Quick Sort

- θ το πρώτο στοιχείο (οδηγός) του πίνακα για ταξινόμηση
- Διαίρεση του πίνακα σε δύο ζώνες

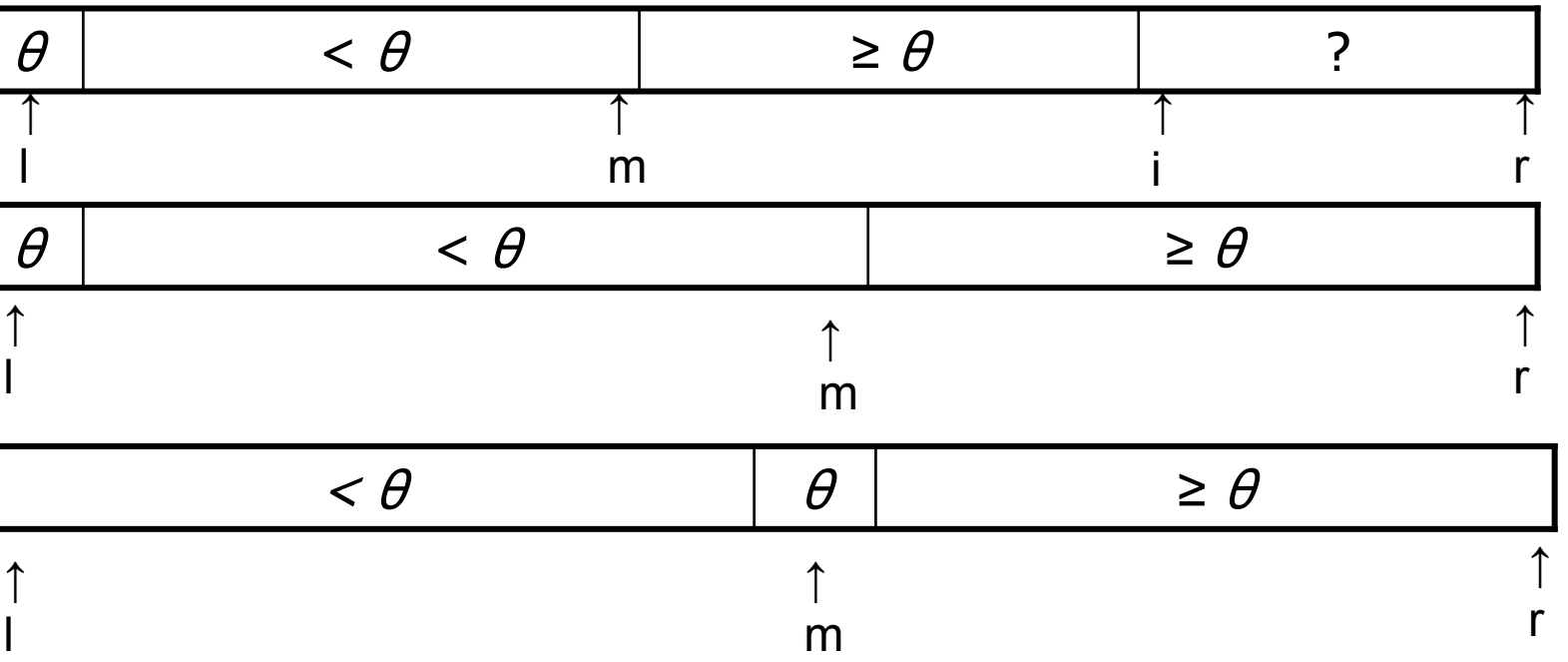
  αρχή του πίνακα: στοιχεία ≤ θ

  τέλος του πίνακα: στοιχεία ≥ θ

  θ στην οριστική του θέση
- Αναδρομική κλήση του αλγόριθμου σε κάθε μία από τις ζώνες όσο δεν είναι ελαττωμένες σε ένα στοιχείο

Quicksort: Οριστική θέση οδηγού θ

- Έστω l (αριστερός δείκτης) και r (δεξιός δείκτης)
- Έστω δύο δείκτες i και m τέτοιοι ώστε για κάθε στιγμή έχουμε: $a_j < \theta$, $l < j \leq m$ και $a_j \geq \theta$, $m < j < i$



