

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

Βρίσκεται σε αυτό που έχει τα

ΠΕΡΙΣΣΟΤΕΡΑ ΣΤΟΙΧΕΙΑ

Όχι κατ' ανάγκη στο σύνολο που έχει τα μεγαλύτερα στοιχεία… βλέπετε γιατί ?

Εύλογο ερώτημα !!!

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

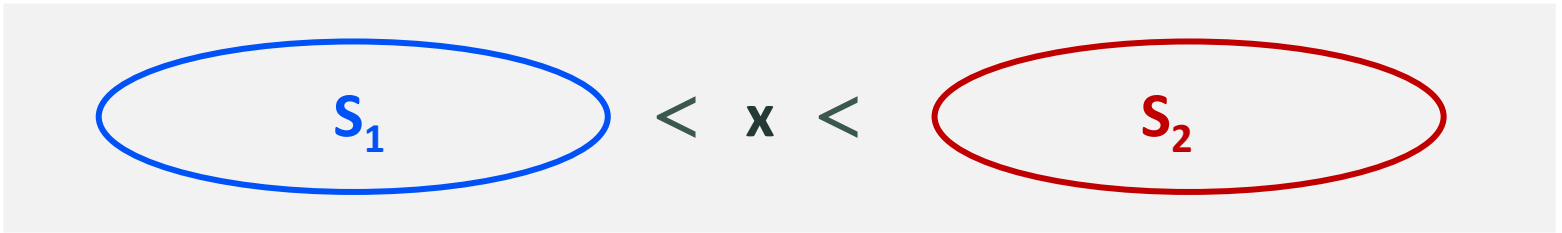
Ας δούμε τη γενική περίπτωση !!!

- Αναζητώ το k-μικρότερο ή k-τάξης στοιχείο του συνόλου S !
- $1 \leq k \leq n$, όπου n η πληθικότητα του S

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

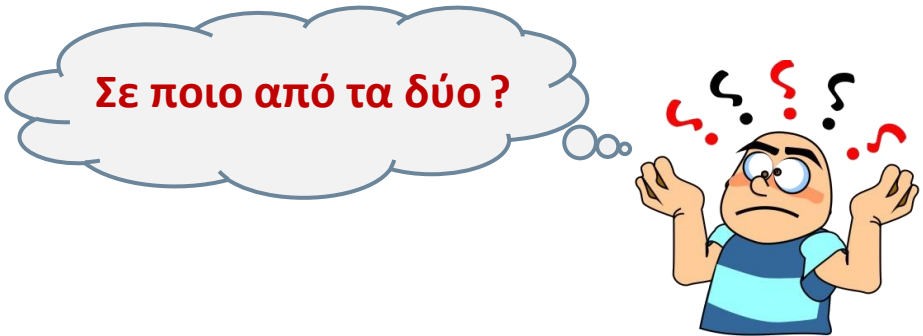
- Διαίρεση (divide)



- Κατάκτηση (conquer)

Είναι προφανές ότι το k-τάξης στοιχείο του συνόλου S που αναζητώ :

- είτε είναι το στοιχείο **x**
- είτε βρίσκεται σε ένα από τα S_1 και S_2



Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

a) $|S_1| = k-1$
b) $|S_1| > k-1$
c) $|S_1| < k-1$

Για την εύρεση του k-τάξης στοιχείου του συνόλου S εξετάζουμε 3 περιπτώσεις !!!

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

a) εάν $|S_1| = k-1$ τότε το **x** είναι το k-στο μικρότερο στοιχείο του S

k = 5 **{3, 5, 1, 7} < 8 < {12, 10, 9, 16}**

Στοιχείο 8

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

b) εάν $|S_1| > k-1$ τότε το k-στο μικρότερο στοιχείο του S βρίσκεται στο S_1 και είναι το k-στο μικρότερο στοιχείο του S_1

k = 3 **{3, 5, 1, 7} < 8 < {12, 10, 9, 16}**

Στοιχείο 5

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)

c) εάν $|S_1| < k-1$ τότε το k-στο μικρότερο στοιχείο του S βρίσκεται στο S_2 και είναι το $(k-|S_1|-1)$-στο μικρότερο στοιχείο του S_2

k = 7 **{3, 5, 1, 7} < 8 < {12, 10, 9, 16}**

Στοιχείο 10

Αλγόριθμος Select(S, k)

Ιδέα Αλγόριθμου !!!

- Διαίρεση (divide)

- Κατάκτηση (conquer)
- Συνδυασμός (combine)

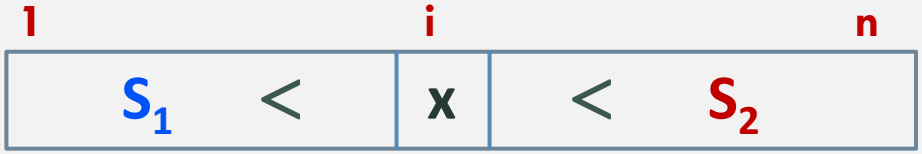
  Το βήμα αυτό της τεχνικής Δ-και-Β στον αλγόριθμο Select() δεν κάνει καμία ενέργεια !!!

Αλγόριθμος Select(S, k)

Αλγόριθμος

Input: ένα σύνολο S με n στοιχεία
Output: το k-τάξης στοιχείο του συνόλου S

1. Διαίρεση (divide)



2. Κατάκτηση (conquer)

- εάν $|S_1| = k-1 \Rightarrow$ return x
- εάν $|S_1| > k-1$ τότε Select(S_1, k)
- εάν $|S_1| < k-1$ τότε Select(S_2, k-$|S_1|$-1)

Αλγόριθμος Select(S, k)

Παράδειγμα

4	2	9	15	7	12	14	8	1	13	6	3	10	11

- Το 7-ής τάξης στοιχείο

Λύση !!!...
Η έξοδος του αλγόριθμου είναι το στοιχείο 8

1	2	3	4	6	7	8	9	10	11	12	13	14	15

6

7-ής τάξης στοιχείο

Αλγόριθμος Select(S, k)

Παράδειγμα

4	2	9	15	7	12	14	8	1	13	6	3	10	11

- Το 7-ής τάξης στοιχείο: $k = 7$
- Επιλογή pivot: $x = 4$

2	1	3	4	9	15	7	12	14	8	13	6	10	11

S_1 S_2

- εάν $|S_1| = k-1 = 6 \Rightarrow$ return x
- εάν $|S_1| > k-1 = 6$ τότε Select(S_1, k)
- εάν $|S_1| = 3 < k-1 = 6$ τότε Select(S_2, $k-|S_1|-1 = 7-3-1 = 3$)

Αλγόριθμος Select(S, k)

Παράδειγμα

9	15	7	12	14	8	13	6	10	11

- Το 3-ής τάξης στοιχείο : $k = 3$
- Επιλογή pivot: $x = 9$

7	8	6	9	15	12	14	13	10	11

S_1 S_2

- εάν $|S_1| = k-1 = 2 \Rightarrow$ return x
- εάν $|S_1| = 3 > k-1 = 2$ τότε Select(S_1, $k = 3$)
- εάν $|S_1| < k-1 = 2$ τότε Select(S_2, $k-|S_1|-1$)

Αλγόριθμος Select(S, k)

Παράδειγμα

7	8	6

- Το 3-ής τάξης στοιχείο : $k = 3$
- Επιλογή pivot: $x = 7$

6	7	8

S_1 S_2

- εάν $|S_1| = k-1 = 2 \Rightarrow$ return x
- εάν $|S_1| > k-1 = 2$ τότε Select(S_1, k)
- εάν $|S_1| = 1 < k-1 = 2$ τότε Select(S_2, $k-|S_1|-1 = 3-1-1 = 1$)

Αλγόριθμος Select(S, k)

Παράδειγμα

8

- Το 1-ής τάξης στοιχείο : $k = 1$
- Επιλογή pivot: $x = 8$

$\{\ \}$ 8 $\{\ \}$

S_1 S_2

Το 7-ής τάξης στοιχείο το 8 !!!

- εάν $|S_1| = 0 = k\text{-}1 \Rightarrow$ return $x = 8$
- εάν $|S_1| > k\text{-}1 = 0$ τότε Select(S_1, k)
- εάν $|S_1| < k\text{-}1 = 0$ τότε Select(S_2, k-$|S_1|$-1)

Αλγόριθμος Select(S, k)

Παράδειγμα

4	2	9	15	7	12	14	8	1	13	6	3	10	11

Το σύνολο εισόδου S

Πράγματι !!!

Select(S, 7) = 8

Το 7-ής τάξης στοιχείο το 8 !!!

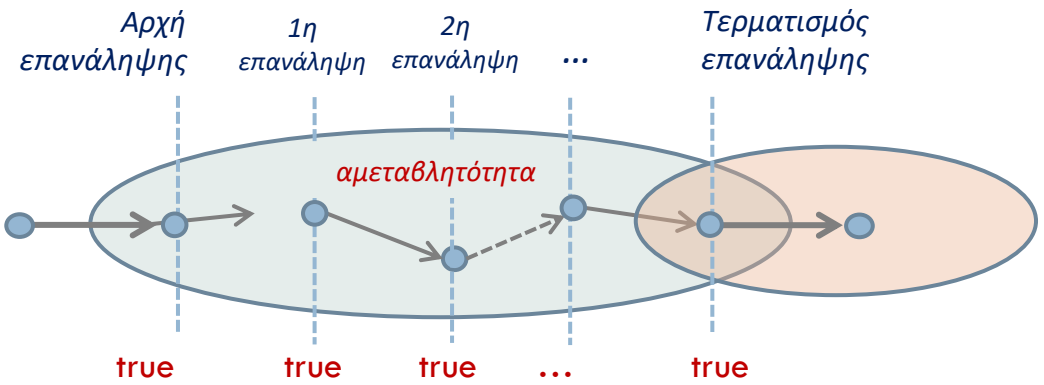
1	2	3	4	6	7	8	9	10	11	12	13	14	15

6

7-ής τάξης στοιχείο

Ορθότητα Αλγόριθμου Select(S, k)

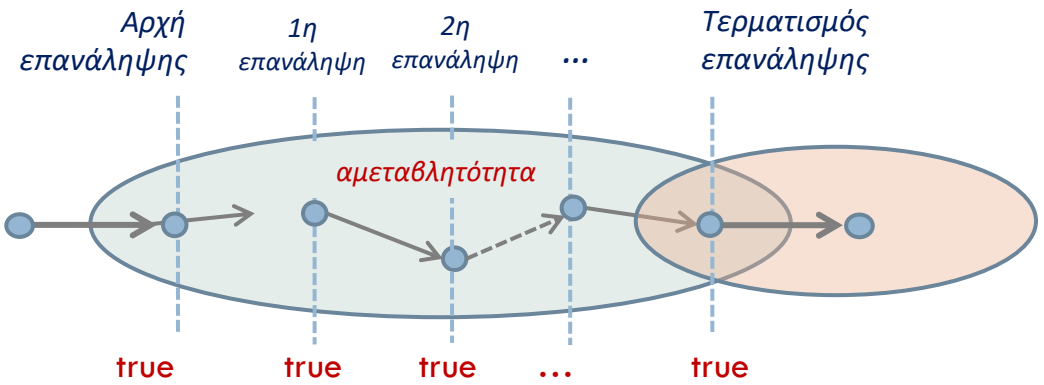
- Θα χρησιμοποιήσουμε τη τεχνική απόδειξης που ονομάζεται «*αμεταβλητότητα επανάληψης*» ή «*αναλλοίωτη συνθήκη επανάληψης*» (loop invariant) !!!



