Αλγόριθμοι και Πολυπλοκότητα
2ο Σύνολο Ασκήσεων

Ονομ/νυμο: Μανάβης Βασίλειος-Ιωάννης
Α.Μ.: 1115201000136

1.

Αλγόριθμος 1:

Ο αλγόριθμος διαιρεί το πρόβλημα σε 3 υποπροβλήματα, διάστασης $\frac{2n}{3}$ τα οποία επιλύει αναδρομικά και συνδυάζει τις λύσεις σε χρόνο O(1). Άρα η αναδρομική εξίσωση είναι η $T(n) = 3T\left(\frac{2n}{3}\right) + O(1) = 3T\left(\frac{n}{3/2}\right) + O(1)$. Θα βρούμε τη πολυπλοκότητα με τη βοήθεια του Master Theorem. Έχουμε a=3, b=1.5 και έστω ε=1.5>0. Επομένως $n^{\log_b(a-\varepsilon)} = n^{\log_{1.5}(3-1.5)} = n^{\log_{1.5}(1.5)} = n^1 = n \Rightarrow 1 = O(n)$ Άρα $T(n) = \Theta\left(n^{\log_{1.5}3}\right) \approx \Theta\left(n^{2.7}\right)$

Αλγόριθμος 2: (Η άσκηση είναι **ημιτελής**! Η συνέχεια είναι στο τέλος.)

Ο αλγόριθμος διαιρεί το πρόβλημα σε 2 υποπροβλήματα, διάστασης $\frac{n}{2}$ τα οποία επιλύει αναδρομικά και συνδυάζει τις λύσεις σε χρόνο O(nlogn). Άρα η αναδρομική εξίσωση είναι η

$$T(n)=2T\left(\frac{n}{2}\right)+O(n\,logn)=.....=2^k T\left(\frac{n}{2^k}\right)+n\sum_{i=0}^{k-1}\log\left(\frac{n}{2^i}\right)=2^k T\left(\frac{n}{2^k}\right)+ n\sum_{i=0}^{k-1}(logn - i) =$$

$$2^k T\left(\frac{n}{2^k}\right)+n\left(k\,logn-\sum_{i=0}^{k-1} i\right)\overset{k=logn}{=} 2^{logn}T(1)+n\left(\log^2 n-\sum_{i=0}^{logn-1} i\right)=a2^{logn}+n\left(\log^2 n-\frac{1}{2}logn(logn-1)\right)$$

$$= a2^{logn}+n\left(\log^2 n-\frac{1}{2}logn(logn-1)\right)=a2^{logn}+n\left(\log^2 n-\frac{1}{2}\log^2 n+\frac{1}{2}\right)=a2^{logn}+n\left(\frac{1}{2}\log^2 n+\frac{1}{2}\right)$$

2.

a)

$T(n)=5T\left(\frac{n}{2}\right)+n\,logn$

Εφαρμόζουμε τη 1η περίπτωση του Master Theorem. Έχουμε a=5, b=2 και έστω ε=1>0. Τότε $n^{\log_b(a-\varepsilon)} = n^{\log_2(5-1)} = n^{\log_2 4} = n^2 \Rightarrow n\,logn = O(n^2)$ Άρα $T(n) = \Theta\left(n^{\log_2 5}\right)$

$\Rightarrow T(n) = \Theta\left(n^{2,32}\right)$

b)

$T(n)=3T(n-1)+1$

$T(n-1)=3T(n-2)+1$

Άρα $T(n)=3[3T(n-2)+1]+1 = 3^2T(n-2)+3^1+3^0 = ... = 3^k T(n-k)+\sum_{i=0}^{k-1} 3^i =$

$3^k T(n-k)+\frac{3^n-1}{2}$ Για k = n έχουμε $3^n T(0)+\frac{3^n-1}{2} = 3^n+\frac{3^n}{2}-\frac{1}{2} = \frac{3}{2}3^n-\frac{1}{2} = \Theta(3^n)$

c)

$T(n)=4\mathrm{T}\left(\frac{n}{2}\right)+n^2$

Εφαρμόζουμε τη 2η περίπτωση του Master Theorem. Έχουμε a=4 και b=2. Τότε $n^{\log_b a}=n^{\log_2 4}=n^2 \Rightarrow n^2=\Theta(n^2)$ Άρα $T(n)=\Theta\left(n^{\log_2 4}\cdot\log n\right)=\Theta\left(n^2\cdot\log n\right)$

Τη χείριστη πολυπλοκότητα έχει ο
αλγόριθμος Β γιατί είναι εκθετική και επομένως ο αλγόριθμος Β αποκλείεται. Ο αλγόριθμος Α έχει πολυπλοκότητα $\Theta\left(n^{2,32}\right)=\Theta\left(n^2\cdot n^{0,32}\right)$ και ο αλγόριθμος C έχει πολυπλοκότητα $\Theta\left(n^2\cdot\log n\right)$

Ας λογαριθμήσουμε τις ποσότητες $n^2\cdot n^{0,32}$ και $n^2\cdot\log n$.
Έχουμε λοιπόν:

$$\log\left(n^2\cdot n^{0,32}\right)\circ\log\left(n^2\cdot\log n\right)\Rightarrow 2\,logn+0.32\,logn\circ 2\,logn+\log(logn)\Rightarrow 0.32\,logn\circ\log(logn)$$

Θα υπολογίσω το όριο

$$\lim_{n\to+\infty}\frac{0.32\,logn}{\log(logn)}\overset{de\,L'Hospital}{=}\lim_{n\to+\infty}\frac{0.32\cdot\frac{1}{n}}{\frac{1}{logn}\cdot\frac{1}{n}}=\lim_{n\to+\infty}\frac{\frac{0.32}{1}}{\frac{1}{logn}}=\lim_{n\to+\infty}0.32\,logn=\infty$$

Οπότε $\log(logn)=O(0.32\,logn)$ *και* $0.32\,logn=\Omega(\log(logn))$ Άρα μπορούμε να πούμε ότι η πολυπλοκότητα $n^2\cdot\log n$ είναι καλύτερη από την $n^{2,32}$. Άρα θα επέλεγα τον αλγόριθμο C.

3.

a)

$$T(n)=3\mathrm{T}\left(\frac{n}{2}\right)+n=3\left[3\mathrm{T}\left(\frac{n}{2^2}\right)+\frac{n}{2}\right]+n=3^2T\left(\frac{n}{2^2}\right)+n+3\frac{n}{2}=.....=3^kT\left(\frac{n}{2^k}\right)+n\sum_{i=0}^{k-1}\left(\frac{3}{2}\right)^i$$

Θέλω $\frac{n}{2^k}=1\Rightarrow n=2^k\Rightarrow logn=k$ Για $k=logn$ έχω $T(n)=3^{logn}T(1)+n\sum_{i=0}^{logn-1}\left(\frac{3}{2}\right)^i$ T(1) = σταθερό = α. Άρα $T(n)=\alpha\cdot3^{logn}+n\sum_{i=0}^{logn-1}\left(\frac{3}{2}\right)^i=\alpha\cdot3^{logn}+2n\left[\left(\frac{3}{2}\right)^{logn}-1\right]=\alpha\cdot3^{logn}+2\mathrm{n}\left(\frac{3}{2}\right)^{logn}-2\mathrm{n}$

b)

$$T(n)=T\left(n^{\frac{1}{2}}\right)+1=\left[T\left(n^{\frac{1}{2^2}}\right)+1\right]+1=.....=T\left(n^{\frac{1}{2^k}}\right)+k$$

Θέλω $n^{\frac{1}{2^k}}=2\Rightarrow\log\left(n^{\frac{1}{2^k}}\right)=\log 2\Rightarrow\frac{1}{2^k}logn=\log 2\Rightarrow logn=2^k\Rightarrow\log(logn)=\log(2^k)\Rightarrow$ Άρα έχω

$$\log(logn)=k\log 2\Rightarrow\log(logn)=k$$

$$T(n)=T(2)+\log(logn)=a+\log(logn)$$

c)

$$T(n)=2\mathrm{T}\left(n^{\frac{1}{2}}\right)+1=2\left[2T\left(n^{\frac{1}{2^2}}\right)+1\right]+1=2^2T\left(n^{\frac{1}{2^2}}\right)+1+2^1=.....=2^kT\left(n^{\frac{1}{2^k}}\right)+\sum_{i=0}^{k-1}2^i$$

Θέλω $n^{\frac{1}{2^k}}=2\Rightarrow\log(logn)=k$ (Αποδείχτηκε στο προηγούμενο ερώτημα)

Άρα έχω $T(n)=2^k T\left(n^{\frac{1}{2^k}}\right)+\sum_{i=0}^{k-1} 2^i = 2^{\log(logn)}T(2)+\sum_{i=0}^{\log(logn)-1} 2^i = a\cdot 2^{\log(logx)}+2^{\log(logx)}-1 =$

$(a+1)2^{\log(logx)}-1$

d)

$$T(n)=T(n-1)+n=T(n-2)+n-1+n=\ldots\ldots\ldots=T(n-k)+\sum_{i=0}^{k-1} n-i=T(n-k)+kn-\sum_{i=0}^{k-1} i$$