Quicksort: παράδειγμα

a: $\underline{6_1}$ 9_2 2_3 7_4 4_5 5_6 8_7

↓

↓

a: x x x $\underline{6}$ x x x

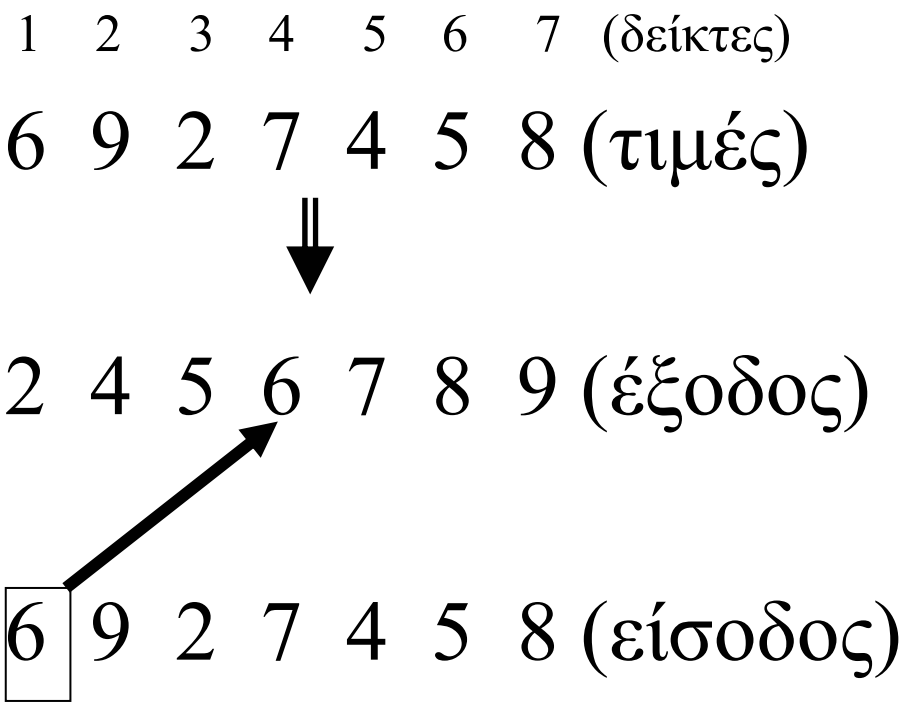
"Εύρεση της τελικής θέσης του οδηγού στην αρχή του αλγόριθμου"

Quick Sort: αλγόριθμος

```
Quick Sort (A, l, r)
    if l < r then
        m=Partition (A, l, r)
        Quick Sort (A, l, m-1)
        Quick Sort (A, m+1 , r)
    end if
```

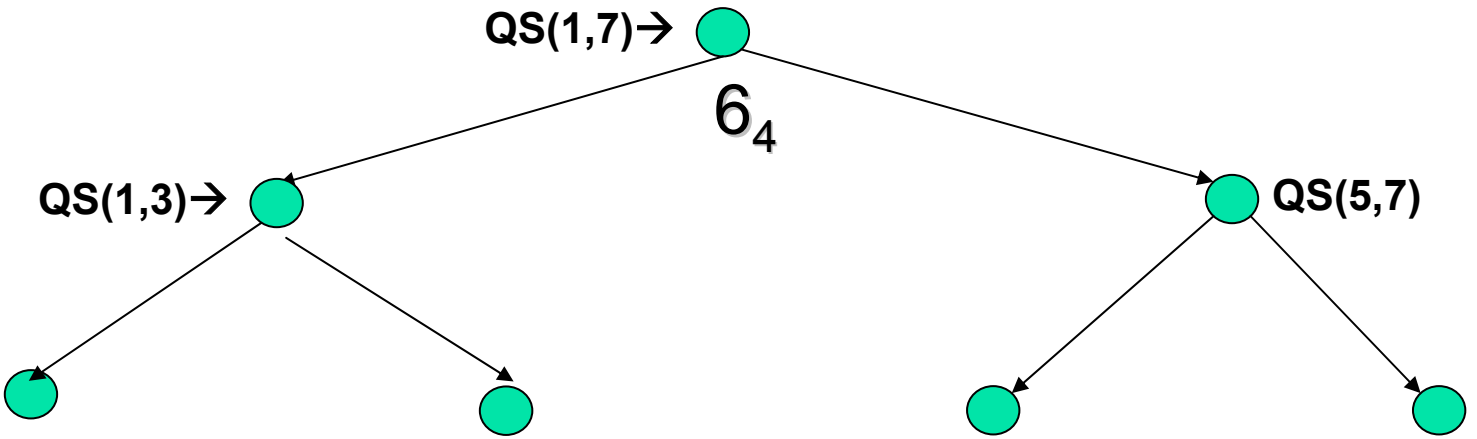
*****

Παράδειγμα


1 2 3 4 5 6 7 (δείκτες)
6 9 2 7 4 5 8 (τιμές)
2 4 5 6 7 8 9 (έξοδος)
6 9 2 7 4 5 8 (είσοδος)