- Η αναλλοίωτη συνθήκη επανάληψης είναι μια *συνθήκη* οι οποία:
  - είναι **true** στην αρχή της επανάληψης,
  - παραμένει **true** σε κάθε εκτέλεση της επανάληψης, και
  - συνεχίζει να είναι **true** και μετά τον τερματισμό της επανάληψης.

Ορθότητα Αλγόριθμου Select(S, k)

- Θα χρησιμοποιήσουμε τη τεχνική απόδειξης που ονομάζεται «*αμεταβλητότητα επανάληψης*» ή «*αναλλοίωτη συνθήκη επανάληψης*» (loop invariant) !!!



- Η αποδεικτική τεχνική της αναλλοίωτης συνθήκης είναι ανάλογη με αυτή της μαθηματικής επαγωγής:

  Και οι δύο χρησιμοποιούν τη βασική περίπτωση (base case) μαζί με ένα επαγωγικό βήμα (inductive step) για να δείξουν ότι μια πρόταση είναι true για όλες τις περιπτώσεις !!!

Ορθότητα Αλγόριθμου Select(S, k)

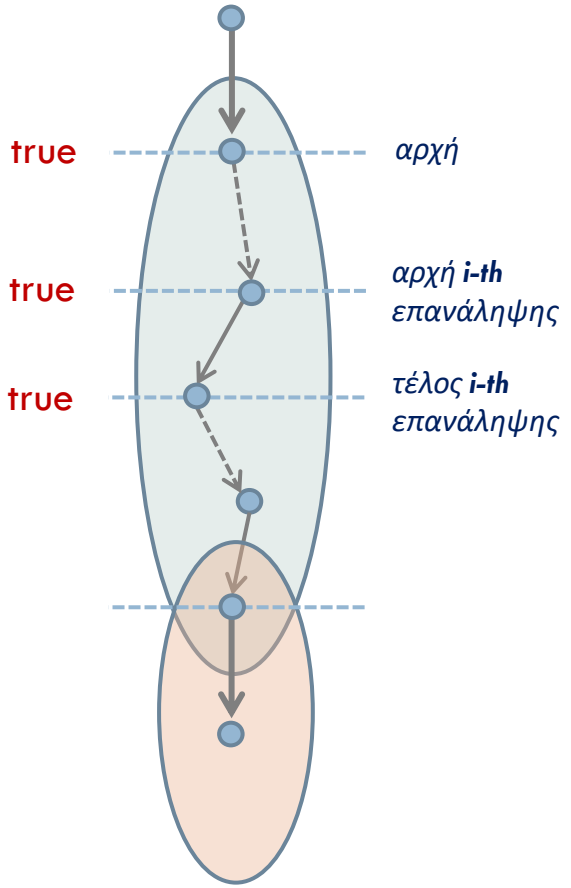
- Απόδειξη μιας *Αναλλοίωτης Συνθήκης Επανάληψης* (Proving a Loop Invariant) !!!

Βασική περίπτωση: Δείξε ότι η αναλλοίωτη συνθήκη είναι true *πριν την πρώτη* επανάληψη του βρόχου (ή κλήση της αναδρομικής συνάρτησης).

Επαγωγική υπόθεση: Έστω ότι η αναλλοίωτη συνθήκη είναι true *αμέσως πριν* από μια τυχαία, έστω την i-th, επανάληψη του βρόχου (ή από μια τυχαία κλήση της αναδρομής).

Επαγωγικό βήμα : Δείξε ότι η αναλλοίωτη συνθήκη είναι true *αμέσως μετά* το τέλος της i-th της επανάληψης του βρόχου (ή της i-th αναδρομικής κλήσης).

Και κάτι ακόμα...
Δείξε ότι ο βρόχος (ή η αναδρομική συνάρτηση) τερματίζει !!!



Ορθότητα Αλγόριθμου Select(S, k)

- Για τον αλγόριθμο Select(S, k) θεωρούμε την παρακάτω αναλλοίωτη συνθήκη επανάληψης:

> *Σε μια αναδρομική κλήση* Select(S′ , k′) *το τρέχον σύνολο επανάληψης* S′ μεγέθους n′ *αποτελείται από όλα τα στοιχεία του αρχικού* S *που έχουν τιμές μεταξύ* min(S′) *και* max(S′) *και η τρέχουσα τάξη επιλογής* k′ *έχει την ιδιότητα: το* k′ -μικρότερο στοιχείο του S′ *είναι το* k-μικρότερο στοιχείο του S.

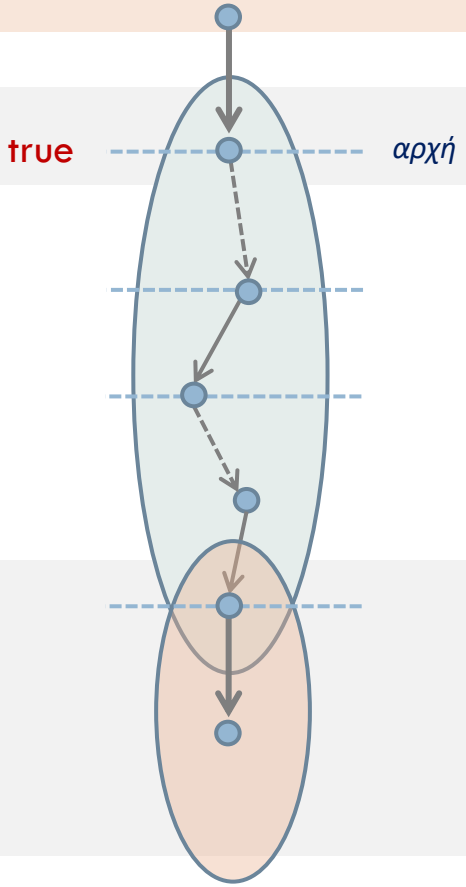
- Για να δείξουμε ότι η παραπάνω πρόταση αποτελεί αναλλοίωτη συνθήκη επανάληψης, αρκεί να δείξουμε ότι:

1. Είναι true στην αρχή της επανάληψης.
2. Παραμένει true σε κάθε εκτέλεσή της.
3. Συνεχίζει να είναι true και μετά τον τερματισμό της επανάληψης.

Ορθότητα Αλγόριθμου Select(S, k)

Απόδειξη αναλλοίωτης συνθήκης επανάληψης !

1. *Είναι true στην αρχή της επανάληψης.*



Αρχικά, το τρέχον σύνολο επανάληψης είναι το S μεγέθους n και η τρέχουσα τάξη επιλογής στοιχείου είναι k.

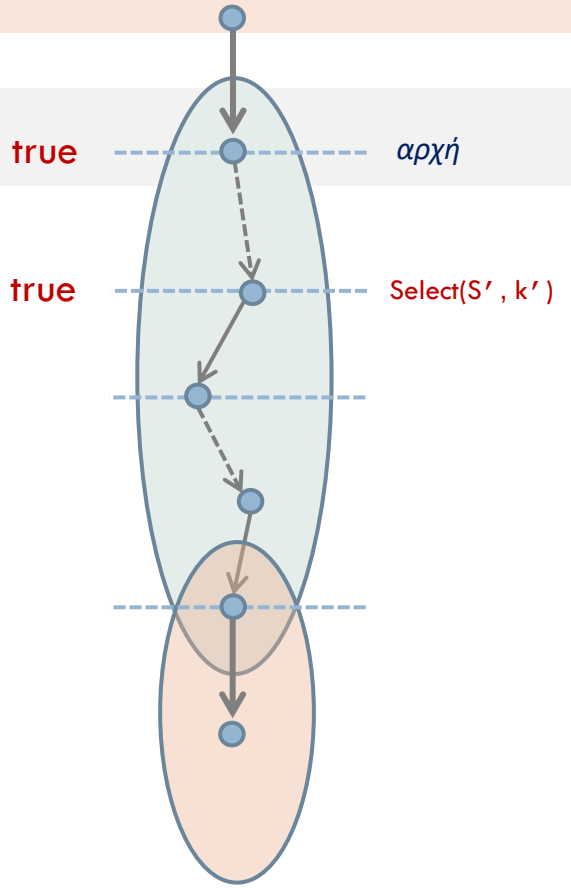
Επομένως η πρόταση είναι προφανώς true.

Σε μια αναδρομική κλήση Select(S, k) *το τρέχον σύνολο επανάληψης* S μεγέθους n *αποτελείται από όλα τα στοιχεία του αρχικού* S *που έχουν τιμές μεταξύ* min(S) *και* max(S) *και η τρέχουσα τάξη επιλογής* k *έχει την ιδιότητα: το* k-μικρότερο στοιχείο του S *είναι το* k-μικρότερο στοιχείο του S.

Ορθότητα Αλγόριθμου Select(S, k)

Απόδειξη αναλλοίωτης συνθήκης επανάληψης !

