ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΤΜΗΜΑ ΙΣΤΟΡΙΑΣ-ΑΡΧΑΙΟΛΟΓΙΑΣ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

ΔΙΔΑΚΤΙΚΗ ΕΝΟΤΗΤΑ 9

ΘΕΜΑ: ΒΑΣΙΚΕΣ ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΑΝΑΛΥΣΗΣ ΑΛΓΟΡΙΘΜΩΝ-ΕΞΙΣΟΡΡΟΠΗΣΗ, ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ

Δρ. Δημήτριος Κ. Κουκόπουλος
Αναπληρωτής Καθηγητής

ΠΕΡΙΕΧΟΜΕΝΟ

- Εξισορρόπηση
- Διαίρει και Βασίλευε
- Εφαρμογές της τεχνικής του Διαίρει και Βασίλευε
  - Δυαδική Αναζήτηση
  - Γρήγορη Ταξινόμηση - QuickSort

ΓΕΝΙΚΗ ΑΡΧΗ ΤΕΧΝΙΚΩΝ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΑΝΑΛΥΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

Καθορισμός ενδιάμεσων στόχων που μπορούν να επιτευχθούν πιο εύκολα και μεσολαβούν για την επίτευξη του τελικού στόχου.

ΤΕΧΝΙΚΗ ΤΗΣ ΕΞΙΣΟΡΡΟΠΗΣΗΣ

- **Γενικός Κανόνας:** Πρέπει να διατηρείται μια ισορροπία ή και ισότητα μεταξύ των διαφόρων μεγεθών που προκύπτουν από τη διάσπαση ενός προβλήματος σε υποπροβλήματα.

- **Επιχειρήματα υπέρ της εξισορρόπησης**
  - Πρέπει να υπάρχει ίση μεταχείριση.
  - Ο χρόνος εκτέλεσης ενός έργου είναι αύξουσα συνάρτηση κάποιου μεγέθους που αντιστοιχεί στο έργο. Αν έχουμε να εκτελέσουμε δυο έργα είναι προτιμότερο να έχουν το ίδιο μέγεθος είτε τα εκτελέσουμε παράλληλα είτε διαδοχικά.

ΕΞΙΣΟΡΡΟΠΗΣΗ-ΘΕΩΡΗΜΑ

Θεώρημα:

Έστω **T(n)** ο χρόνος επίλυσης ενός προβλήματος μεγέθους **n**, όπου οι συναρτήσεις **T** και **T'** είναι αυστηρά αύξουσες. Έστω $\mathbf{n_1}$, $\mathbf{n_2}$ αντίστοιχα τα μεγέθη δυο υποπροβλημάτων με $\mathbf{n_1+n_2=n}$. Τότε ο ελάχιστος χρόνος για τη λύση και των δυο υποπροβλημάτων σειριακά δίνεται, όταν $\mathbf{n_1=n_2=n/2}$.

ΑΠΟΔΕΙΞΗ

Συνολικός χρόνος: $\mathbf{T(n_1)+T(n_2)=T(n_1)+T(n-n_1)=f(n_1)}$.

Για να υπάρχει ελάχιστο πρέπει $\mathbf{f'(n_1)=0}$. Δηλαδή, $\mathbf{T'(n_1)-T'(n-n_1)=0}$. Οπότε, $\mathbf{T'(n_1)=T'(n-n_1)}$.

Επειδή, η **T’** είναι αυστηρά αύξουσα συνάρτηση η παραπάνω ισότητα ισχύει όταν $\mathbf{n_1=n-n_1}$, δηλαδή $\mathbf{n_1=n_2=n/2}$.

Ακόμη ισχύει $\mathbf{f''(n_1)=T''(n_1)+T''(n-n_1)>0}$. Άρα, για $\mathbf{n_1=n_2=n/2}$ η συνάρτηση $\mathbf{f(n_1)}$ εμφανίζει ελάχιστο.