Quick Sort

Παράδειγμα: 6_1 9_2 2_3 7_4 4_5 5_6 8_7



Αλγόριθμος Partition

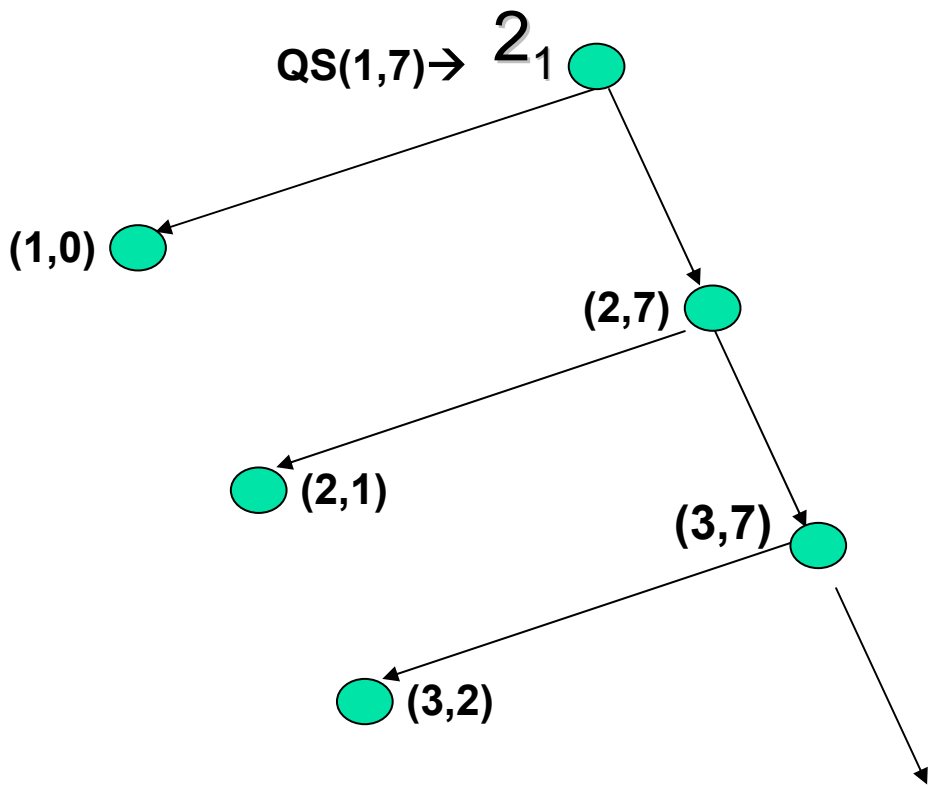
```
Partition (A, l, r)

θ = a[l], m = l
for i = l+1 to r do
      if a[i] < θ
        m = m+1
        swap (A[i], A[m])
     end if
end for
swap (A[l], A[m])
return m /* οριστική θέση οδηγού θ */
```

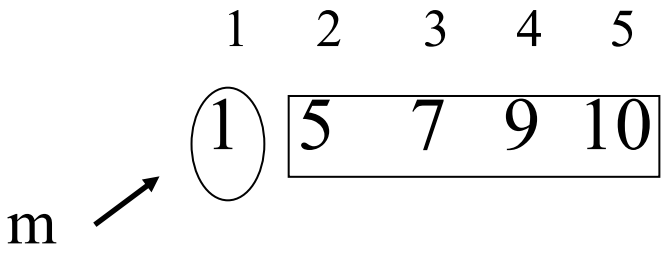
****

Quick Sort - worst case

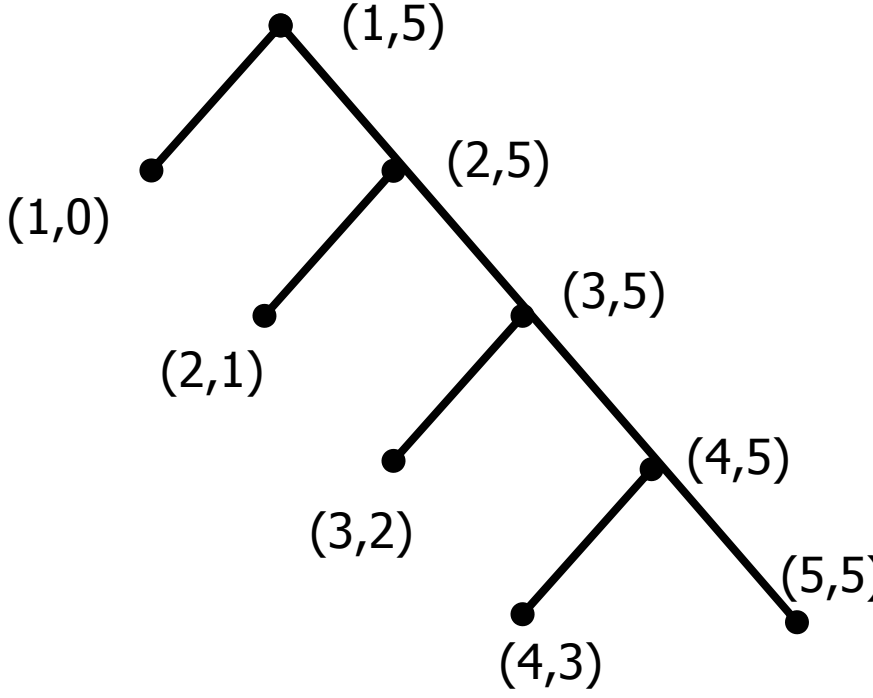
Παράδειγμα: 2_1 9_2 17_3 24_4 32_5 40_6 58_7