2. *Παραμένει* true *σε κάθε εκτέλεσή της.*



Έστω ότι η πρόταση είναι true στην αρχή μιας κλήσης Select(S′ , k′)

δηλ. το τρέχον σύνολο επανάληψης S′ μεγέθους n′ *αποτελείται από όλα τα στοιχεία του αρχικού* S *που έχουν τιμές μεταξύ* min(S′) *και* max(S′) *και το* k′ -μικρότερο στοιχείο του S′ *είναι το* k-μικρότερο στοιχείο του S

Τότε για τα σύνολα της διαμέρισης του S′ που προκύπτουν στο Βήμα 1, έστω S_1' και S_2' μεγέθους i -1 και n′ - i, ισχύει:

- το S_1' *αποτελείται από όλα τα στοιχεία του αρχικού* S *με τιμές μεταξύ* $\min(S_1')$ *και* $\max(S_1')$, και
- το ίδιο ισχύει και για το άλλο σύνολο της διαμέρισης S_2'

Ορθότητα Αλγόριθμου Select(S, k)

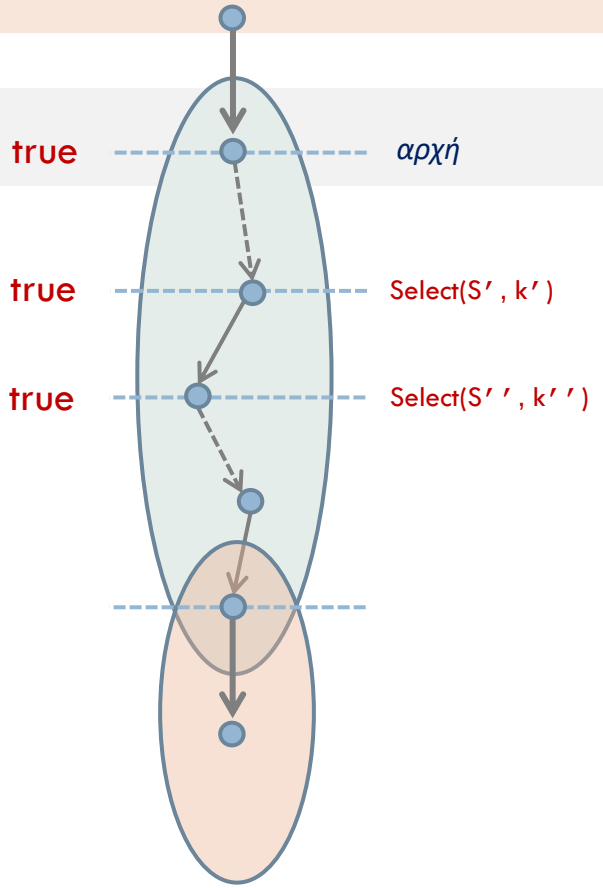
Απόδειξη αναλλοίωτης συνθήκης επανάληψης !

2. *Παραμένει true σε κάθε εκτέλεσή της.*

Εάν Select(S′′, k′′) είναι η επόμενη κλήση από την Select(S′, k′), όπου S′′ είναι είτε S_1' είτε S_2',

τότε είναι εύκολο να δειχθεί εξετάζοντας περιπτώσεις ότι το k′′-μικρότερο στοιχείο του S′′ ισούται με το k′-μικρότερο στοιχείο του S′,

το οποίο από υπόθεση ισούται με το k-μικρότερο στοιχείο του S.



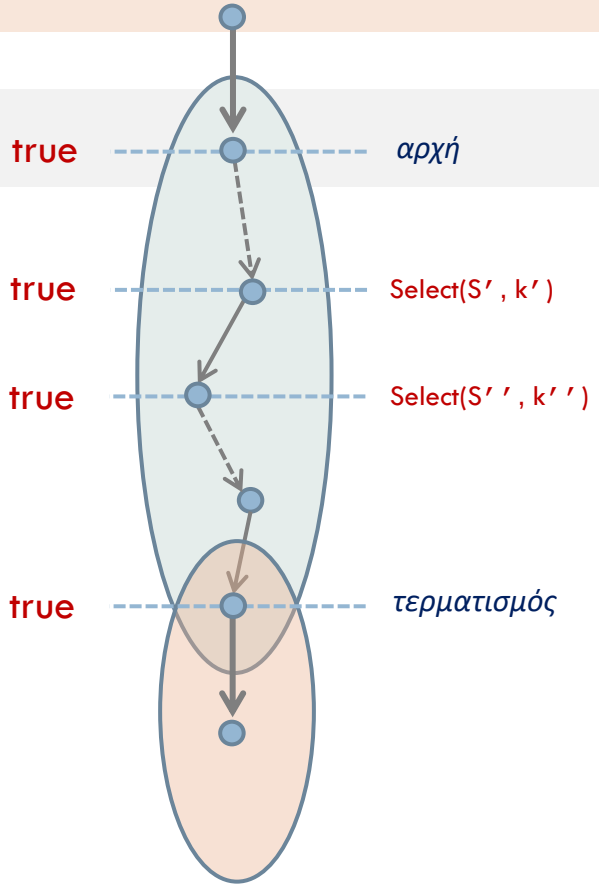
Ορθότητα Αλγόριθμου Select(S, k)

Απόδειξη αναλλοίωτης συνθήκης επανάληψης !

3. *Συνεχίζει να είναι true και μετά τον τερματισμό της.*

Επειδή δεν υπάρχει κάποια ειδική συνθήκη για το βήμα του τερματισμού (είναι ακριβώς το βήμα που τυχαίνει να διακόπτεται από την τιμή που επιστρέφεται),

η απόδειξη της *διατήρησης της αναλλοίωτης συνθήκης σε κάθε επανάληψη* αποδεικνύει τη διατήρηση της συνθήκης και μετά τον τερματισμό !!!



Ορθότητα Αλγόριθμου Select(S, k)

Απόδειξη αναλλοίωτης συνθήκης επανάληψης !

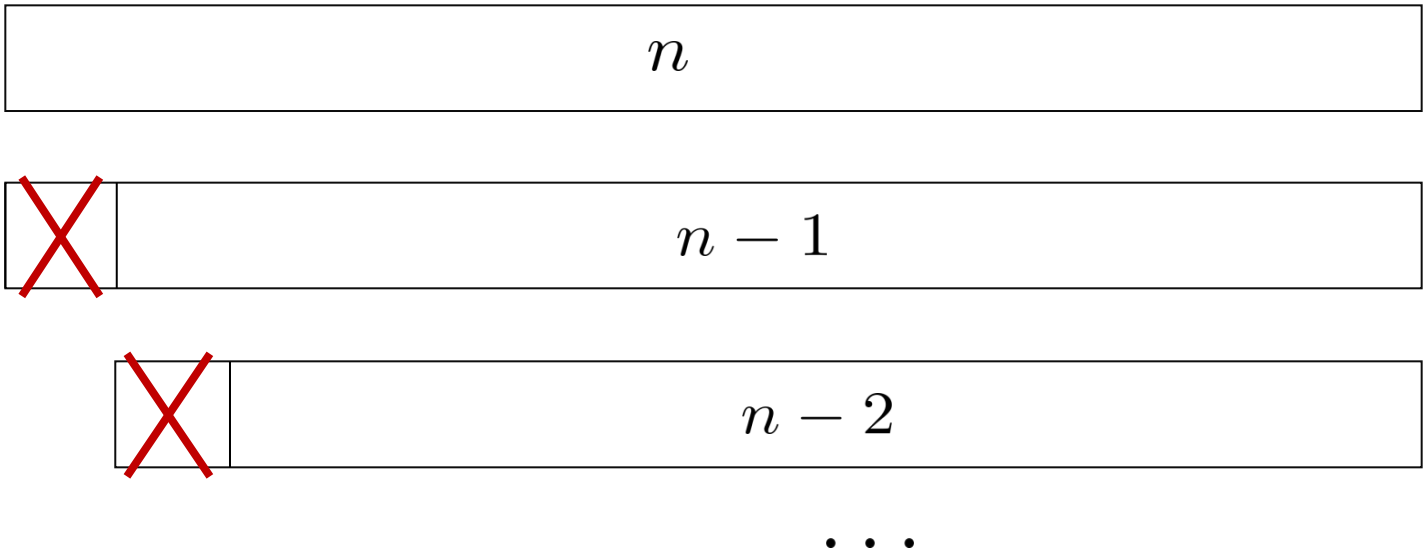
- Έχοντας δείξει την αναλλοίωτη συνθήκη επανάληψης (loop invariant), είναι εύκολο να δει κανείς ότι ο αλγόριθμος **Select(S, k)** είναι σωστός.
- Έστω Select(S′ , k′) είναι η τελευταία κλήση του αλγόριθμου. Τότε το Βήμα 2 επιστρέφει το k′ -μικρότερο στοιχείο του S′ , το οποίο από την αναλλοίωτη συνθήκη επανάληψης, είναι ίσο με το k-μικρότερο στοιχείο του αρχικού S. Επομένως, εάν ο αλγόριθμος τερματίσει, τότε θα τερματίσει με τη σωστή έξοδο.

- Μένει να δείξουμε ότι **ο αλγόριθμος *πάντα τερματίζει* !**

Πράγματι ισχύει, διότι το μέγεθος του τρέχοντος συνόλου σε κάθε αναδρομική κλήση του αλγόριθμου μειώνεται κατά τουλάχιστον 1 στοιχείο και επίσης η κλήση Select(S′ , k′) τερματίζει πάντα όταν το |S′ | = 1.

Πολυπλοκότητα Select(S, k)

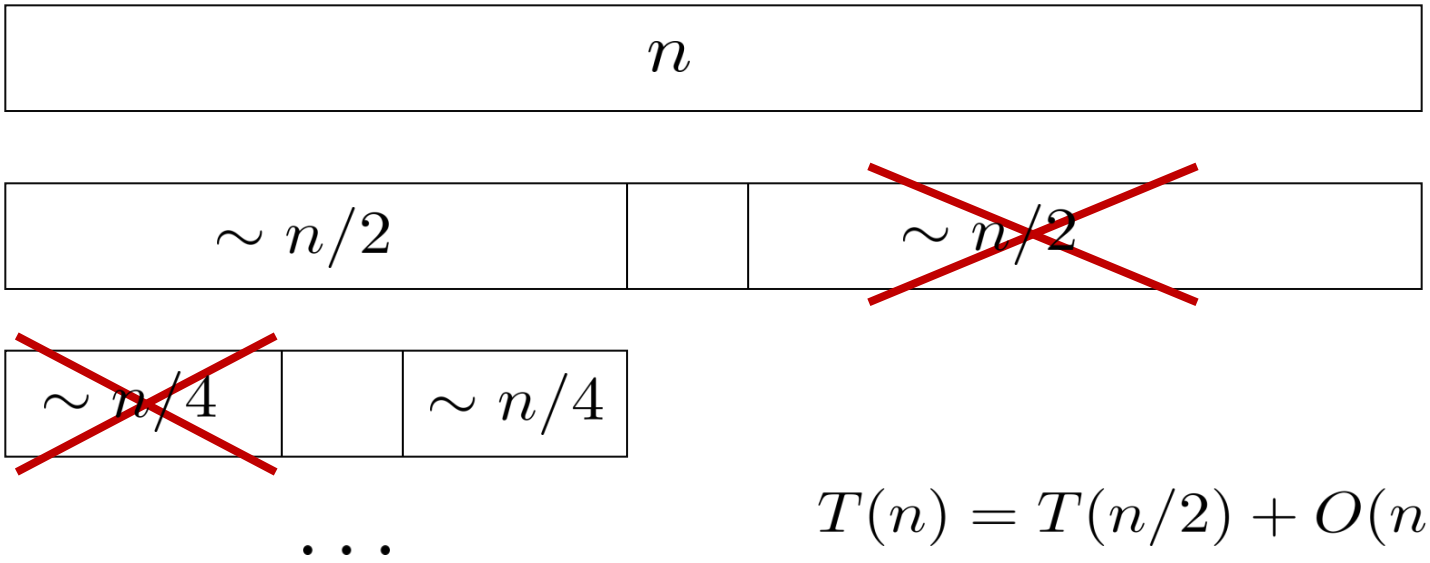
- **Χειρίστη Περίπτωση**



$$T(n) = \sum_{i=1}^{n-1}(n-i) \quad \Longrightarrow \quad T(n) = O(n^2)$$

Πολυπλοκότητα Select(S, k)