ΕΞΙΣΟΡΡΟΠΗΣΗ-ΠΑΡΑΔΕΙΓΜΑ

- Έστω το πρόβλημα της αναζήτησης σε λίστα **n** στοιχείων.
- Κατά τη διαδοχική αναζήτηση η σύγκριση με το 1ο στοιχείο ανάγει την επίλυση του προβλήματος σε μικρότερο στιγμιότυπο που ορίζεται από λίστα μεγέθους n-1 στοιχείων. Επομένως, η διαδοχική αναζήτηση δεν εφαρμόζει την τεχνική της εξισορρόπησης.
- Στη δυαδική αναζήτηση η πρώτη διάσπαση παράγει δυο στιγμιότυπα μισού μεγέθους από το αρχικό. Το ίδιο συμβαίνει και σε κάθε επόμενη διάσπαση. Επομένως, η δυαδική αναζήτηση εφαρμόζει την τεχνική της εξισορρόπησης που την κάνει καλύτερο αλγόριθμο από τη διαδοχική αναζήτηση.

ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (DIVIDE AND CONQUER)

Τυπικός Κανόνας: Αντί να εξετάσουμε ένα πρόβλημα που αφορά ένα μεγάλο μέγεθος εισόδου άμεσα, διασπούμε την είσοδο σε μικρότερα κομμάτια, λύνουμε το πρόβλημα για κάθε μικρό κομμάτι ξεχωριστά και μετά πληρώνοντας ένα επιπλέον κόστος συνδέουμε τα επιμέρους αποτελέσματα για να δώσουμε λύση στο αρχικό μέγεθος εισόδου του προβλήματος.

Εναλλακτικός Κανόνας: Μέθοδοι που ανάγουν αρχικό πρόβλημα σε όλο και μικρότερα προβλήματα με την αναδρομική εφαρμογή τους.

Σύμφωνα με την αρχή της εξισορρόπησης τα κομμάτια πρέπει να είναι ισομεγέθη.

ΑΝΑΛΥΣΗ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ

Υπόθεση

Έστω ότι η είσοδος αντιπροσωπεύεται από την παράμετρο **n** που μπορεί να μεγαλώσει αυθαίρετα. Χωρίς βλάβη της γενικότητας μπορούμε να υποθέσουμε ότι $\mathbf{n=2^p}$, $\mathbf{p=\log n}$.

Περιπτώσεις Εφαρμογής Μεθόδου

- **1η Περίπτωση:** Απαιτείται η λύση ενός μόνο από τα δυο υποπροβλήματα που προκύπτουν από μια διάσπαση (π.χ. δυαδική αναζήτηση).
- **2η Περίπτωση:** Απαιτείται η λύση και των δυο υποπροβλημάτων που προκύπτουν από μια διάσπαση (π.χ. ταξινόμηση QuickSort).

$1^η$ ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (1/4)

- Μετά την **πρώτη** διάσπαση έχουμε 2 υποστιγμιότυπα μεγάθους n/2 το καθένα. Μετά τη **δεύτερη** διάσπαση έχουμε 4 υποστιγμιότυπα μεγάθους n/4 το καθένα. Στη **j-στη διάσπαση** έχουμε $\mathbf{2^j}$ υποστιγμιότυπα μεγέθους $\mathbf{2^{p-j} = n/2^j}$ το καθένα. Η τελευταία διάσπαση είναι η **διάσπαση p**, όπου κάθε στιγμιότυπο έχει μέγεθος 1.