Quick Sort - worst case



$$T_n = \begin{cases} (n-1) + T_{n-1} & n \geq 1 \\ 0 \quad \delta\iota\alpha\phi. \end{cases}$$



Πολυπλοκότητα Quick Sort

1. **Χείριστη περίπτωση**
   αύξουσες ή φθίνουσες ακολουθίες αριθμών
   →"διαμερίσεις εκφυλισμένες"

$$T_n = \begin{cases} n-1+T_{n-1} \text{ αν } n \geq 1 \\ 0 \text{ διαφορετικά} \end{cases}$$

Επομένως $O(n^2)$

Πολυπλοκότητα Quick Sort

2. **Κατά μέσο όρο**

Όλες οι θέσεις για την οριστική θέση του στοιχείου *θ* είναι ισοπίθανες $c = \frac{1}{n}$

Έχουμε $T_n = n-1+ \frac{1}{n} \sum_{k=1}^{n}(T_{k-1}+ T_{n-k})$

και λόγω συμμετρίας $T_n = n-1+ \frac{2}{n} \sum_{k=1}^{n} T_{k-1}$

Τελικά $T_n = O(n\log n)$

Μέση πολυπλοκότητα Quick Sort

Θεωρούμε όλες τις θέσεις για την τοποθέτηση του οδηγού **θ** ισοπίθανες (= $\frac{1}{n}$). Ο μέσος αριθμός συγκρίσεων T_n για n≥2 υπολογίζεται ως εξής:

$T_0=T_1=0$ (θεωρούμε τις συγκρίσεις μεταξύ των στοιχείων μόνο).

$$T_n=n+1+\frac{1}{n}\sum_{k=1}^{n}(T_{k-1}+T_{n-k})$$ και λόγω συμμετρίας

$$T_n=n+1+\frac{2}{n}\sum_{k=1}^{n}T_{k-1} \rightarrow nT_n=n^2+n+2\sum_{k=1}^{n}T_{k-1}$$

Μέση πολυπλοκότητα Quick Sort

Για n-1 έχουμε:

$$T_{n-1}=n+\frac{2}{n-1}\sum_{k=1}^{n}T_{k-1} \rightarrow (n-1)T_{n-1}=n^2-n+2\sum_{k=1}^{n}T_{k-1}$$

Αφαιρώντας κατά μέλη, έχουμε μετά την απλοποίηση:

$$nT_n=(n+1)T_{n-1}+2n$$

Μέση πολυπλοκότητα Quick Sort

Διαιρώντας με n(n+1) έχουμε:

$$\frac{T_n}{n+1} = \frac{T_{n-1}}{n} + \frac{2}{n+1} = \frac{C_2}{3} + \sum_{k=3}^{n} \frac{2}{k+1}$$

Προσεγγίζοντας:

$$\frac{T_n}{n+1} \approx 2 \sum_{k=1}^{n} \frac{1}{k} \approx 2 \int_{1}^{n} \frac{1}{x} dx = 2\ln(n)$$

Τελικά: $T_n \approx 1{,}38n\log n$

Εναλλακτική λύση διαμέλισης

- Η προηγούμενη λύση δεν είναι συμμετρική
- Μια άλλη παρουσίαση του Quick Sort συνίσταται στο να εντοπίσουμε την τελική θέση του θ με δύο δείκτες που ξεκινούν από το 1 και το n και οι οποίοι συγκλίνουν προς την τελική θέση του θ.
- Χρησιμοποίηση "φρουρών" στα αριστερά και στα δεξιά του πίνακα. Μπορούμε να θέσουμε ένα μικρότερο στοιχείο από το θ στα αριστερά και ένα μεγαλύτερο στα δεξιά.

Αλγόριθμος Quick Sort

```
void QuickSort(int g, int d) {
    int i, j, θ, t;
    if (g < d) {
       θ = a[d]; i = g-1; j = d;
       do {
          do
             ++i;
          while (a[i] < θ);
          do
             --j;
          while (a[j] > θ);
          t = a[i]; a[i] = a[j]; a[j] = t;
       } while (j > i);
       a[j] = a[i]; a[i] = a[d]; a[d] = t;
       QuikSort (g, i-1);
       QuikSort (i+1, d);
    }
}*******
```