- **Ιδανική Περίπτωση**

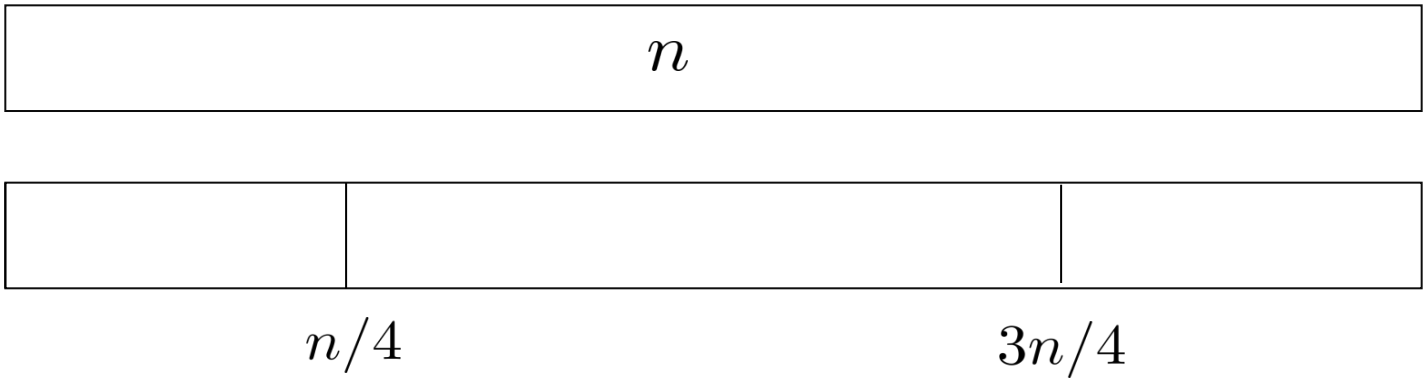


$$T(n) = T(n/2) + O(n)$$

$$T(n) = O(n)$$

Πολυπλοκότητα Select(S, k)

- **Μέση Περίπτωση**

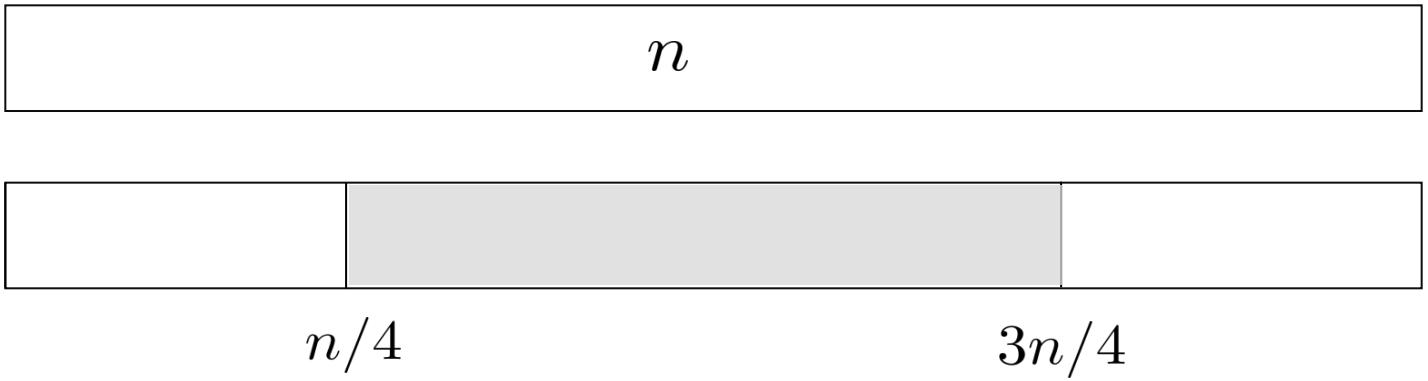


Ονομάζουμε "**καλή επιλογή**" Pivot εάν ανήκει μεταξύ των στοιχείων που βρίσκονται από το 25% έως το 75% των στοιχείων του S.

Τότε τα δύο τμήματα S_1 και S_2 θα έχουν μέγεθος το τουλάχιστον $3/4$ του αρχικού n.

Πολυπλοκότητα Select(S, k)

- **Μέση Περίπτωση**

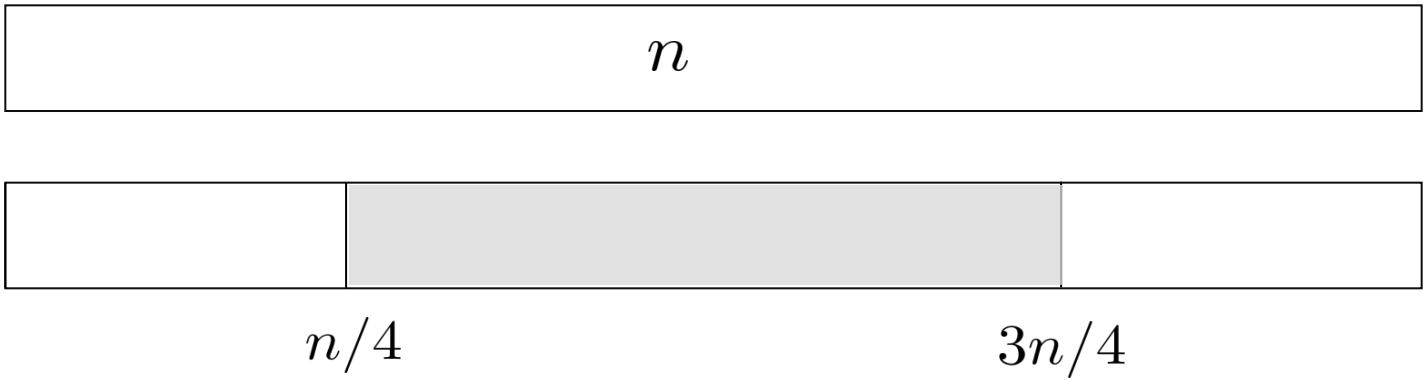


Το εύρος [1/4n, 3/4n] καλύπτει τα **½** στοιχεία.

Η πιθανότητα το Pivot $\in$ [1/4n, 3/4n] = ½.

Πολυπλοκότητα Select(S, k)

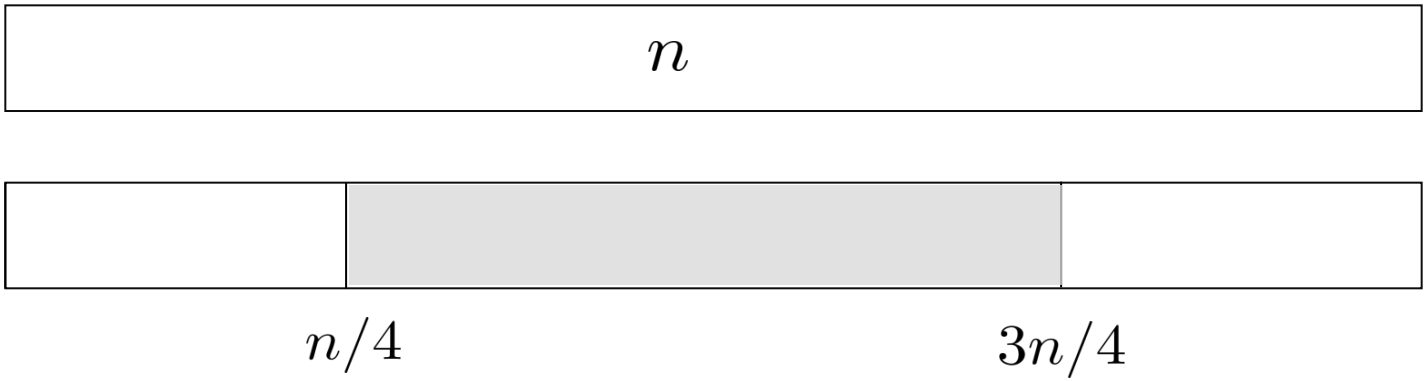
- **Μέση Περίπτωση**



- ✓ Επομένως, κατά μέσο όρο, μετά από 2 τυχαίες επιλογές Pivot θα έρθει καλή επιλογή!!!
- ✓ Η «καλή επιλογή» θα μειώσει το μέγεθος του προβλήματος το πολύ σε $3/4$ του αρχικού.