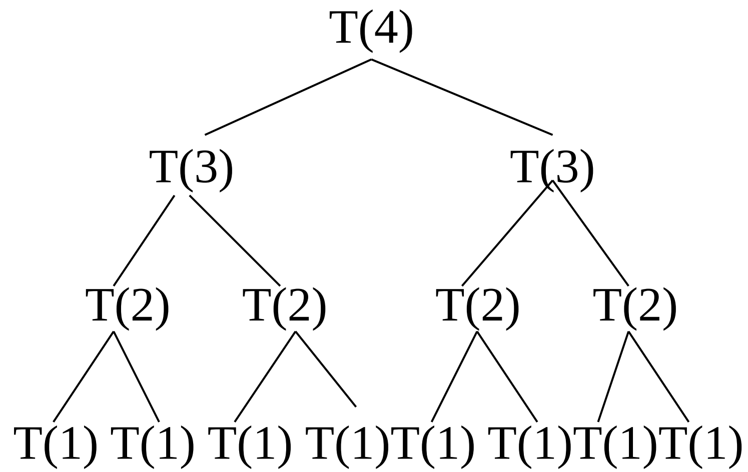
- Η λύση θα ξεκινήσει με μικρότερα κομμάτια και θα συνεχίζεται με σύνθεση δυο κομματιών κάθε φορά που θα δίνει λύση για κάθε μεγαλύτερο κομμάτι προηγούμενης διάσπασης.

- **Υποθέσεις:** Έστω ότι εκτελώ $k \leq p$ διασπάσεις και τις αριθμώ από την τελευταία προς την πρώτη. Όλα τα κομμάτια στην **i-στη διάσπαση** είναι ισομεγέθη και απαιτούν τον ίδιο χρόνο **T(i)**.

$1^η$ ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (2/4)

- Για n=16, p=4 και αριθμό διασπάσεων k=3 έχουμε:

Πλήθος Κομματιών $2^j=2^{p-i}$	**Μέγεθος** $2^{p-j}=2^i$		**Στιγμιότυπα και χρόνος που απαιτεί το καθένα**
1	16	j=0, i=4	T(4)
2	8	j=1, i=3	T(3) T(3)
4	4	j=2, i=2	T(2) T(2) T(2) T(2)
8	2	j=3, i=1	T(1) T(1) T(1) T(1) T(1) T(1) T(1) T(1)



$1^{η}$ ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (3/4)

- Ισχύει ότι **T(i)=T(i-1)+E(i)**, $\mathbf{i=i_0+1,\ldots, p}$ όπου

  T(i-1): χρόνος εκτέλεσης ενός στιγμιότυπου επόμενης διάσπασης,

  E(i): επιπλέον χρόνος λύσης στιγμιότυπου από χρόνο λύσης στιγμιότυπου επόμενης διάσπασης

- Π.χ. στη δυαδική αναζήτηση η μια σύγκριση που ανάγει πρόβλημα στην αναζήτηση μισής κάθε φορά λίστας από την προηγούμενη αναζήτηση είναι ο επιπλέον χρόνος. Ο χρόνος αυτός για όλες τις διασπάσεις **i** είναι **E(i)=1**.

- Πάντα γνωρίζω τον χρόνο που απαιτείται στην κατώτατη διάσπαση, δηλαδή η αρχική μου συνθήκη είναι $\mathbf{T(i_0)=b}$, όπου **b** ένας γνωστός αριθμός.

1^{η} ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (4/4)

- Η λύση της αναδρομικής συνθήκης με τη χρήση της αρχικής συνθήκης δίνει:

  T(i)=T(i-1)+E(i)

  =[T(i-2)+E(i-1)]+E(i)

  =[T(i-3)+E(i-2)]+E(i-1)+E(i)

  =…

$$= b + \sum_{j=i_0+1}^{i} \mathrm{E(j)}, \quad i = i_0, \ldots, p$$

- Ο χρόνος που απαιτείται για τη λύση του συνολικού προβλήματος μπορεί να υπολογιστεί θέτοντας **i=p**:

$$\mathrm{T(p)} = \mathrm{b} + \sum_{j=i_0+1}^{p} \mathrm{E(j)}$$

$2^{η}$ ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (1/2)