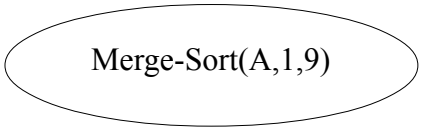
Θέλω $(n-k)=2\Rightarrow n-2=k$ Άρα έχω $T(n)=T(2)+(n-2)n-\sum_{i=0}^{n-2-1} i = a+n^2-2n-\sum_{i=0}^{n-3} i =$

$a+n^2-2n-\frac{1}{2}[(n-3)(n-2)]=a+n^2-2n-\frac{1}{2}(n^2-2n-3n+6)=a+n^2-2n-\frac{n^2}{2}+\frac{5n}{2}-3=$

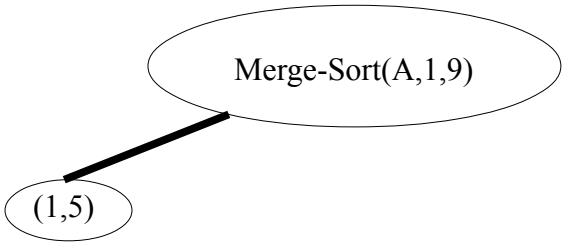
$a+\frac{n^2}{2}+\frac{n}{2}-3$

4.

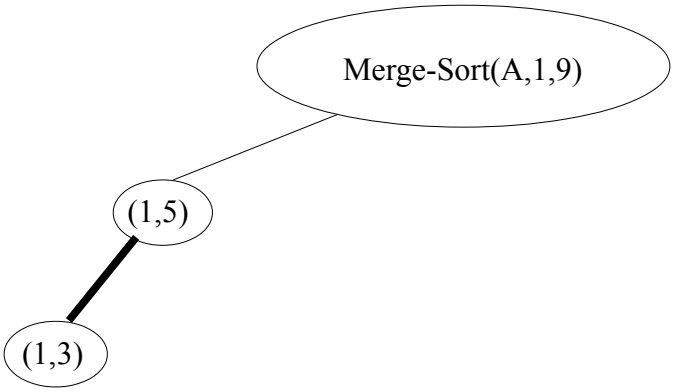
Ο πίνακάς μου περιέχει τα στοιχεία A={1,3,6,16,7,21,35,12,20}. Καλούμε λοιπόν την Merge-Sort (A,l,r) όπου l = 1 και r = 9, δείκτες στο πρώτο και τελευταίο στοιχείο του πίνακα αντίστοιχα.



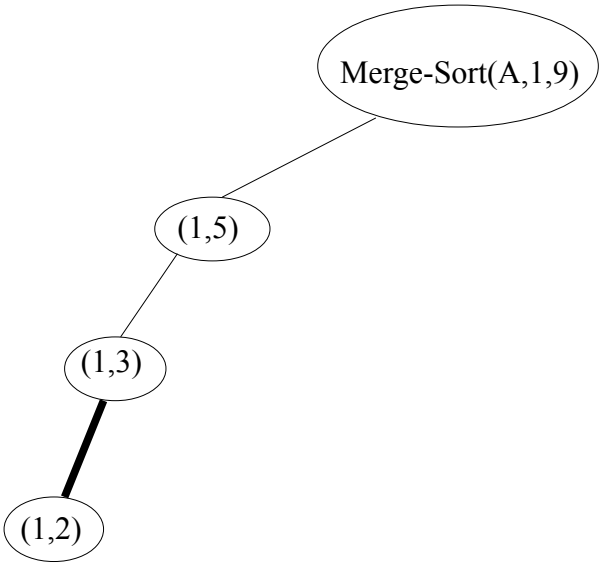
Ο δείκτης l είναι μικρότερος του r. Ορίζουμε δείκτη $q=\lfloor (l+r)/2 \rfloor$ Άρα $q=(1+9)/2=5$ Καλώ την Merge-Sort(A,l,q), δηλαδή τη Merge-Sort(A,1,5). Οπότε έχω:



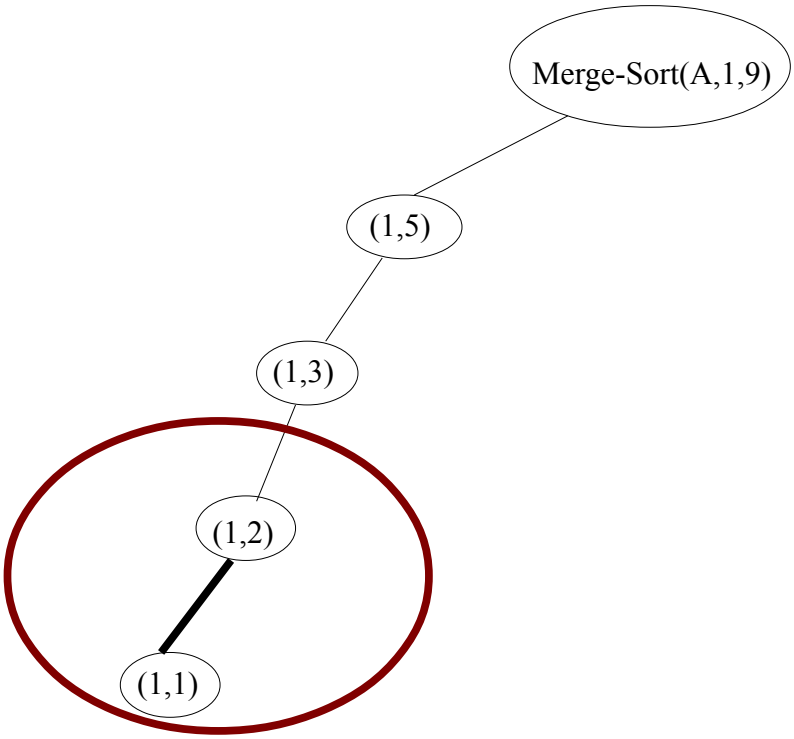
Ελέγχω τους δείκτες. Ο δείκτης 1 είναι μικρότερος του 5. Οπότε $q=\lfloor (1+5)/2 \rfloor = 6/2 = 3$ Καλώ την Merge-Sort(A,1,3). Οπότε έχω:



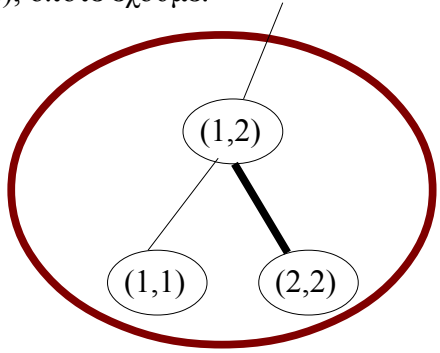
Ελέγχω τους δείκτες. Ο δείκτης 1 είναι μικρότερος του 3. Οπότε $q=\lfloor (1+3)/2 \rfloor = 4/2 = 2$ Καλώ την Merge-Sort(A,1,2). Οπότε έχω:



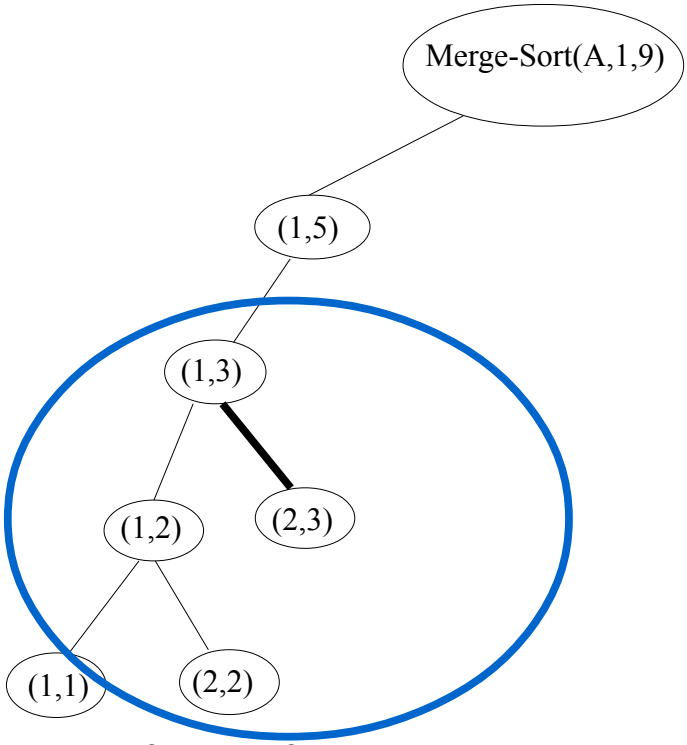
Ελέγχω τους δείκτες. Ο δείκτης 1 είναι μικρότερος του 2. Οπότε $q = \lfloor (1+2)/2 \rfloor = \lfloor 3/2 \rfloor = 1$
Καλώ την Merge-Sort(A,1,1). Οπότε έχω:



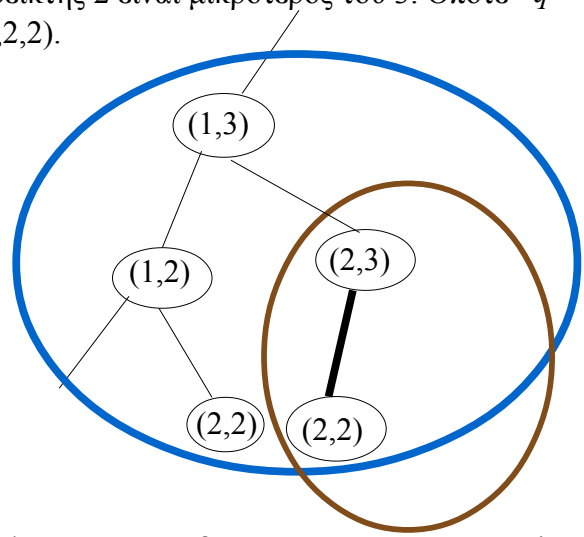
Ο δείκτης 1 δεν είναι μικρότερος του 1, άρα η Merge-Sort(A,1,1) τελείωσε και επιστρέφουμε στο σώμα της Merge-Sort(A,1,2) και όπου q = 1 και r = 2. Τώρα καλούμε την Merge-Sort(A,q+1,r), δηλαδή τη Merge-Sort(A,2,2), οπότε έχουμε:



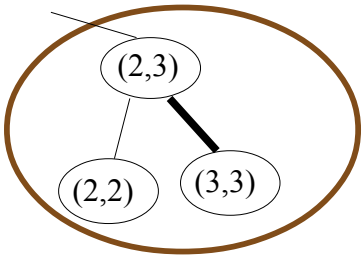
Ο δείκτης 2 δεν είναι μικρότερος του 2, άρα η Merge-Sort(A,2,2) τελείωσε και επιστρέφουμε στο σώμα της Merge-Sort(A,1,2) και κάνουμε Merge τους 2 υποπίνακες. Από τον πίνακα A = {**1**, **3**, 6, 16, 7, 21, 35, 12, 20}, οι δύο υποπίνακες είναι ουσιαστικά οι δείκτες στο 1ο και στο 2ο στοιχείο του πίνακα, δηλαδή το 1 και το 3. Η Merge τα ταξινομεί (αν και είναι ήδη ταξινομημένα) βάζοντας πρώτα το 1 και μετά το 3. Τελείωσε η Merge-Sort(A,1,2) και συνεχίζουμε την εκτέλεση της Merge-Sort(A,1,3) με τη κλήση της συνάρτησης Merge-Sort(A,q+1,r) όπου q = 1 και r = 3, δηλαδή καλούμε την Merge-Sort(A,2,3).



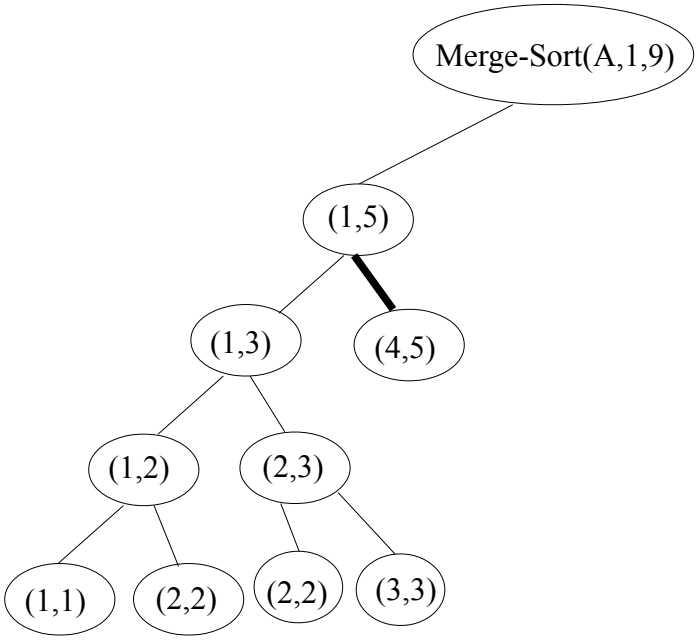
Ελέγχουμε τους δείκτες. Ο δείκτης 2 είναι μικρότερος του 3. Οπότε $q = \lfloor (2+3)/2 \rfloor = \lfloor 5/2 \rfloor = 2$
Καλούμε την Merge-Sort(A,2,2).



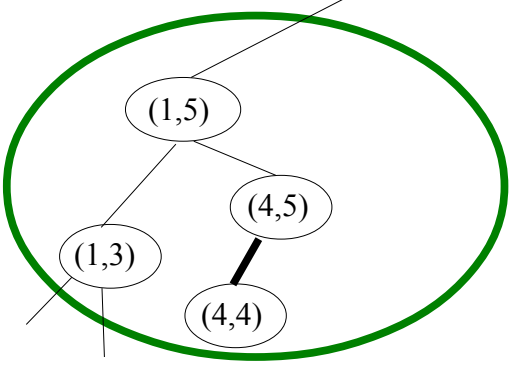
Την Merge-Sort(A,2,2) την εξετάσαμε ακριβώς προηγουμένως. Θα τελειώσει χωρίς να κάνει τίποτε και ο έλεγχος θα επιστρέψει στο σώμα της Merge-Sort(A,2,3) και θα συνεχίσει με τη κλήση της Merge-Sort(A,q+1,r) όπου q = 2 και r = 3. Δηλαδή θα κληθεί η Merge-Sort(A,3,3).



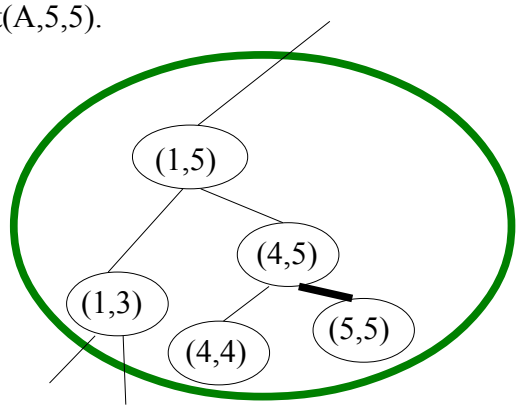
Ο δείκτης 3 δεν είναι μικρότερος του 3 οπότε τελειώνει η Merge-Sort(A,3,3) και επιστρέφουμε στο σώμα της Merge-Sort(A,2,3) και θα συνεχιστεί με το Merge των δύο υποπινάκων, δηλαδή του δείκτη 2 και του δείκτη 3. A = {**1**, **3**, **6**, 16, 7, 21, 35, 12, 20} Άρα, ο δείκτης 2 δείχνει στον αριθμό 3 και ο δείκτης 3 δείχνει στον αριθμό 6. Ταξινομούνται τα δύο αυτά στοιχεία. Το 3 είναι μικρότερο του 6 οπότε τελείωσε το merging. Επιστρέφουμε στο σώμα της Merge-Sort(A,1,3). Συνεχίζουμε με το Merging των υποπινάκων (1,2) => B={1,3} και (2,3) => Γ={3,6}. Το 1 είναι μικρότερο του 3 και το 3 μικρότερο του 6. Οπότε είναι ήδη ταξινομημένα τα στοιχεία και έχουμε έναν νέο πίνακα, έστω **Δ={1,3,6}**. Τελείωσε η εκτέλεση της Merge-Sort(A,1,3) και επιστρέφουμε στο σώμα της Merge-Sort(A,1,5) και συνεχίζουμε με την εκτέλεση της Merge-Sort(A,q+1,r) όπου q = 3 και r = 5. Δηλαδή καλούμε την Merge-Sort(A,4,5). Έχουμε λοιπόν μέχρι στιγμής:



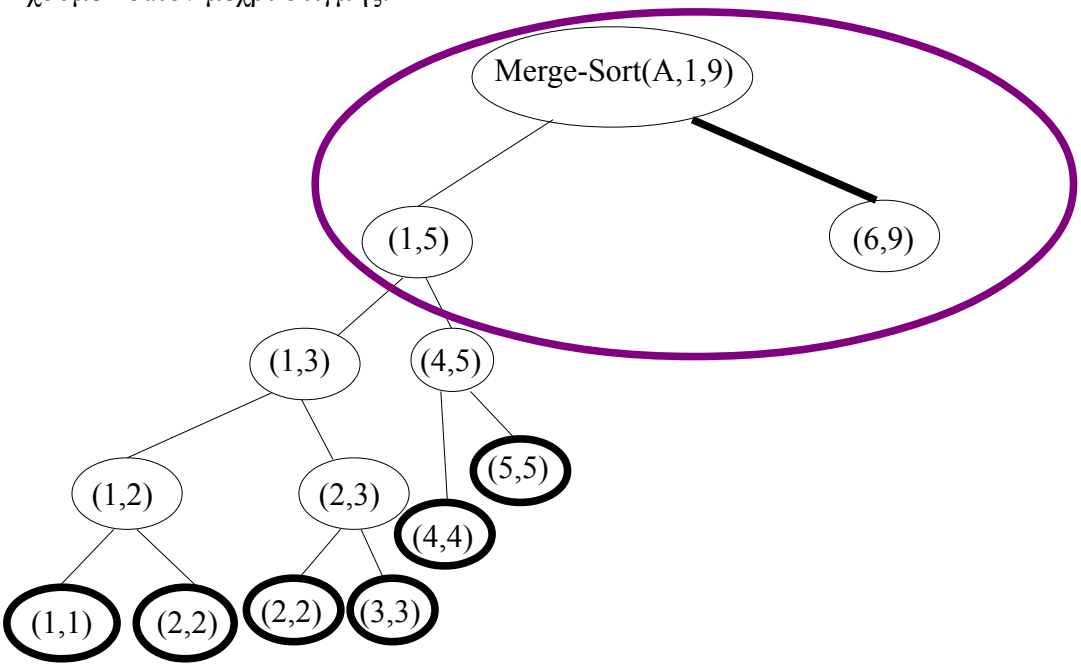
Ελέγχουμε τους δείκτες. Ο δείκτης 4 είναι μικρότερος του 5. Οπότε $q = \lfloor (4+5)/2 \rfloor = \lfloor 9/2 \rfloor = 4$
Καλούμε τη Merge-Sort(A,4,4).