Πολυπλοκότητα Select(S, k)

- **Μέση Περίπτωση**

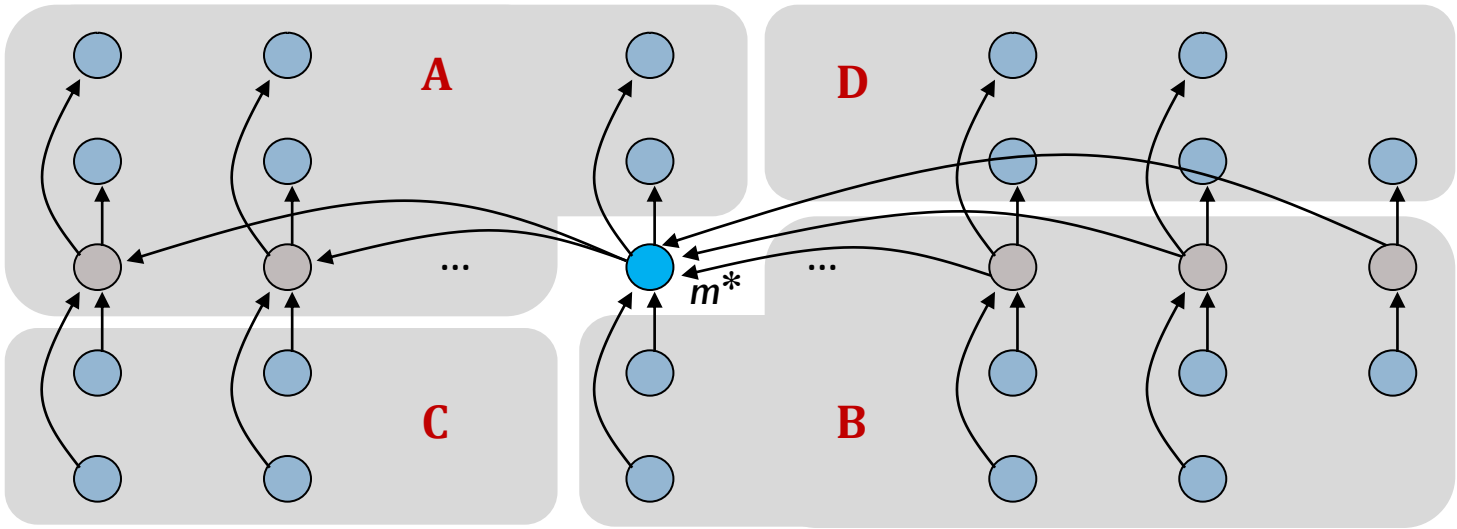


- ✓ Άρα, ο μέσος χρόνος του απλού αλγόριθμου περιγράφεται από την αναδρομική σχέση:

$$T(n) \leq T(3n/4) + O(n) \implies T(n) = O(n)$$

Αλγόριθμος Quick-Select(S, k)

Divide & Conquer

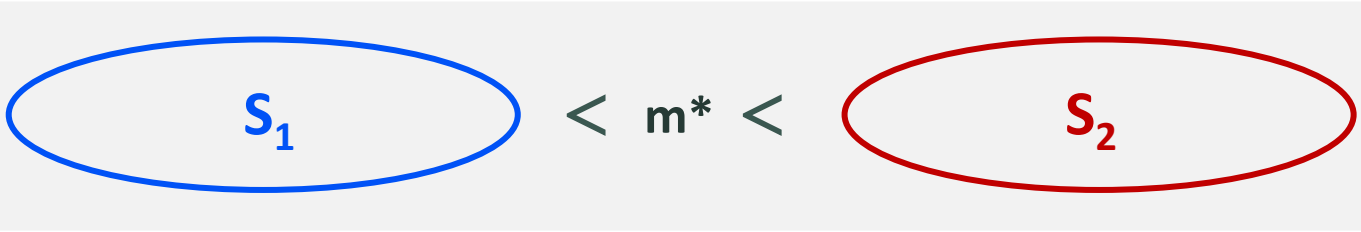


Αλγόριθμος Quick-Select(S, k)

Ιδέα Αλγόριθμου !!!

Input: ένα σύνολο S με n στοιχεία
Output: το k-τάξης στοιχείο του συνόλου S

1. Διαίρεση (divide)



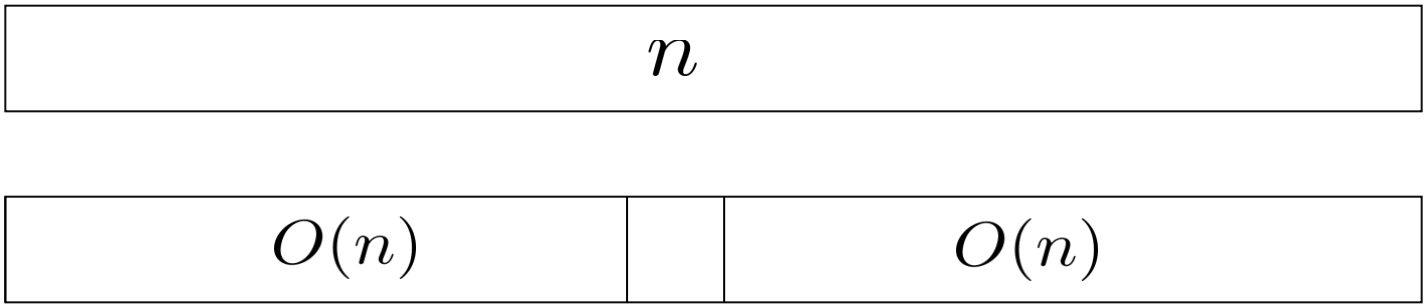
Select() vs Quick-Select()

Διαφορετική επιλογή των στοιχείων διάσπασης **x** και **m***

Αλγόριθμος Quick-Select(S, k)

Ιδέα Αλγόριθμου !!!

- ✓ Θέλουμε με μια κατάλληλη επιλογή του Pivot σε κάθε επανάληψη να επιτυγχάνουμε:



- ✓ Τότε θα δείξουμε ότι στη χειρίστη περίπτωση ο αλγόριθμος απαιτεί χρόνο:

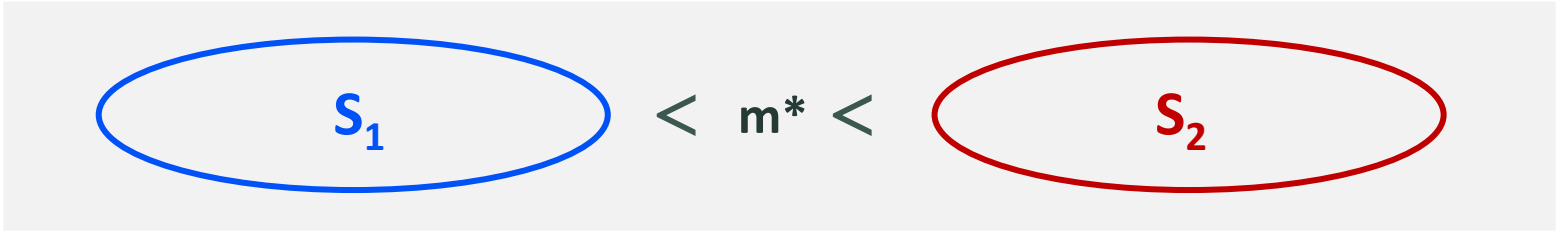
$$T(n) = O(n)$$

Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

Input: ένα σύνολο S με n στοιχεία
Output: το k-τάξης στοιχείο του συνόλου S

1. Διαίρεση (divide)



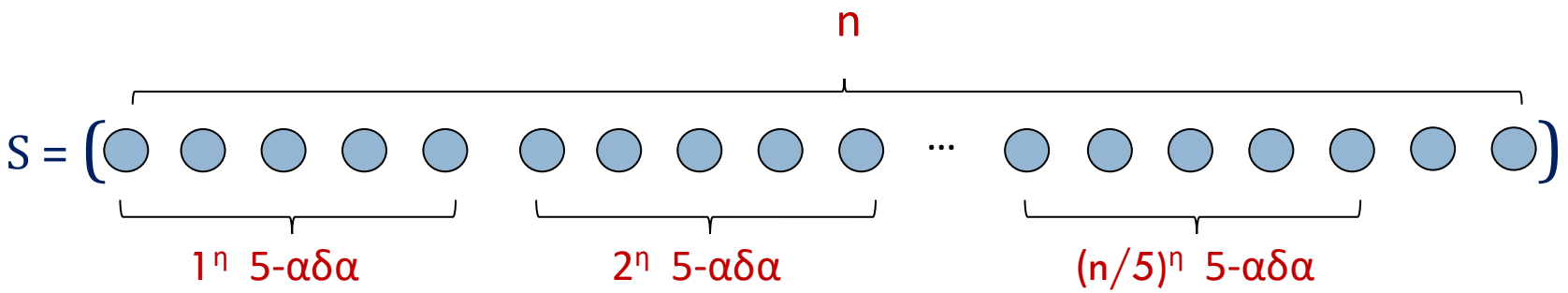
2. Κατάκτηση (conquer)

- εάν $|S_1| = k-1 \Rightarrow$ return m*
- εάν $|S_1| > k-1$ τότε Select(S_1, k)
- εάν $|S_1| < k-1$ τότε Select(S_2, k-$|S_1|$-1)

Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

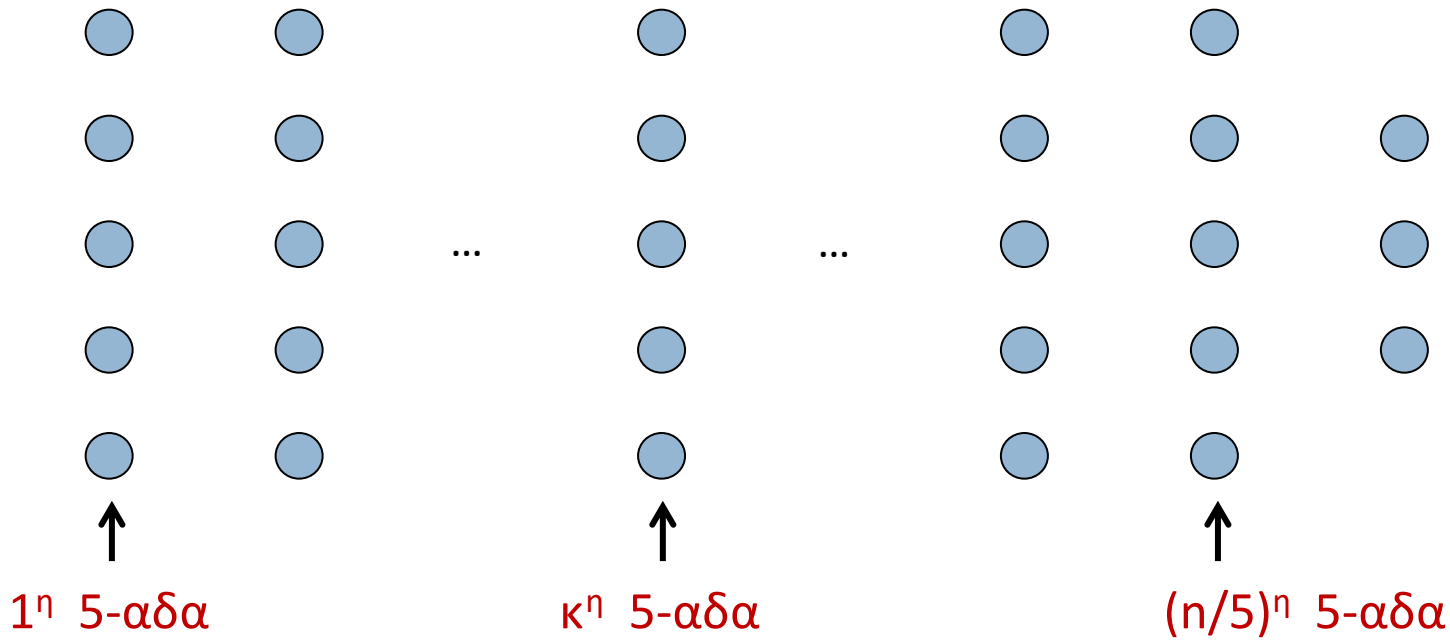
Βήμα **1.** Χωρίζουμε την ακολουθία S μήκους n σε n/5 υποακολουθίες μήκους 5 η κάθε μία (εκτός πιθανώς από την τελευταία).