- Απαιτείται η λύση και των δυο υποπροβλημάτων που προκύπτουν από κάθε διάσπαση.
- Ισχύει ότι $\mathbf{T(i)=2T(i-1)+E(i)}$.
- Αν $\boldsymbol{\tau}\mathbf{(i)}$ είναι ο συνολικός χρόνος για να εκτελεστούν όλα τα στιγμιότυπα της διάσπασης i, τότε
  $\boldsymbol{\tau}\mathbf{(i)=2^{p-i}T(i)=2^{p-i}(2T(i-1)+E(i))=2^{p-i+1}T(i-1)+2^{p-i}E(i)}$.
- Όμως, $\boldsymbol{\tau}\mathbf{(i-1)=2^{p-(i-1)}T(i-1)}$. Οπότε,
  $\boldsymbol{\tau}\mathbf{(i)=}\boldsymbol{\tau}\mathbf{(i-1)+2^{p-i}E(i),\ i=i_0+1,\ldots,p}$
- Η αρχική συνθήκη είναι $\tau(\mathrm{i}_0)=2^{p-i_0}\mathrm{T}(\mathrm{i}_0)=2^{p-i_0}\mathrm{b}$

2^{η} ΠΕΡΙΠΤΩΣΗ ΕΦΑΡΜΟΓΗΣ ΜΕΘΟΔΟΥ ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ (2/2)

- Επομένως,

$$\tau(\mathrm{i}) = 2^{\mathrm{p}-\mathrm{i}_0} b + \sum_{j=i_0+1}^{i} 2^{p-j} \mathrm{E(j)}$$

- Ο χρόνος που απαιτείται για τη λύση του συνολικού προβλήματος μπορεί να υπολογιστεί θέτοντας **i=p**:

$$\tau(\mathrm{p}) = 2^{\mathrm{p}-\mathrm{p}} \mathrm{T(p)} = \mathrm{T(p)} = 2^{\mathrm{p}-\mathrm{i}_0} b + \sum_{j=i_0+1}^{p} 2^{p-j} \mathrm{E(j)}$$

ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ

- Εφαρμόζουμε την τεχνική διαίρει και βασίλευε.
- Παρατηρούμε ότι είμαστε στην πρώτη περίπτωση εφαρμογής της μεθόδου όπου ακολουθείται μια διαδρομή στο δέντρο διάσπασης.
- Αν **T(i)=αριθμός συγκρίσεων για να φτάσω σε κόμβο της i-στης διάσπασης**, τότε **T(i)=T(i-1)+1**
- Στην τελευταία διάσπαση όλες οι υπολίστες έχουν 1 στοιχείο και απαιτούν 1 σύγκριση. Άρα, **$T(i_0)=1$**.
- Από την ανάλυση της μεθόδου διαίρει και βασίλευε γνωρίζουμε ότι:

$$\mathrm{T(p)} = b + \sum_{j=i_0+1}^{p} \mathrm{E(j)} = 1 + p - i_0$$

όπου **$p-i_0$** είναι το βάθος του δέντρου διάσπασης.

- Όμως, για κάθε n υπάρχει ένας φυσικός αριθμός d: $2^d \le n < 2^{d+1}$ και στη χειρότερη περίπτωση $p - i_0 = d = \lfloor \log n \rfloor$

- Άρα, $\mathrm{T(p)} = 1 + \lfloor \mathrm{logn} \rfloor = \Pi\mathrm{X}\Pi\mathrm{(n)}$

ΑΛΓΟΡΙΘΜΟΣ QUICKSORT

Πρόβλημα: Έστω L μια αρχική λίστα από n στοιχεία. Θέλω να τη διατάξω σε αύξουσα σειρά.

Βασικός Κανόνας Αλγορίθμου

- Επιλέγω έναν από τους αριθμούς της λίστας, **x,** και διασπώ την αρχική λίστα **L** σε δυο υποπίνακες, ώστε όλες οι τιμές του **L** που είναι μικρότερες του **x** να βρίσκονται στον αριστερό υποπίνακα και όλες οι τιμές του **L** που είναι μεγαλύτερες του **x** να βρίσκονται στο δεξιό υποπίνακα.
- Κάνω το ίδιο με καθέναν από τους δυο υποπίνακες, οπότε παίρνω 4 υποπίνακες, όπου οι τιμές του πρώτου υποπίνακα είναι μικρότερες από τις τιμές του δεύτερου, οι τιμές του δεύτερου είναι μικρότερες από τις τιμές του τρίτου και οι τιμές του τρίτου είναι μικρότερες από τις τιμές του τέταρτου υποπίνακα.
- Η ίδια διαδικασία επαναλαμβάνεται για κάθε υποπίνακα μέχρι οι τιμές της λίστας να διαταχθούν σε αύξουσα σειρά.