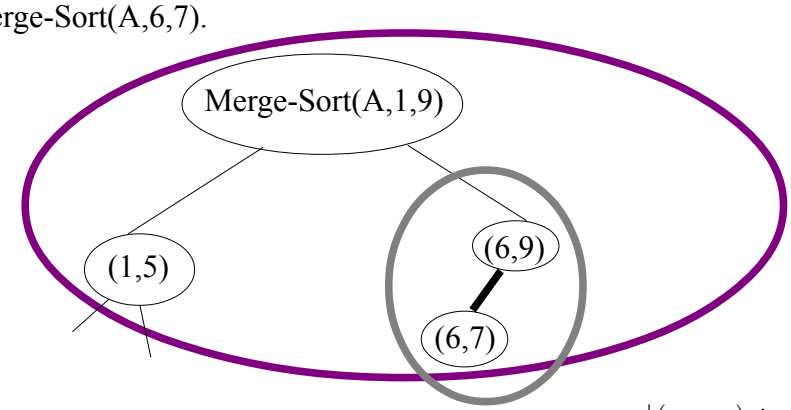
Ελέγχουμε τους δείκτες. Ο δείκτης 4 δεν είναι μικρότερος του δείκτη 4. Άρα επιστρέφουμε στο σώμα της Merge-Sort(A,4,5). Συνεχίζουμε με τη κλήση της Merge-Sort(A,q+1,r) όπου q = 4 και r = 5. Άρα καλούμε την Merge-Sort(A,5,5).



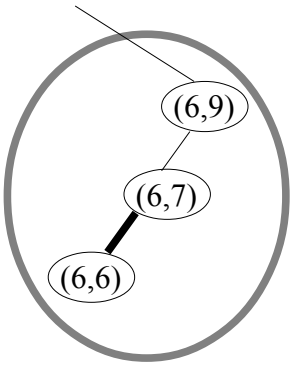
Ελέγχουμε τους δείκτες. Ο δείκτης 5 δεν είναι μικρότερος του 5. Άρα επιστρέφουμε στο σώμα της Merge-Sort(A,4,5) και συνεχίζουμε με το merging των (4,4) & (5,5). A = {1, 3, 6, **16**, **7**, 21, 35, 12, 20}. Ο δείκτης 4 δείχνει στον αριθμό 16, ο 5 στον αριθμό 7. 16>7, άρα αλλάζουμε τη σειρά των αριθμών. Οπότε έχουμε τον πίνακα, έστω **E={7, 16}**. Τέλειωσε το merging και επιστρέφουμε στο σώμα της Merge-Sort(A,1,5). Συνεχίζουμε με το merging των πινάκων **Δ={1, 3, 6}** και **E={7, 16}**. Είναι ταξινομημένα τα στοιχεία. Έτσι δημιουργείται ο πίνακας **Z={1, 3, 6, 7, 16}**. Τελειώσαμε με την Merge-Sort(A,1,5) και επιστρέφουμε στο σώμα της Merge-Sort(A,1,9). Συνεχίζουμε με τη κλήση της Merge-Sort(A,q+1,r) όπου q = 5 και r = 9. Δηλαδή καλούμε την Merge-Sort(A,6,9). Έχουμε λοιπόν μέχρι στιγμής:



Ελέγχουμε τους δείκτες. Ο δείκτης 6 είναι μικρότερος του 9. Άρα $q = \lfloor (6+9)/2 \rfloor = \lfloor 15/2 \rfloor = 7$
Οπότε καλούμε τη Merge-Sort(A,6,7).

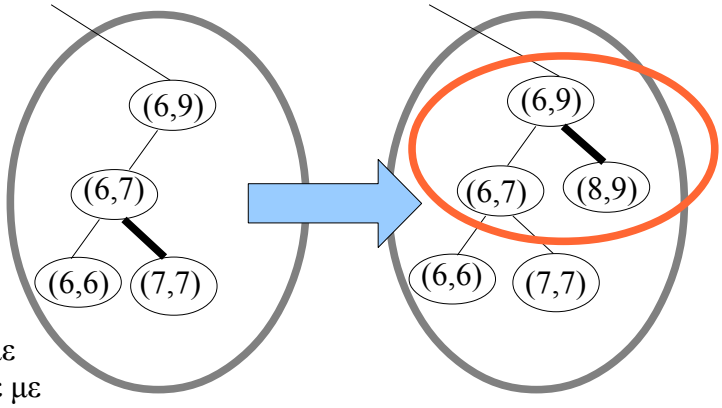


Ελέγχουμε τους δείκτες. Ο δείκτης 6 είναι μικρότερος του 7. Άρα $q = \lfloor (6+7)/2 \rfloor = \lfloor 13/2 \rfloor = 6$
Οπότε καλούμε τη Merge-Sort(A,6,6).



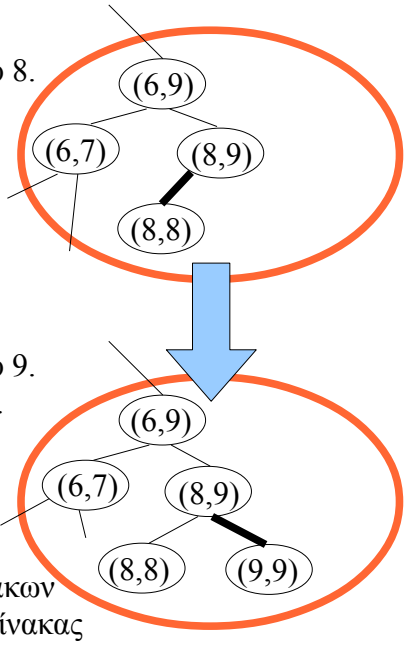
Ελέγχουμε τους δείκτες. Ο δείκτης 6 δεν είναι μικρότερος του 6. Επιστρέφουμε λοιπόν στο σώμα

της Merge-Sort(A,6,7) και συνεχίζουμε την εκτέλεση της με τη κλήση της Merge-Sort(A,q+1,r), όπου q = 6 και r = 7. Δηλαδή καλούμε τη Merge-Sort(A,7,7).



Ελέγχουμε τους δείκτες. Ο δείκτης 7 δεν είναι μικρότερος του 7. Οπότε επιστρέφουμε στο σώμα της Merge-Sort(A,6,7). Συνεχίζουμε με το merging των υποπινάκων (6,6) και (7,7). A={1,3,6,16,7,**21**,**35**,12,20} Ο δείκτης 6 δείχνει στον αριθμό 21. Ο δείκτης 7 δείχνει στον αριθμό 35. 21<35 οπότε είμαστε ΟΚ. Δημιουργείται ένας υποπίνακας **Κ={21,35}**. Τελειώσαμε με το merging οπότε επιστρέφουμε στο σώμα της Merge-Sort(A,6,9). Συνεχίζουμε με τη κλήση της Merge-Sort(A,q+1,r) όπου q = 7 και r = 9. Δηλαδή καλούμε τη Merge-Sort(A,8,9).