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

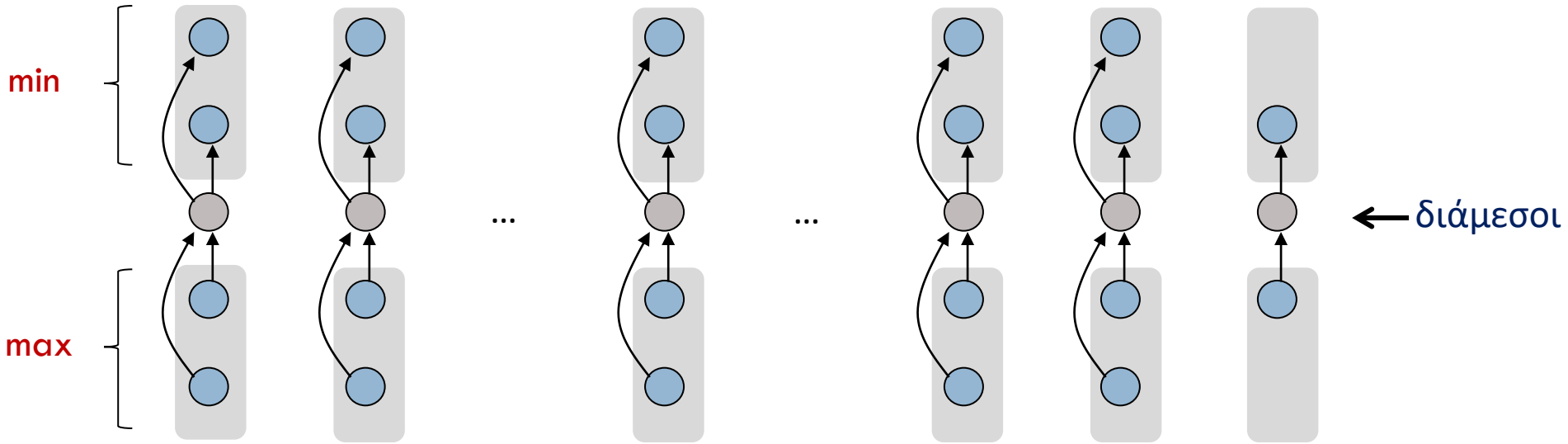
Βήμα **2.** Βρίσκουμε το μέσο κάθε μιας από τις $n/5$ υποακολουθίες μήκους 5 (εκτός πιθανώς από την τελευταία).



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

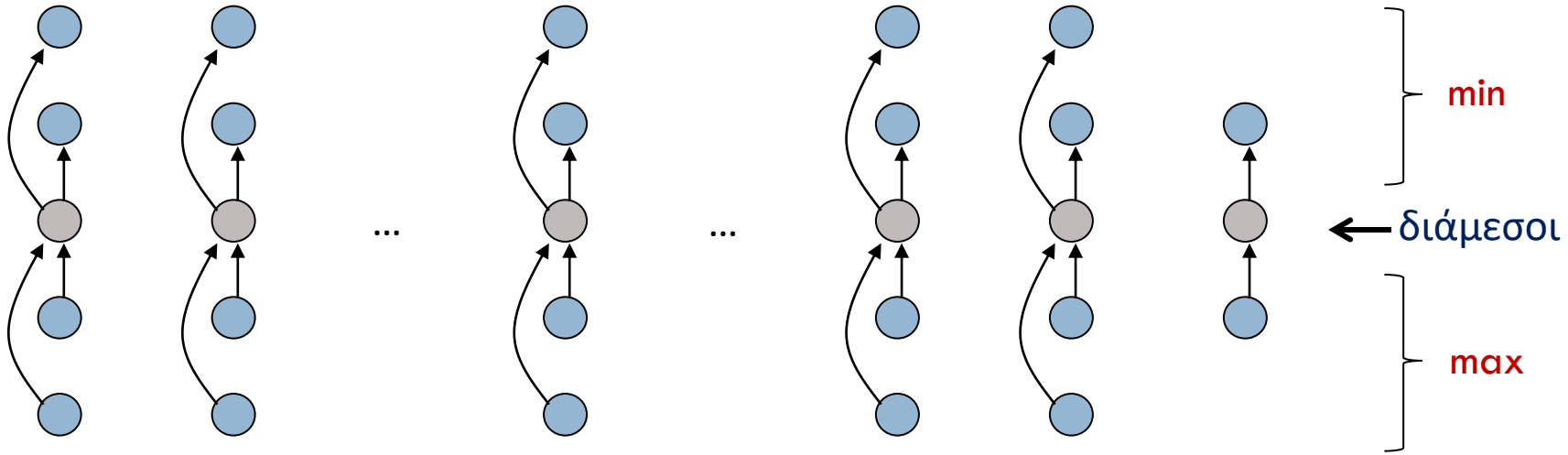
Βήμα **2.** Βρίσκουμε το μέσο κάθε μιας από τις $n/5$ υποακολουθίες μήκους 5 (εκτός πιθανώς από την τελευταία).



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

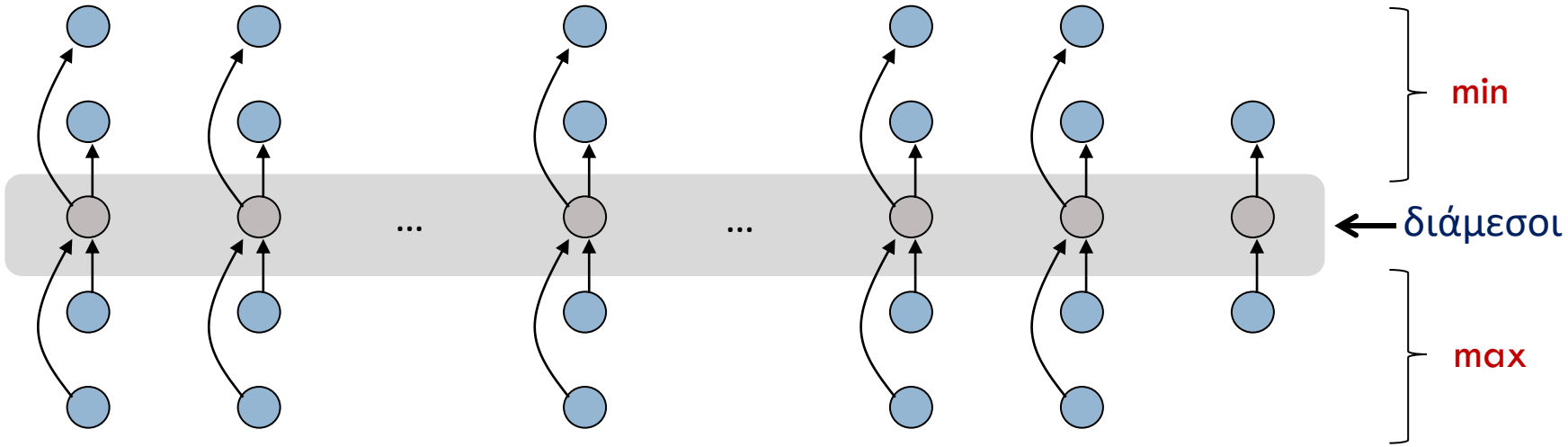
Βήμα **2.** Βρίσκουμε το μέσο κάθε μιας από τις $n/5$ υποακολουθίες μήκους 5 (εκτός πιθανώς από την τελευταία).



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

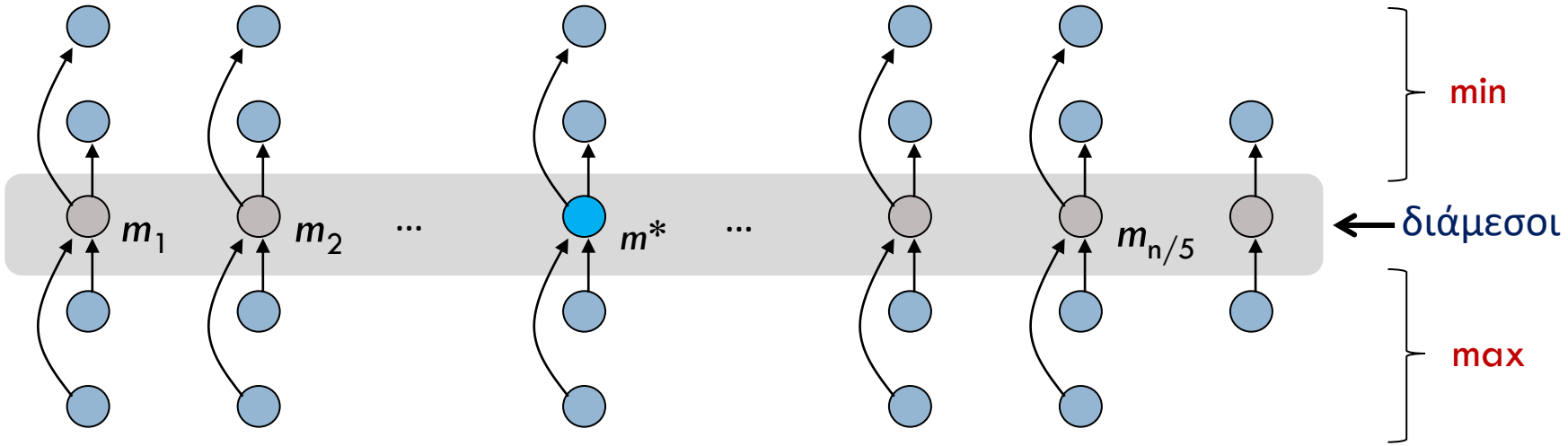
Βήμα **3.** Χρησιμοποιώντας την Quick-Select αναδρομικά, βρίσκουμε τη διάμεσο m* των διαμέσων των n/5 υποακολουθιών.1



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

Βήμα **3.** Χρησιμοποιώντας την Quick-Select αναδρομικά, βρίσκουμε τη διάμεσο m* των διαμέσων των n/5 υποακολουθιών.1