ΑΛΓΟΡΙΘΜΟΣ QUICKSORT-ΨΕΥΔΟΚΩΔΙΚΑΣ

Αλγόριθμος QuickSort(L, k, m)

Είσοδος: Τυχαία λίστα L, πρώτη θέση λίστας k, τελευταία θέση λίστας m

Έξοδος: Ταξινομημένη λίστα L

```
if (k<m)
    κλήση υποαλγορίθμου διάσπασης divide(L, k, m, j)
    if (k<j-1) QuickSort(L, k, j-1)
    if (j+1<m) QuickSort(L, j+1, m)
end_if
```

ΑΛΓΟΡΙΘΜΟΣ QUICKSORT-ΥΠΟΑΛΓΟΡΙΘΜΟΣ ΔΙΑΣΠΑΣΗΣ

Αλγόριθμος Διάσπασης Divide(L, k, m, j)

Είσοδος: Δείκτες αρχικής και τελικής θέσης υπολίστας k, m και στοιχεία υπολίστας L

Έξοδος: Δείκτης j που ορίζει θέση x μετά τη διάσπαση, δηλαδή L(j) = x, νέα υπολίστα με L(i) <= x για k <= i <= j, L(i) >= x για j <= i <=m

- Θέτουμε **i=k**, **j=m+1** και **x=L(i).**
- Αυξάνουμε το **i** κατά 1 μέχρι **L(i)>=x**.
- Μειώνουμε το **j** κατά 1 μέχρι **L(j)<x**.
- Αν **i<j**, το **L(i)** ανήκει σε πάνω υπολίστα και αποθηκεύεται σε **j** και το **L(j)** ανήκει σε κάτω υπολίστα και αποθηκεύεται σε **i**.
- Αν **i=j** ή **i>j** μπαίνει το **x** σε θέση **L(j)**.

ΑΛΓΟΡΙΘΜΟΣ QUICKSORT-ΨΕΥΔΟΚΩΔΙΚΑΣ ΥΠΟΑΛΓΟΡΙΘΜΟΥ ΔΙΑΣΠΑΣΗΣ

```
i=k
j=m+1
x=temp=L(i)
while (i<j) do
    do
       i=i+1
    while (L(i)<temp) AND (i<m)
    do
       j=j-1
    while (L(j)>temp) AND (j>k)
    if (j>i) then swap(L(j), L(i))
end_while
swap(L(k), L(j))
return j
```

QUICKSORT-ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΧΕΙΡΟΤΕΡΗΣ ΠΕΡΙΠΤΩΣΗΣ

- Αν το **x** είναι ακραίο στοιχείο μιας λίστας **L** τότε μια από τις δυο υπολίστες είναι κενή. Οπότε παραβιάζεται η αρχή της εξισορρόπησης.

- Στην περίπτωση αυτή οι **m-k** συγκρίσεις γίνονται **n-1** συγκρίσεις στην πρώτη διάσπαση (**m=n**, **k=1**).

- Επίσης, στην περίπτωση αυτή οι **m-k** συγκρίσεις γίνονται **n-2** συγκρίσεις στη δεύτερη διάσπαση.

- Τελικά, **ΠΧΠ(n) = 1+2+…+(n-1)=n(n-1)/2** συγκρίσεις.

QUICKSORT-ΠΑΡΑΔΕΙΓΜΑ (1/4)

Πρόβλημα: Να διαταχθεί σε αύξουσα σειρά ο πίνακας L=[15,11, 8, 7, 6, 10, 16, 12] με τη χρήση του αλγορίθμου QuickSort.

Απάντηση:

Βήμα 1: Θέτουμε i=1, j=9 και x=L(i)=15.