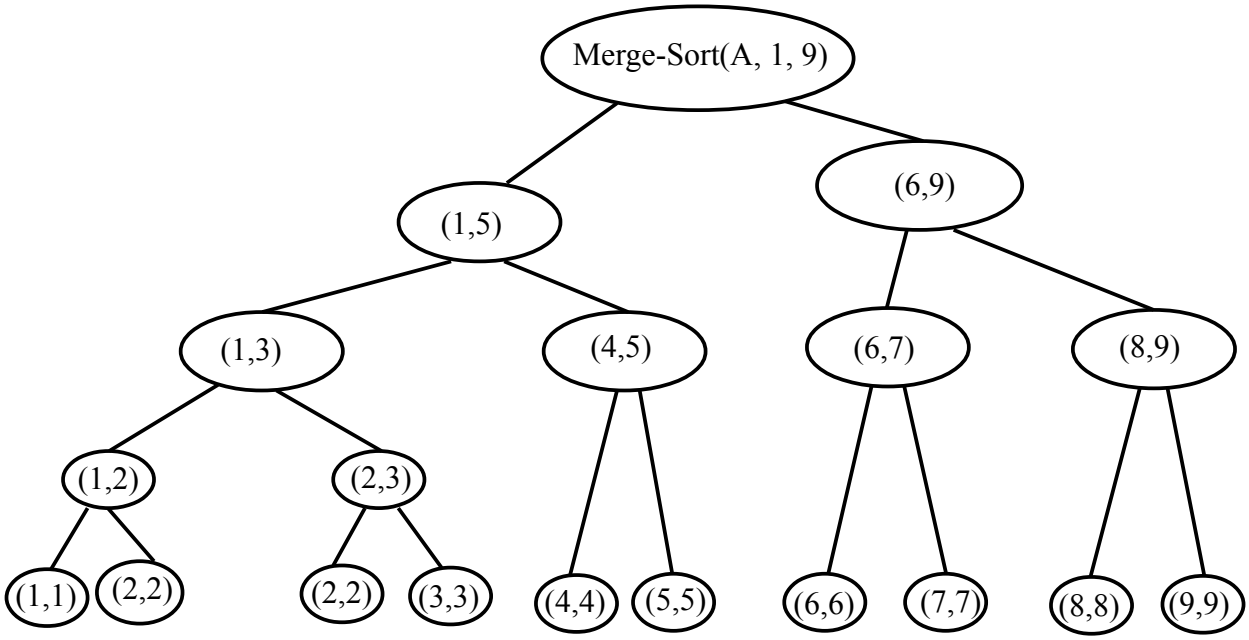
Ελέγχουμε τους δείκτες. Ο δείκτης 8 είναι μικρότερος του 9. Άρα $q = \lfloor (8+9)/2 \rfloor = \lfloor 17/2 \rfloor = 8$ Οπότε καλούμε τη Merge-Sort(A,8,8).



Ελέγχουμε τους δείκτες. Ο δείκτης 8 δεν είναι μικρότερος του 8. Οπότε επιστρέφουμε στο σώμα της Merge-Sort(A,8,9). Στη συνέχεια, καλούμε την Merge-Sort(A,q+1,r). Όπως βρήκαμε προηγουμένως, q = 8 και επίσης ξέρουμε ότι r = 9. Άρα θα καλέσουμε την Merge-Sort(A,9,9).

Ελέγχουμε τους δείκτες. Ο δείκτης 9 δεν είναι μικρότερος του 9. Άρα επιστρέφουμε και πάλι στο σώμα της Merge-Sort(A,8,9). Σειρά τώρα έχει το merging. A={1,3,6,16,7,21,35,**12**,**20**}. Ο δείκτης 8 δείχνει στον αριθμό 12. Ο δείκτης 9 δείχνει στον αριθμό 20. 12<20 οπότε είμαστε ΟΚ. Έτσι δημιουργείται ένας υποπίνακας **Λ={12,20}**. Επιστρέφουμε τώρα στο σώμα της Merge-Sort(A,6,9) και σειρά έχει το merging των υποπινάκων **Κ={21,35}** και **Λ={12,20}**. Από το merging προκύπτει ο υποπίνακας **Μ={12, 20, 21, 35}**. Τελειώσαμε με την Merge-Sort(A,6,9). Επιστρέφουμε στην Merge-Sort(A,1,9) και σειρά έχει το τελευταίο βήμα, δηλαδή το merging των δημιουργηθέντων υποπινάκων **Ζ** και **Μ**. Έτσι προκύπτει ο τελικός, ταξινομημένος πίνακας A={1, 3, 6, 7, 12, 16, 20, 21, 35} και σχηματίζεται το τελικό δέντρο αναδρομικών κλήσεων του αλγορίθμου MergeSort για τον πίνακα Α.

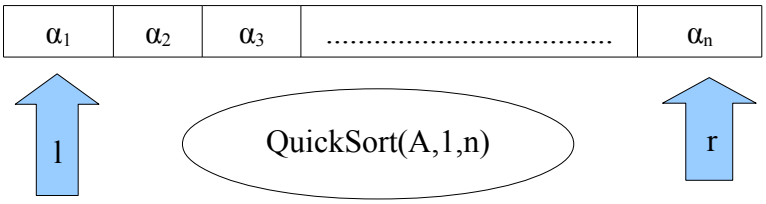
Το σχήμα βρίσκεται στην επόμενη σελίδα



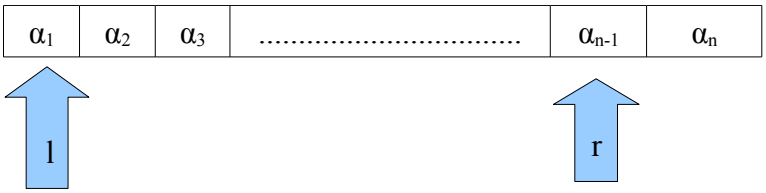
5.

a)

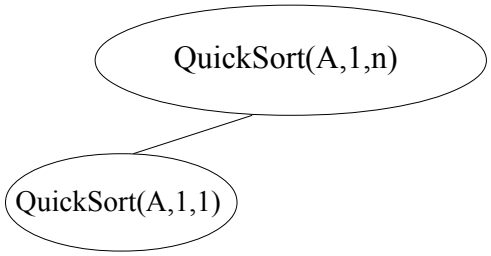
Έστω ο πίνακας $A = \{\alpha_1, \alpha_2, \alpha_3 ..... \alpha_n\}$ ταξινομημένος κατά αύξουσα σειρά. Θα εφαρμόσουμε την QuickSort(A,l,r) όπου l είναι δείκτης στο 1ο στοιχείο του πίνακα και r δείκτης στο τελευταίο στοιχείο.



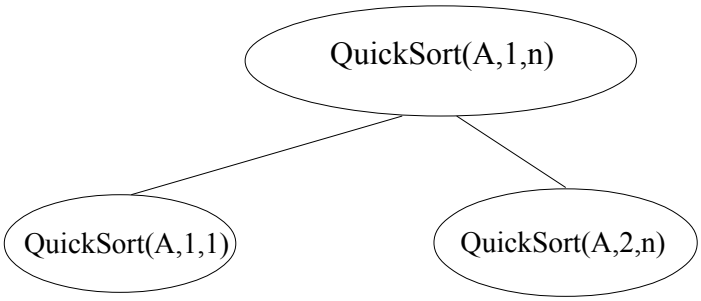
Ελέγχουμε τους δείκτες. Ο l είναι μικρότερος του n. Συνεχίζουμε με το partitioning. Ας επιλέξουμε το πρώτο στοιχείο σα pivot. Θα συγκρίνουμε το pivot με το στοιχείο στο οποί δείχνει ο r, δηλαδή το α_n. Ο πίνακας είναι ήδη ταξινομημένος κατά αύξουσα σειρά. Οπότε pivot < α_n. Άρα ο δείκτης r μετακινείται μια θέση αριστερά στο προτελευταίο στοιχείο, δηλαδή το α_{n-1}.



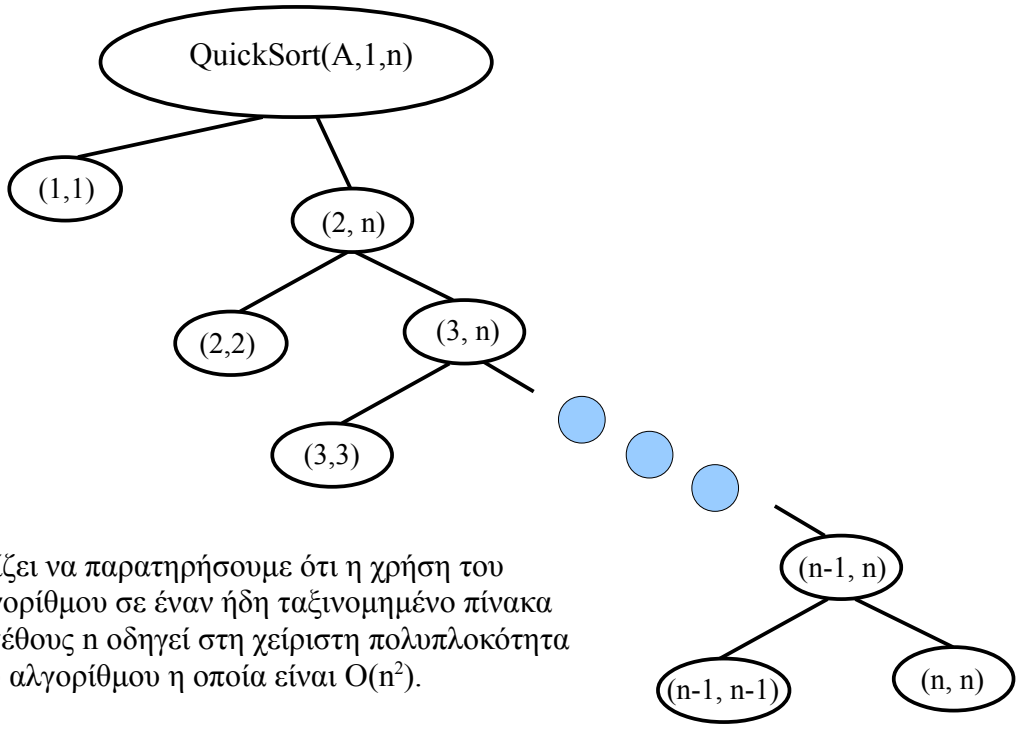
Συγκρίνουμε το pivot με το αn-1. Ο πίνακας είναι ήδη ταξινομημένος, άρα pivot < α_{n-1}. Παρατηρούμε ότι η διαδικασία επαναλαμβάνεται μέχρι ο δείκτης r να ταυτιστεί με τον l. Το pivot στοιχείο μας ήταν αρχικά στη πρώτη θέση και παρέμεινε εκεί. Οπότε τώρα θα κληθεί η QuickSort(A,1,1).