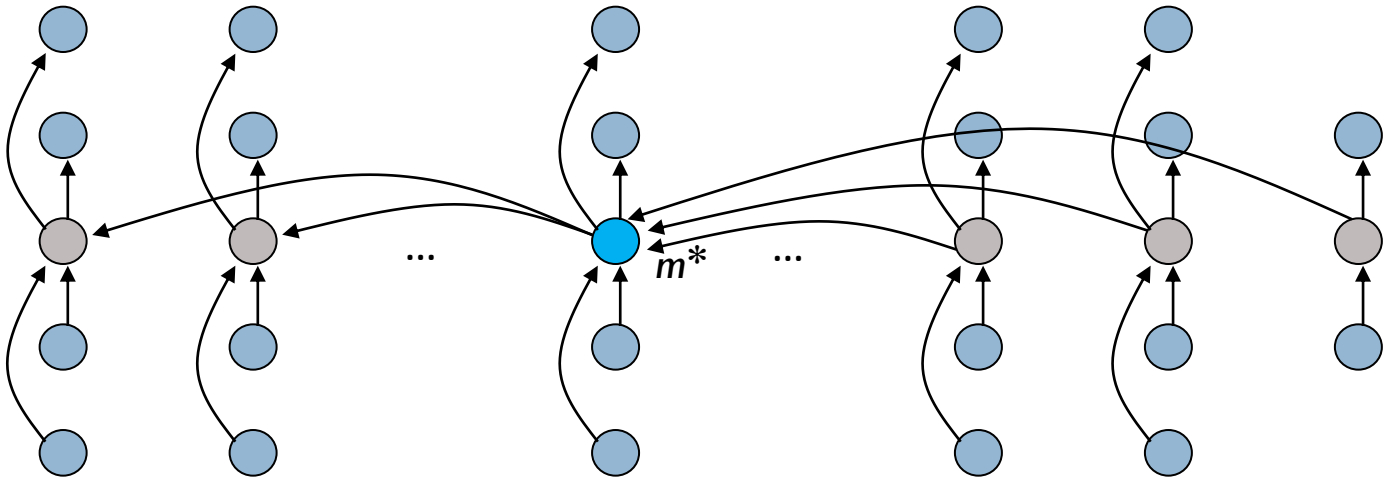


$$\{m_1, m_2, \ldots\} \leq m^* \leq \{\ldots, m_{(n/5)-1}, m_{n/5}\}$$

Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

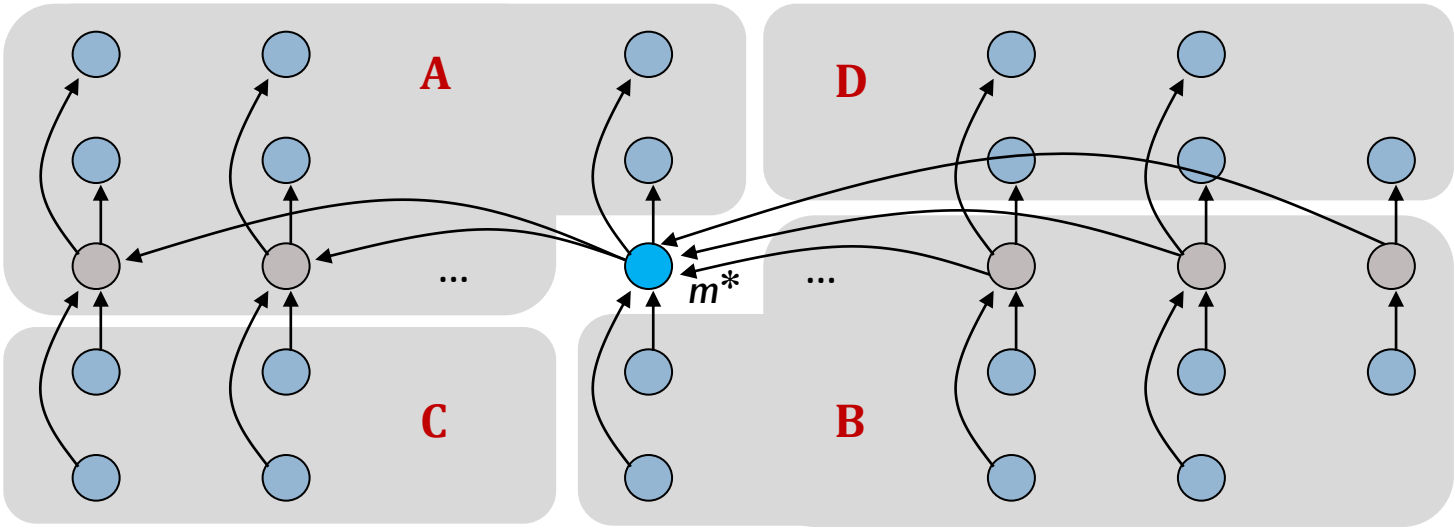
Βήμα **3.** Χρησιμοποιώντας την Quick-Select αναδρομικά, βρίσκουμε τη διάμεσο m* των διαμέσων των n/5 υποακολουθιών.1



Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

Βήμα **4.** Διαμερίζουμε την ακολουθία S με pivot τη διάμεσο m*.



$A \leq m^* \leq B$ $C \; ? \; m^* \; ? \; D$

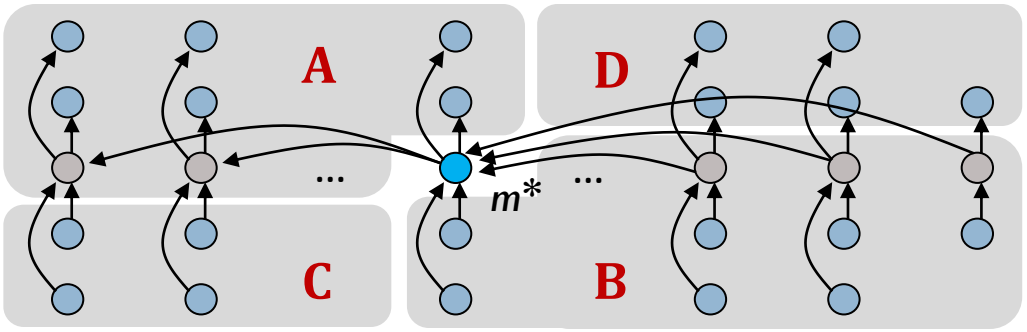
Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

Βήμα **4.** Διαμερίζουμε την ακολουθία S με pivot τη διάμεσο m*.

$S_1 \leftarrow A$

$S_2 \leftarrow B$



Συγκρίνουμε τα στοιχεία των χωρίων **C** και **D** με το m*

$\mathbf{A} \leq m^* \leq \mathbf{B}$ $\qquad$ $\mathbf{C}\ ?\ m^*\ ?\ \mathbf{D}$

$$S_1 \leftarrow A \cup \{x \mid x \in C \cup D \ \&\ x < m^*\}$$

$$S_2 \leftarrow B \cup \{x \mid x \in C \cup D \ \&\ x > m^*\}$$

Αλγόριθμος Quick-Select(S, k)

Αλγόριθμος

Βήμα **5.** Έλεγχος και αναδρομική επίλυση:

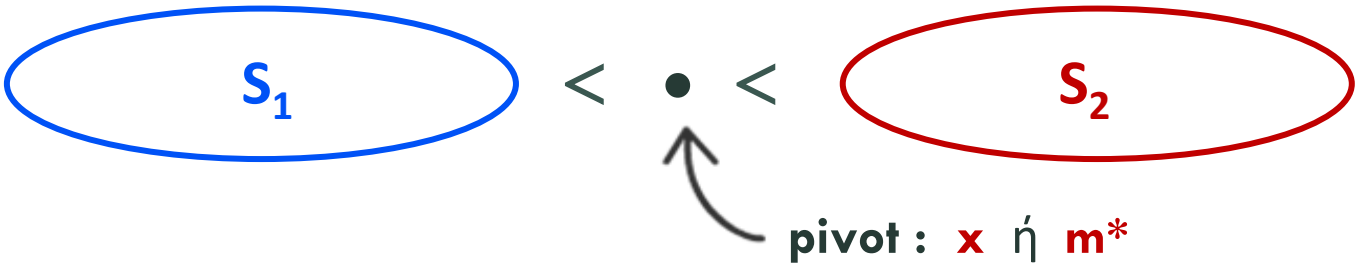
- εάν $|S_1| = k-1 \Rightarrow$ return m^*
- εάν $|S_1| > k-1$ τότε Quick-Select(S_1, k)
- εάν $|S_1| < k-1$ τότε Quick-Select(S_2, $k-|S_1|-1$)

Ορθότητα Αλγόριθμου Quick-Select(S, k)

- Προκύπτει άμεσα από την λειτουργία του αλγόριθμου και από την ορθότητα του αλγορίθμου **Select(S, k)** !!!

Θυμίζουμε !!!

Η μόνη διαφορά των αλγορίθμων **Select(S, k)** και **Quick-Select(S, k)** είναι η διαφορετική επιλογή των στοιχείων διάσπασης (pivots) **x** και **m***, αντίστοιχα.

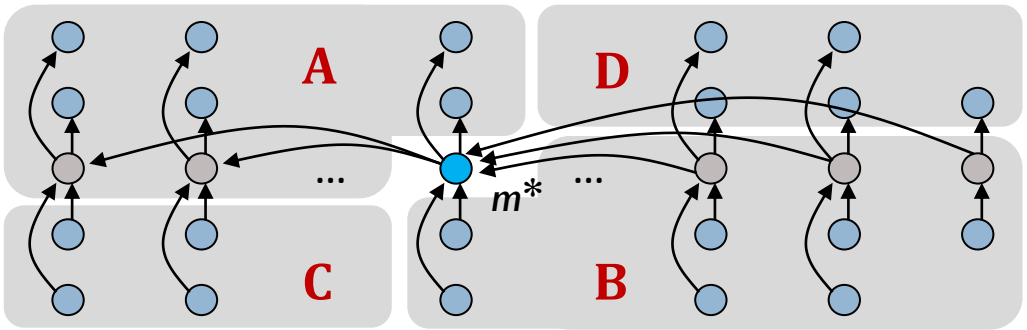


Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- Θα δείξουμε ότι ο αλγόριθμος είναι γραμμικός στο μέγεθος της εισόδου, δηλαδή,

