Δομές Δεδομένων και Αλγόριθμοι

Χρήστος Γκόγκος

ΤΕΙ Ηπείρου

Χειμερινό Εξάμηνο 2014-2015
Παρουσίαση 7 - Merge Sort

Η αλγοριθμική προσέγγιση Διαίρει και Βασίλευε

1. Διαίρεση του προβλήματος σε 2 ή περισσότερα υποπροβλήματα
2. Επίλυση καθενός από τα υποπροβλήματα αναδρομικά. Αν έχουν μικρό μέγεθος επίλυση απευθείας
3. Συνδυασμός των λύσεων ώστε να αποκτηθεί η λύση στο αρχικό πρόβλημα

Εντοπισμός κίβδηλου κέρματος

Έστω ότι έχετε 16 χρυσά κέρματα και μια ζυγαριά. Γνωρίζετε ότι ένα από αυτά είναι κίβδηλο. Πως μπορείτε να το εντοπίσετε με μόνο 4 ζυγίσεις βαρών;

Εντοπισμός κίβδηλου κέρματος

Έστω ότι έχετε 16 χρυσά κέρματα και μια ζυγαριά. Γνωρίζετε ότι ένα από αυτά είναι κίβδηλο. Πως μπορείτε να το εντοπίσετε με μόνο 4 ζυγίσεις βαρών;
Χρησιμοποιώντας την τεχνική “Διαίρει και Βασίλευε”

Ταξινόμηση με συγχώνευση-Merge Sort

Υπάρχει αναφορά στο Merge-Sort από τον John Von Neuman στο 1945!

1. Διαίρεση της ακολουθίας των n στοιχείων σε 2 υποακολουθίες με $\frac{n}{2}$ στοιχεία η καθεμία
2. Ταξινόμησε αναδρομικά τις υποακολουθίες
3. Συγχώνευσε τις ταξινομημένες υποακολουθίες ώστε να σχηματιστεί η τελική ταξινομημένη ακολουθία

Ο αλγόριθμος MergeSort

MergeSort(A)

// Η ακολουθία A έχει n στοιχεία
if $n = 1$
    return A
else
    Υπολογισμός της θέσης m του μεσαίου στοιχείου
    MergeSort(A_l) // A_l είναι το πρώτο μισό του A
    MergeSort(A_r) // A_r είναι το δεύτερο μισό του A
    Merge(A_l, A_r) // συγχώνευση A_l και A_r

Η συνάρτηση merge της Merge Sort

```
//  a = Sorted array of  size  n/2
//  b = Sorted array of  size  n/2
//  c = Resulting array  having  all  items of  a and b sorted
 int  i=0,  j=0,  k=0;
while  (( i<n/2) && (j<n/2)){
     if  (a[i]<b[j ]){
         c[k]=a[i ];
         i++;
    }  else  {
         c[k]=b[j ];
         j++;
    }
    k++;
}
//  handle items from either  a or  b  that  are  not  already placed into
      c
```

Ανάλυση της απόδοσης της Merge Sort

Αν κατασκευαστεί το δένδρο αναδρομών (στο οποίο απεικονίζονται ως κλάδοι ενός δένδρου οι κλήσεις των αναδρομικών συναρτήσεων) μπορούν να διαπιστωθούν τα ακόλουθα:

- Το ύψος του δένδρου είναι $log_2 n$
- Ο αριθμός των επιπέδων του δένδρου είναι $log_2 n + 1$
- Σε κάθε επίπεδο j υπάρχουν 2^j υποπροβλήματα και το καθένα από αυτά έχει μέγεθος $\frac{n}{2^j}$

Αν θεωρήσουμε ότι η συγχώνευση των ταξινομημένων πινάκων εκτελείται σε cn χρόνο τότε ο αλγόριθμος Merge Sort εκτελείται σε $cnlog_2 n + cn$ χρόνο

Ανάλυση της απόδοσης της Merge Sort

Αν κατασκευαστεί το δένδρο αναδρομών (στο οποίο απεικονίζονται ως κλάδοι ενός δένδρου οι κλήσεις των αναδρομικών συναρτήσεων) μπορούν να διαπιστωθούν τα ακόλουθα:

- Το ύψος του δένδρου είναι $log_2 n$
- Ο αριθμός των επιπέδων του δένδρου είναι $log_2 n + 1$
- Σε κάθε επίπεδο j υπάρχουν 2^j υποπροβλήματα και το καθένα από αυτά έχει μέγεθος $\frac{n}{2^j}$

Αν θεωρήσουμε ότι η συγχώνευση των ταξινομημένων πινάκων εκτελείται σε cn χρόνο τότε ο αλγόριθμος Merge Sort εκτελείται σε $cnlog_2 n + cn$ χρόνο
Αυτό συμβαίνει διότι ο αριθμός των λειτουργιών που εκτελούνται σε κάθε επίπεδο j είναι ίσος με: (αριθμό υποπροβλημάτων) $\times$ (μέγεθος κάθε υποπροβλήματος) $2^j c \frac{n}{2^j} = cn$

Και λόγω του ότι ο αριθμός των επιπέδων στο δένδρο είναι $log_2 n + 1$ προκύπτει ότι ο συνολικός αριθμός λειτουργιών είναι $cn(log_2 n + 1) = cnlog_2 n + cn$