Ελέγχουμε τους δείκτες. Ο δείκτης 1 δεν είναι μικρότερος του 1. Οπότε επιστρέφουμε στο σώμα της QuickSort(A,1,n) και συνεχίζουμε με τη κλήση της QuickSort(A,2,n).



Παρατηρούμε ότι αρχικά ανάγει το πρόβλημα σε πρόβλημα τάξης n-1 αφαιρώντας κάθε φορά ένα στοιχείο το οποίο θεωρείται ταξινομημένο. Οπότε το δέντρο των αναδρομικών κλήσεων του αλγορίθμου θα είναι το εξής:



Αξίζει να παρατηρήσουμε ότι η χρήση του αλγορίθμου σε έναν ήδη ταξινομημένο πίνακα μεγέθους n οδηγεί στη χειριστη πολυπλοκότητα του αλγορίθμου η οποία είναι $O(n^2)$.

b)

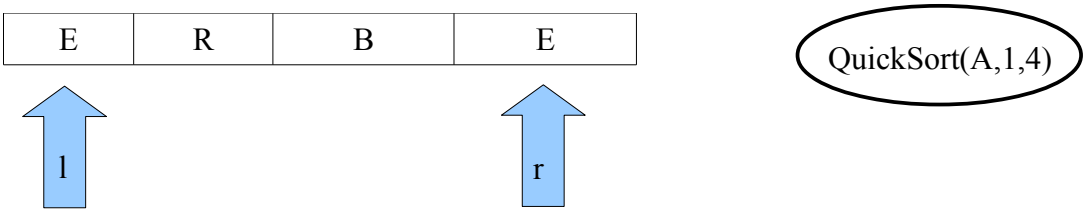
Όπως είπα και παραπάνω, η πολυπλοκότητα του αλγορίθμου για τη χείριστη περίπτωση είναι $O(n^2)$. Αυτό προκύπτει εύκολα αφού τα partitions δίνουν πίνακες μεγέθους n, n-1, n-2 …..... 2. Αθροίζοντας τις ποσότητες προκύπτει $n + (n-1) + (n-2) + ….. + 2 = (n+2)(n-1)/2 = O(n^2)$. Στη χείριστη περίπτωση, η πολυπλοκότητα της HeapSort είναι $nlogn = \Omega(n^2)$. Δηλαδή η χείριστη πολυπλοκότητα της HeapSort είναι καλύτερη από εκείνη της QuickSort. Όμως σπάνια εμφανίζεται η χείριστη πολυπλοκότητα της QuickSort. Τις περισσότερες φορές, είναι η καλύτερη επιλογή καθώς ταξινομεί πολύ γρήγορα τα στοιχεία ενός πίνακα.

c)

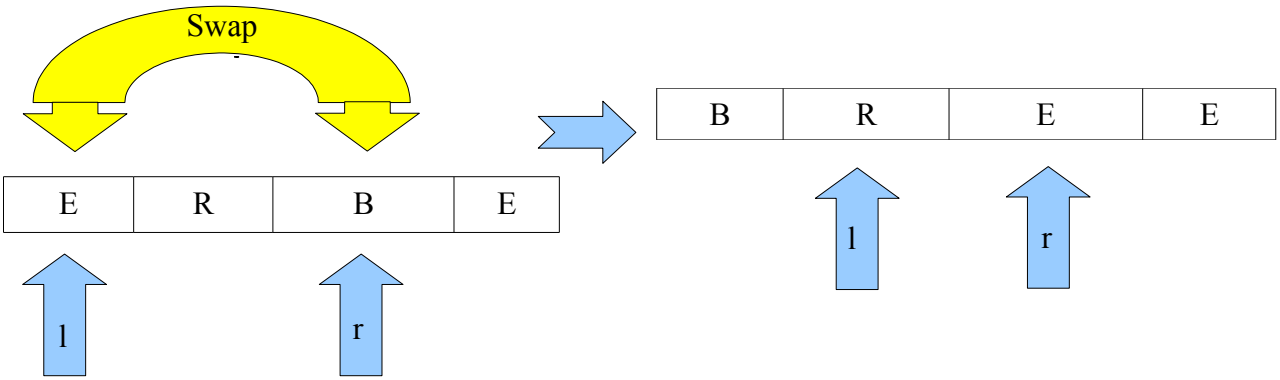
Όσον αφορά τη χείριστη περίπτωση, ο αλγόριθμος της QuickSort **ΔΕΝ** είναι ο βέλτιστος καθώς όπως είδαμε στο προηγούμενο ερώτημα, όσον αφορά τη χείριστη περίπτωση, η πολυπλοκότητα της HeapSort είναι καλύτερη. Άρα, όσον αφορά τη χείριστη περίπτωση, ο αλγόριθμος της QuickSort δεν είναι ο βέλτιστος.

d)

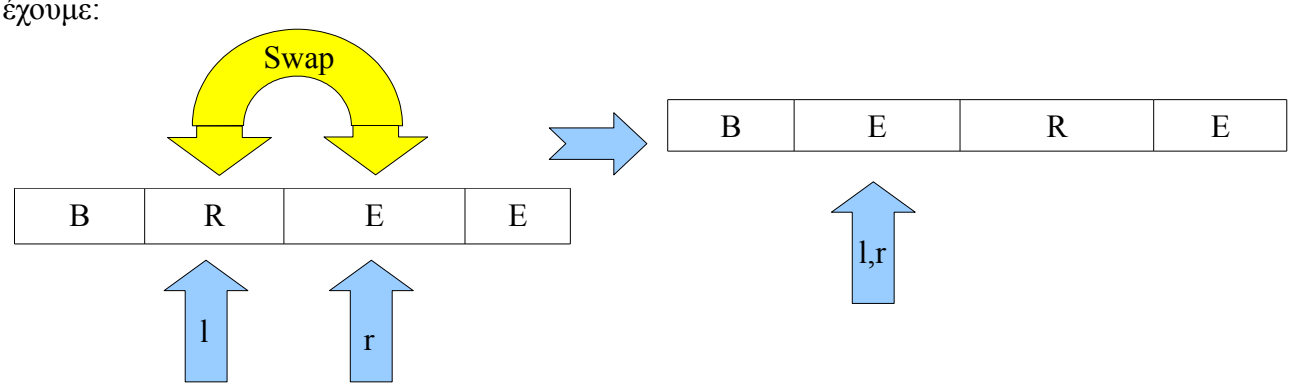
Έχουμε 4 στοιχεία. Έστω ότι έχουμε τον πίνακα A={E, R, B, E}. Άρα θα καλέσουμε αρχικά την QuickSort(A,1,4).