Βήμα 2: Αυξάνω το i κατά 1 μέχρι L(i=7)>x.

Βήμα 3: Μειώνω το j κατά 1 μέχρι L(j=8)<x.

Βήμα 4: Επειδή i<j, έχουμε swap(L(7), L(8)).

Νέος Πίνακας: L=[15,11, 8, 7, 6, 10, 12, 16].

Βήμα 5: Αυξάνω το i κατά 1 μέχρι L(i=8)>x.

Βήμα 6: Μειώνω το j κατά 1 μέχρι L(j=7)<x.

Βήμα 7: Επειδή i>j, έχουμε swap(L(1), L(7)).

Νέος Πίνακας: L=[12, 11, 8, 7, 6, 10, 15, 16]

QUICKSORT-ΠΑΡΑΔΕΙΓΜΑ (2/4)

Βήμα 8: Εφαρμόζω QuickSort στην αριστερή υπολίστα L(1),…, L(6). Θέτουμε i=1, j=7 και x=L(i)=12.

Βήμα 9: Αυξάνω το i κατά 1 μέχρι i=6.

Βήμα 10: Μειώνω το j κατά 1 μέχρι L(j=6)<x.

Βήμα 11: Επειδή i=j, έχουμε swap(L(1), L(6)).

Νέος Πίνακας: L=[10, 11, 8, 7, 6, 12, 15, 16]

Βήμα 12: Εφαρμόζω QuickSort στην αριστερή υπολίστα L(1),…, L(5). Θέτουμε i=1, j=6 και x=L(i)=10.

Βήμα 13: Αυξάνω το i κατά 1 μέχρι L(i=2)>x.

Βήμα 14: Μειώνω το j κατά 1 μέχρι L(j=5)<x.

Βήμα 15: Επειδή i<j, έχουμε swap(L(2), L(5)).

Νέος Πίνακας: L=[10, 6, 8, 7, 11, 12, 15, 16].

QUICKSORT-ΠΑΡΑΔΕΙΓΜΑ (3/4)

Βήμα 16: Αυξάνω το i κατά 1 μέχρι L(i=5)>x.

Βήμα 17: Μειώνω το j κατά 1 μέχρι L(j=4)<x.

Βήμα 18: Επειδή i>j, έχουμε swap(L(1), L(4)).

Νέος Πίνακας: L=[7, 6, 8, 10, 11, 12, 15, 16].

Βήμα 19: Εφαρμόζω QuickSort στην αριστερή υπολίστα L(1),…, L(3).
Θέτουμε i=1, j=4 και x=L(i)=7.

Βήμα 20: Αυξάνω το i κατά 1 μέχρι L(i=3)>x.

Βήμα 21: Μειώνω το j κατά 1 μέχρι L(j=2)<x.

Βήμα 22: Επειδή i>j, έχουμε swap(L(1), L(2)).

Νέος Πίνακας: L=[6, 7, 8, 10, 11, 12, 15, 16].

QUICKSORT-ΠΑΡΑΔΕΙΓΜΑ (4/4)

Βήμα 23: Επειδή η αριστερή υπολίστα έχει 1 στοιχείο, L(1), αυτό είναι σωστά διατεταγμένο και δεν εφαρμόζω QuickSort.
Βήμα 24: Επειδή η δεξιά υπολίστα έχει 1 στοιχείο, L(3), αυτό είναι σωστά διατεταγμένο και δεν εφαρμόζω QuickSort. Βήμα 25: Επειδή η δεξιά υπολίστα από τη διάσπαση του βήματος 18 έχει 1 στοιχείο, L(5), αυτό είναι σωστά διατεταγμένο και δεν εφαρμόζω QuickSort.
Βήμα 26: Επειδή η δεξιά υπολίστα από τη διάσπαση του βήματος 7 έχει 1 στοιχείο, L(8), αυτό είναι σωστά διατεταγμένο και δεν εφαρμόζω QuickSort.
Τελικός Πίνακας: L=[6, 7, 8, 10, 11, 12, 15, 16].