$$T(n) = O(n)$$

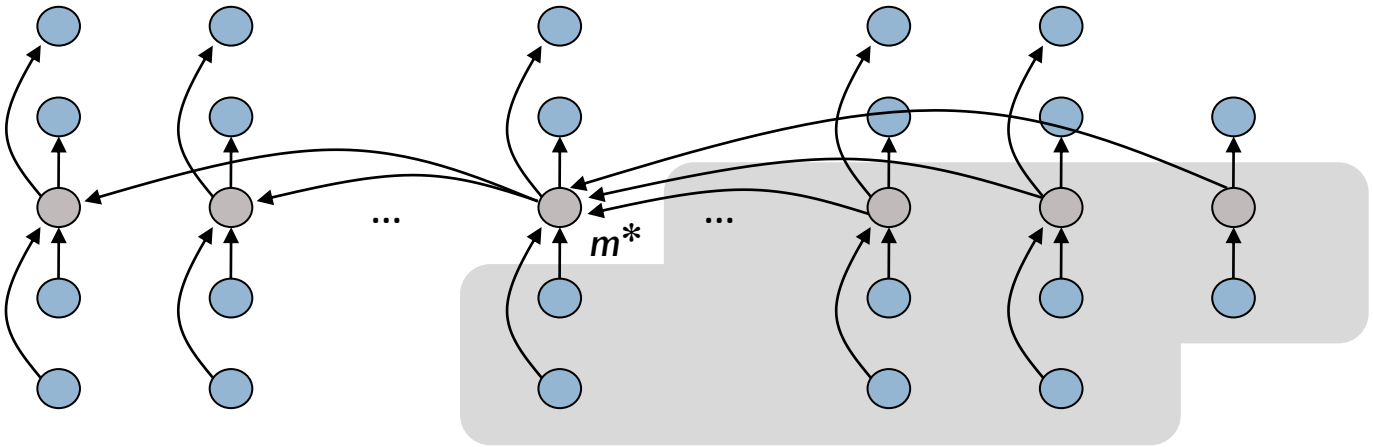
για κάθε είσοδο μήκους $n > 1$.



Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

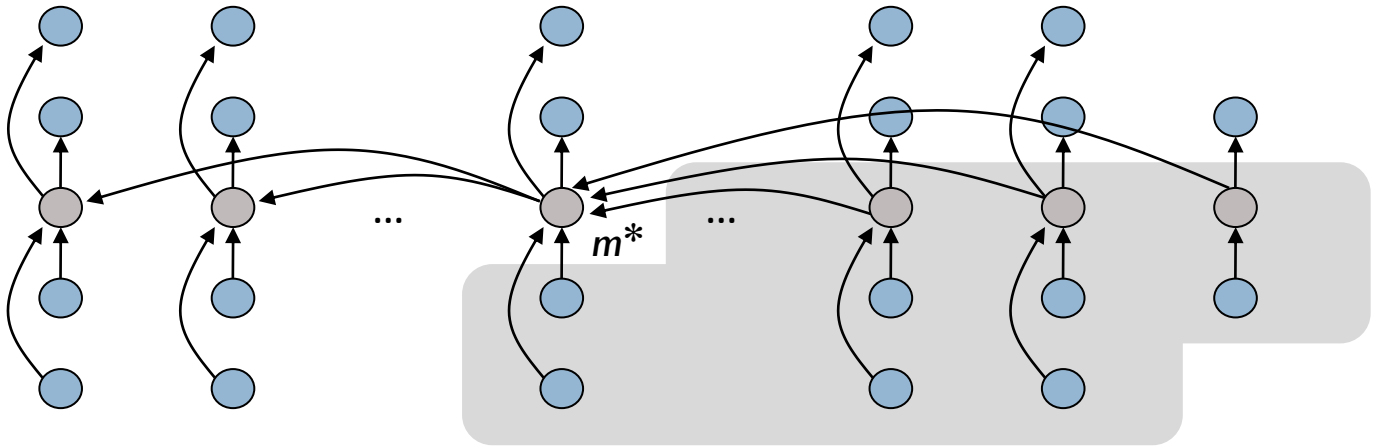
  Αρχικά προσδιορίζουμε ένα κάτω φράγμα για το πλήθος των στοιχείων που είναι μεγαλύτερα από το m*.



Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Τουλάχιστον οι μισές 5-αδες (από τις $\lceil n/5 \rceil$ συνολικά) συνεισφέρουν 3 στοιχεία που είναι μεγαλύτερα του m*.

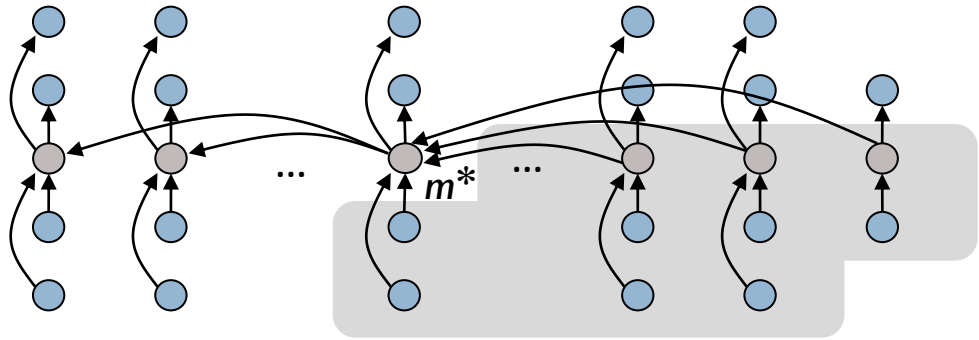


Εντός από την 5-αδα που έχει λιγότερα στοιχεία όταν το n δεν διαιρείται ακριβώς δια του 5 και από την 5-αδα που περιέχει το m*.

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

Επομένως το πλήθος των στοιχείων που είναι μεγαλύτερα από το στοιχείο m* είναι



$$3(\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \rceil - 2) \geq \frac{3n}{10} - 6$$

Αντίστοιχα, το πλήθος των στοιχείων που είναι μικρότερα του m* είναι τουλάχιστον

$$3n/10 - 6$$

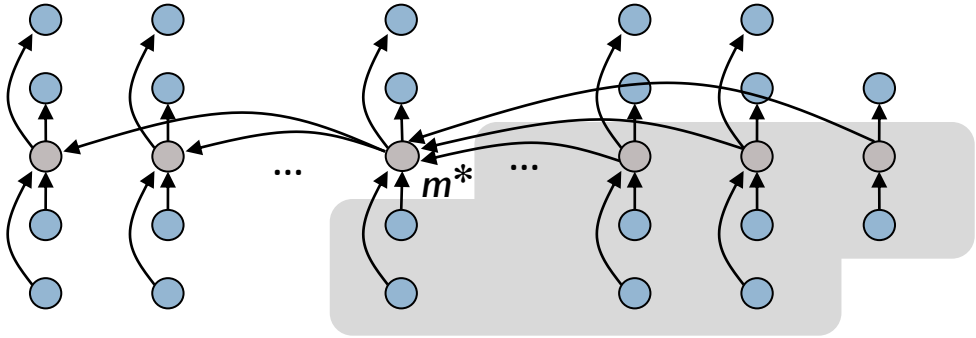
Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Άρα, στην αναδρομική κλήση στο Βήμα 5, ο αλγόριθμος θα δεχθεί ως είσοδο το πολύ

$$7n/10 + 6$$

στοιχεία στη χειρότερη περίπτωση.



Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Τώρα μπορούμε να διατυπώσουμε μια αναδρομική σχέση για το χρόνο εκτέλεσης

  $$T(n)$$

  του αλγόριθμου στη χειρίστη περίπτωση.

  - Τα Βήματα 1, 2 και 4 απαιτούν χρόνο $O(n)$
  - Το Βήμα 3 απαιτεί χρόνο $T(\lceil n/5 \rceil)$
  - Το Βήμα 5 απαιτεί χρόνο $T(7n/10 + 6)$

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Δεχόμαστε (θα αποσαφηνιστεί σύντομα) ότι για οποιαδήποτε είσοδο με στοιχεία <140 απαιτείται χρόνος

$$O(1)$$

Με την παραδοχή αυτή παίρνουμε:

$$T(n) = \begin{cases} O(1) & n < 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & n \geq 140 \end{cases}$$

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Επομένως, ισχύει:

$$T(n) = \begin{cases} O(1) & n < 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & n \geq 140 \end{cases}$$

  Θα δείξουμε ότι ο χρόνος εκτέλεσης $T(n)$ είναι γραμμικός, δηλαδή,

$$T(n) \leq cn$$

  για κάποια μεγάλη σταθερά c και για όλα τα $n > 0$

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Αρχικά υποθέτουμε ότι

  $$T(n) \leq cn$$

  για κάποια επαρκώς μεγάλη σταθερά c και για όλα τα n < 140
  (η παραδοχή αυτή ισχύει εφόσον η σταθερά c είναι αρκετά μεγάλη).

  Αντικαθιστώντας, έχουμε

  $$T(n) \leq c\lceil n/5 \rceil + c(7n/10 + 6) + an$$

  Όπου *α* μια σταθερά τέτοια ώστε η συνάρτηση που αντιπροσωπεύει ο όρος *α*n να φράσσεται άνω από την ποσότητα O(n) για όλα τα n > 0.

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Επομένως,

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\leq cn/5 + c + 7cn/10 + 6c + an \\ &\leq 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an). \end{aligned}$$

  Ισχύει: $T(n) \leq cn$ εάν

$$-cn/10 + 7c + an \leq 0 \qquad (1)$$

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Η ανισότητα

$$-cn/10 + 7c + an \leq 0 \qquad (1)$$

  είναι ισοδύναμη με την ανισότητα

$$c \geq 10a(n/(n-70)) \qquad (2)$$

  όταν $n > 70$.

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

Από την ανισότητα:

$$c \geq 10a(n/(n-70)) \qquad (2)$$

δεδομένου ότι υποθέσαμε πως $n \geq 140$, έχουμε ότι:

$$n/(n-70) \leq 2$$

και επομένως εάν επιλέξουμε

$$c \geq 20a$$

η ανισότητα (2) ικανοποιείται.

Πολυπλοκότητα Αλγόριθμου Quick-Select(S, k)

- **Ανάλυση Αλγόριθμου**

  Δεν υπάρχει τίποτε το ιδιαίτερο όσον αφορά τη σταθερά 140.

  Μπορούμε να την αντικαταστήσουμε με οποιοδήποτε ακέραιο > 70 και να επιλέξουμε τη σταθερά c ανάλογα.

  Δείξαμε ότι για τον Quick-Select(S, k) ισχύει:

$$T(n) \leq cn$$

  Άρα, ο αλγόριθμος είναι γραμμικός στο μέγεθος της εισόδου, δηλ.

$$T(n) = O(n)$$

  για κάθε είσοδο μήκους $n > 1$. □

Ευχαριστώ για τη Συμμετοχή σας !!!

Σταύρος Δ. Νικολόπουλος

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Ιωαννίνων

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