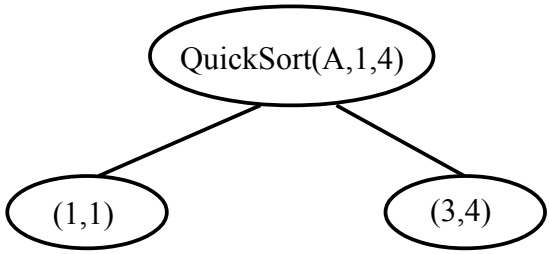
Ελέγχουμε τους δείκτες l και r. Παρατηρούμε ότι l < r. Συνεχίζουμε με το partitioning. Διαλέγουμε το πρώτο στοιχείο σα pivot. Οι δείκτες δείχνουν σε ίδιες τιμές, άρα δε χρειάζεται να ανταλλάξουμε τα στοιχεία. Μετακινούμε λοιπόν τον δείκτη r μια θέση αριστερά. Συγκρίνουμε τις τιμές των δεικτών. Παρατηρούμε ότι η τιμή του pivot είναι μεγαλύτερη από το στοιχείο του δείκτη r. Οπότε τα ανταλλάζουμε. Ανταλλάξαμε στοιχεία οπότε τώρα, αντί να μετακινήσουμε τον δείκτη r, θα μετακινήσουμε τον δείκτη l κατά μία θέση προς τα δεξιά αυτή τη φορά. Άρα θα έχουμε:



Ελέγχουμε τις τιμές των δεικτών l και r. Παρατηρούμε ότι το στοιχείο στο οποίο δείχνει ο δείκτης l είναι μεγαλύτερο από εκείνο του δείκτη r. Επομένως τα ανταλλάζουμε. Επειδή όμως ανταλλάξαμε στοιχεία, τώρα θα μετακινηθεί ο δείκτης r μια θέση αριστερά. Οι δείκτες πλέον ταυτίζονται. Άρα έχουμε:

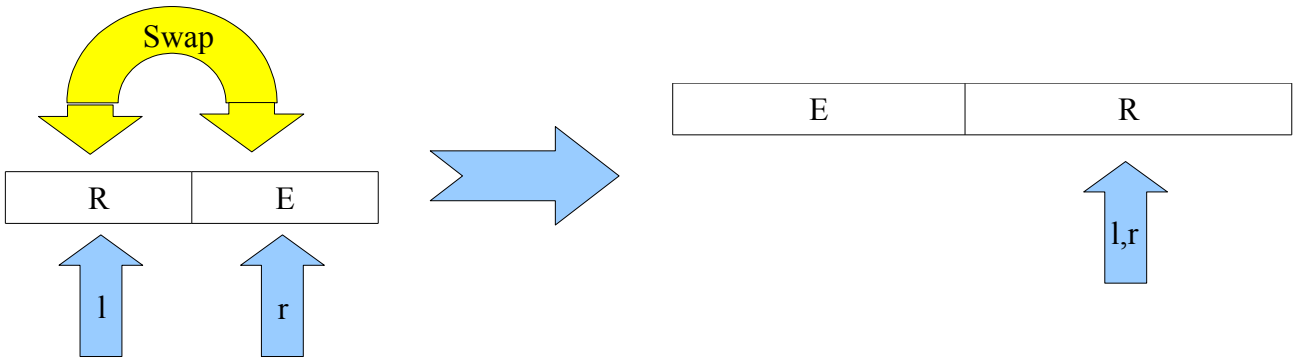


Τελειώσαμε με το partitioning. Ακολουθεί η κλήση των δύο υποπινάκων. Δηλαδή του {B} και του K = {R, E}. Καλούμε λοιπόν την QuickSort(A,1,1). Ελέγχουμε τους δείκτες όμως εκείνοι ταυτίζονται. Άρα επιστρέφουμε πάλι στο σώμα της QuickSort(A,1,4) και συνεχίζουμε με την εκτέλεση της QuickSort(A,3,4). Έχουμε λοιπόν:

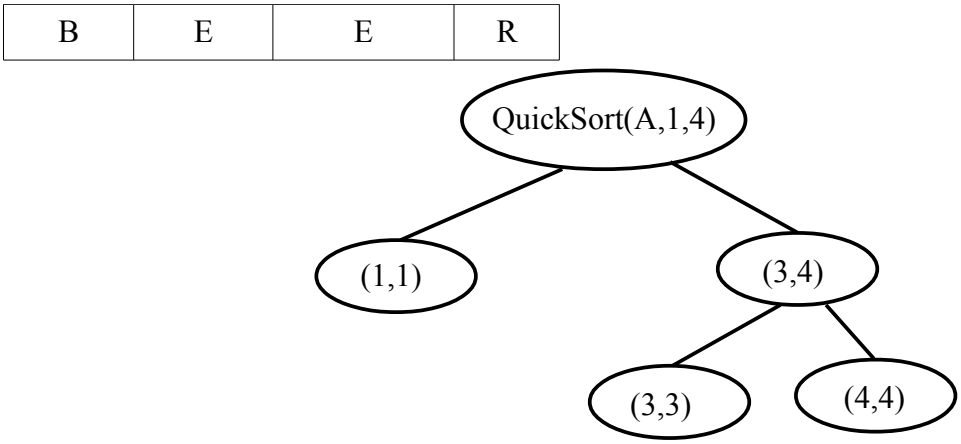


Τώρα θα δουλέψουμε με τον υποπίνακα K = {R, E}. Ελέγχουμε τους δείκτες. Ο 3 είναι μικρότερος του 4. Άρα συνεχίζουμε με το partitioning. Επιλέγουμε ως pivot, το πρώτο στοιχείο. Βλέπουμε ότι

το στοιχείο στο οποίο δείχνει ο δείκτης r είναι μικρότερος του pivot. Οπότε πρέπει να τα ανταλλάξουμε. Επειδή όμως ανταλλάξαμε στοιχεία θα μετακινηθεί ο δείκτης l μια θέση προς τα δεξιά. Έχουμε λοιπόν:



Οι δείκτες ταυτίστηκαν, οπότε τελείωσε το partitioning. Προχωρούμε με τη κλήση της QuickSort για τους 2 υποπίνακες που δημιουργήθηκαν, δηλαδή της QuickSort(A,3,3) και της QuickSort(A,4,4). Αυτοί όμως οι υποπίνακες θεωρούνται ταξινομημένοι καθώς αποτελούνται από ένα στοιχείο ο καθένας. Οι δύο πρώτες θέσεις ταξινομήθηκαν προηγουμένως, ταξινομήθηκαν μόλις και οι δύο τελευταίες οπότε μπορούμε να δούμε τον πίνακα ταξινομημένο καθώς και το δέντρο των αναδρομικών κλήσεων.



6.

a)

Έστω h το ύψος του σωρού και n το πλήθος των κόμβων του. Θα αποδείξω τη σχέση επαγωγικά.

Επαγωγική βάση: Θα δείξω ότι ισχύει η σχέση για h=0. Έστω χ είναι ο αριθμός των κόμβων στο τελευταίο επίπεδο ενός πλήρους δυαδικού δέντρου. Άρα $n=2^{h+1}-1$. Παρατηρήστε ότι το n-χ είναι περιττός επειδή το πλήρες δυαδικό δέντρο έχει περιττό πλήθος κόμβων (μια δύναμη του 2 μείον 1). Άν το n είναι περιττός τότε το χ είναι άρτιος, άρα όλοι οι εσωτερικοί κόμβοι έχουν παιδιά. Σύμφωνα με θεώρημα, # εσωτερικών κόμβων = # φύλλων – 1. Άρα n = # φύλλων + # εσωτερικών κόμβων = 2(# φύλλων) – 1 => # φύλλων = (n+1)/2 = $\lceil n/2^{0+1} \rceil$ επειδή n περιττός. Άρα ισχύει η επαγωγική βάση.

Επαγωγικό βήμα: Θα δείξω ότι αν η σχέση ισχύει για ύψος h-1, θα ισχύει και για h. Έστω n_h είναι ο αριθμός των κόμβων στο ύψος h ενός δέντρου T, n κόμβων. Έστω τώρα ένα δέντρο T' το οποίο προκύπτει όταν από το T αφαιρέσουμε όλα τα φύλλα. Θα έχει n' = n – n_0 κόμβους. Όμως από την επαγωγική βάση γνωρίζουμε ότι n_0 = $\lceil n/2 \rceil$, άρα n' = n - $\lceil n/2 \rceil$ = $\lfloor n/2 \rfloor$ Παρατηρούμε ότι οι κόμβοι που βρίσκονταν στο ύψος h του δέντρου T ανέβηκαν ένα επίπεδο και βρίσκονται στο ύψος h-1 του δέντρου T'. Ορίζοντας με n'_{h-1} τον αριθμό των κόμβων στο ύψος h-1 του δέντρου T', έχουμε

$n_h = n'_{h-1}$. Έχουμε λοιπόν: $n_h = n'_{h-1} \le \lceil n'/2^h \rceil = \lceil \lfloor n/2 \rfloor / 2^h \rceil \le \lceil (n/2)/2^h \rceil = \lceil n/2^{h+1} \rceil$ Απεδείχθη.

b)

Ο σωρός ύψους h με μέγιστο πλήθος στοιχείων μπορεί να θεωρηθεί ως ένα πλήρες δυαδικό δέντρο. Έτσι λοιπόν, στο επίπεδο h υπάρχουν 2^{h-1} στοιχεία. Άρα το μέγιστο πλήθος στοιχείων ενός σωρού ύψους h είναι το άθροισμα του πλήθους των στοιχείων που υπάρχουν στο κάθε επίπεδο. Δηλαδή μέγιστο πλήθος στοιχείων στο σωρό είναι $\sum_{i=0}^{h-1} 2^i = 2^h - 1$. Ένας σωρός ύψους h με ελάχιστο πλήθος στοιχείων σημαίνει πως το προηγούμενο επίπεδο, δηλαδή το h – 1 έχει μέγιστο πλήθος στοιχείων. Το τελευταίο επίπεδο θα περιέχει ένα στοιχείο, το ελάχιστο δηλαδή που μπορεί να έχει. Άρα έχουμε ότι το ελάχιστο πλήθος στοιχείων σε έναν σωρό ύψους h είναι:

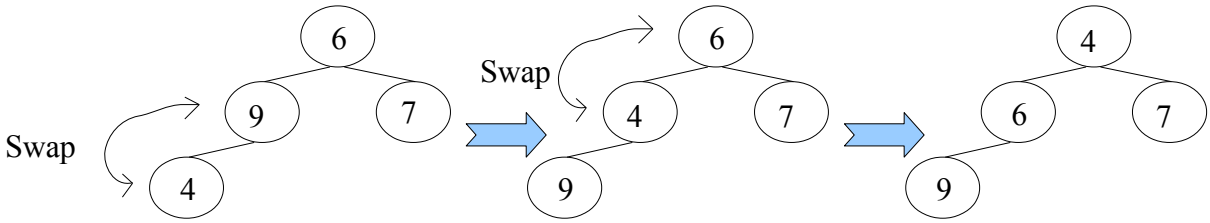
$$\underbrace{2^{h-1}-1}_{\text{πλήθος στοιχείων προτελευταίου επιπέδου}} + \underbrace{1}_{\text{πλήθος στοιχείων τελευταίου επιπέδου}} = 2^{h-1}$$

c)

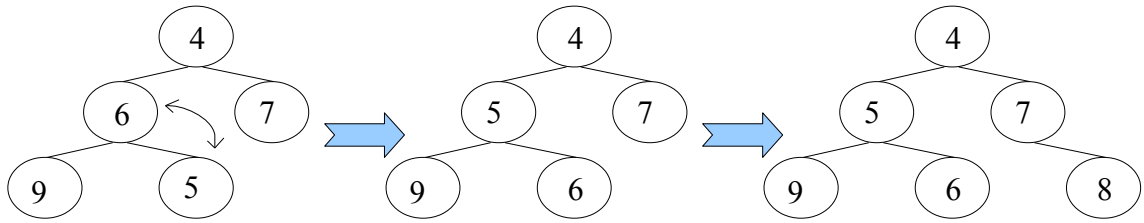
Θεωρούμε ότι οι θέσεις του πίνακα αντιστοιχούν σε μία κατά πλάτος επίσκεψη των κόμβων του σωρού. Όταν ένας πίνακας είναι ταξινομημένος κατά φθίνουσα σειρά, τότε το πρώτο του στοιχείο, δηλαδή αυτό που αντιστοιχεί στη ρίζα είναι το μέγιστο. Επίσης τα παιδιά κάθε κόμβου είναι έχουν πάντα μικρότερη τιμή από τον πατέρα. Το δέντρο που δημιουργείται μετά από την εισαγωγή όλων των στοιχείων είναι σχεδόν πλήρες. Ικανοποιούνται όλες οι προϋποθέσεις του σωρού, οπότε ένας πίνακας σε φθίνουσα διάταξη είναι σωρός.

d)

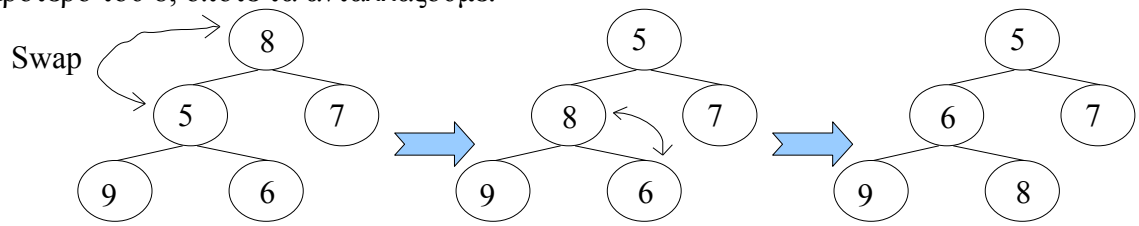
Θα ταξινομήσουμε τα στοιχεία 6, 9, 7, 4, 5, 8. Αρχικά θα δημιουργήσουμε το σωρό. Εισάγουμε το πρώτο στοιχείο. Εισάγουμε και το δεύτερο. Το 6 είναι μικρότερο του 9, οπότε συνεχίζουμε με την εισαγωγή του τρίτου στοιχείου. Το 6 είναι μικρότερο του 7 οπότε συνεχίζουμε με την εισαγωγή του επόμενου στοιχείου. Ο πατέρας του όμως είναι μεγαλύτερος, επομένως πρέπει να τα ανταλλάξουμε. Ο πατέρας όμως του 4 είναι μεγαλύτερος, άρα πρέπει να τους ανταλλάξουμε και αυτούς.



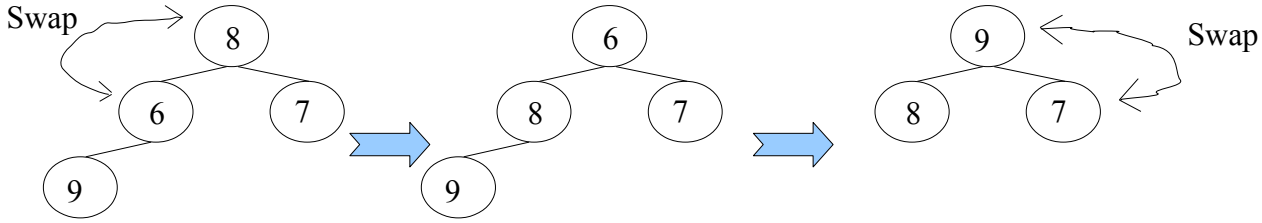
Αφού όλα είναι εντάξει προχωράμε με την εισαγωγή του επόμενου στοιχείου. Το 5 όμως είναι μικρότερο από το πατέρα. Άρα πρέπει να τα ανταλλάξουμε. Είμαστε εντάξει, οπότε συνεχίζουμε με την εισαγωγή του νέου στοιχείου. Το 8 είναι μεγαλύτερο από τον πατέρα του οπότε είμαστε εντάξει.



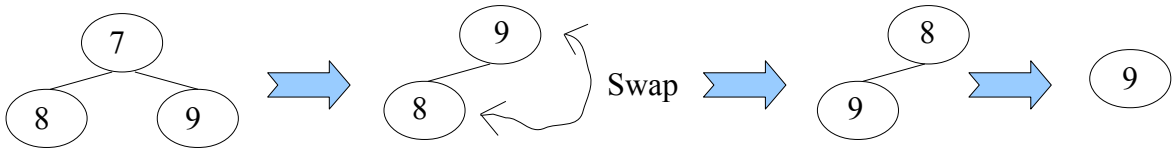
Η ρίζα είναι ταξινομημένη. Οπότε την αφαιρούμε και στη θέση της βάζουμε το τελευταίο στοιχείο του σωρού, δηλαδή το 8. Βλέπουμε ότι η νέα ρίζα είναι μεγαλύτερη από τουλάχιστον ένα παιδί, οπότε την ανταλλάζουμε με το μικρότερο παιδί. Το 8 πλέον είναι πατέρας του 6. Όμως το 6 είναι μικρότερο του 8, οπότε τα ανταλλάζουμε.



Η ρίζα είναι ταξινομημένη οπότε αφαιρείται. Έτσι έχουμε μέχρι στιγμής το 4 και το 5. Το τελευταίο στοιχείο (8) πάει στη θέση της ρίζας. Όμως είναι μεγαλύτερη από τουλάχιστον ένα παιδί της οπότε και την ανταλλάζουμε με το μικρότερο παιδί του. Όλα είναι εντάξει και η ρίζα είναι ταξινομημένη. Οπότε τα στοιχεία 4,5,6 είναι ταξινομημένα. Οπότε αφαιρείται η ρίζα και τη θέση της παίρνει το τελευταίο στοιχείο δηλαδή το 9.



Η νέα ρίζα όμως έχει τουλάχιστον ένα παιδί με μικρότερη τιμή. Οπότε ανταλλάζουμε το στοιχείο της ρίζας με εκείνο του μικρότερου παιδιού. Όλα ΟΚ, επομένως η ρίζα αφαιρείται και το στοιχείο είναι ταξινομημένο. Οπότε μέχρι τώρα είναι ταξινομημένα τα στοιχεία 4,5,6,7. Τη θέση της ρίζας παίρνει το τελευταίο στοιχείο, δηλαδή το 9. Το 9 είναι όμως μεγαλύτερο από το παιδί του, το 8. Επομένως τα ανταλλάζουμε. Είμαστε ΟΚ, οπότε αφαιρείται η ρίζα. Έμεινε ένα μόνο στοιχείο, το 9. Το τοποθετούμε στον πίνακα και πλέον έχουμε ταξινομήσει τα στοιχεία.



<u>Συνέχεια της 1 (Αλγόριθμος 2)</u>

Είχαμε μείνει σε αυτή τη σχέση: $T(n) = a \cdot 2^{\log n} + n\left(\frac{1}{2}\log^2 n + \frac{1}{2}\right) = a \cdot n + \frac{1}{2} n \log^2 n + \frac{1}{4}$

Αντιλαμβανόμαστε ότι η ποσότητα $n\log^2 n$ υπερισχύει της n. Άρα η πολυπλοκότητα είναι $O\left(n\log^2 n\right)$