Κωνσταντίνος Σκορδούλης

ΑΜ: 1115 2016 00155

Sort Function (worst case)

void sort(Item a[], int l, int r)

{ int i, j;

for (i = l+1; i <= r; i++)

for (j = i; j > l; j--)

compexch(a[j-1], a[j]);

}

Η συνάρτηση αυτή αποτελείται από δύο **for**(όπου η μία είναι εμφωλευμένη, δηλαδή **for{ for{ } }**). Για να υπολογίσουμε την πολυπλοκότητα του **worst case**, θα πρέπει να πάρουμε την εξής περίπτωση:

(η **sort** ξεκινά την διαδικασία της, από το **L** στοιχείο του πίνακα και τελειώνει στο στοιχείο **r**)

* **L=0=>i=1** Σε αυτή την περίπτωση η **«εξωτερική»** **for**, εκτελείται(i=1;i<= r;i++) **r φορές**, ενώ η **«εσωτερική» for**, εκτελείται (j=i++=2; j>L=0;j--), δηλαδή **δύο φορές**(γιατί πάντα εκτελείται από **L+2 μέχρι L; j--**).
  + Δηλαδή ο χρόνος εκτέλεσης θα είναι (**a\*2)\*r + b0** (όπου **a** ο χρόνος εκτέλεσης του **compexch(a[j-1], a[j])** (όπου, όπως είπαμε στο φροντιστηριο, είναι σταθερού χρόνου Ο(1))και **b** ο χρόνος εκτέλεσης του **int i,j;**).
  + Διώχνοντας τα **a και b0** , κρατάμε τον **dominant όρο,** ο οποίος είναι ο **r.**
  + Δηλαδή η **πολυπλοκότητα** της συνάρτησης είναι Ο(r) ή καλύτερα **Ο(n)**.

Main

int main(int argc, char \*argv[]){

int i, N = atoi(argv[1]);

sw =atoi(argv[2]);

int \*a = malloc(N\*sizeof(int));

if (sw)

for (i = 0; i < N; i++)

a[i] = 1000\*(1.0\*rand()/RAND\_MAX);

else

while (scanf("%d", &a[N]) == 1) N++;

sort(a, 0, N-1);

for (i = 0; i < N; i++) printf("%3d ", a[i]);

printf("\n");

}

Καταρχάς, υποθέτουμε ότι ο χρόνος εκτέλεσης των 3 πρώτων εντολών είναι **b1**. Έχουμε λοιπόν 2 περιπτώσεις, για την **for**:

* **sw=0** Εκτελείται **Ν** φορές η εμφωλευμένη **for**, όπου η εντολή a[i]=……; έχει σταθερό χρόνο εκτέλεσης **b2** , καθώς η **rand()**  εκτελείται σε σταθερό χρόνο. Θα έχουμε **Ν\*b2**
* **sw!=0** Εκτελείται **Μ** φορές η **while** μέχρι η **scanf** να επιστρέψει 0,δηλαδή να ξεμείνουμε από μνήμη(η εντολή Ν++ έχει χρόνο εκτέλεσης **b3**).Θα έχουμε **M\*b3**.

Στη συνέχεια , η πολυπλοκότητα της **sort** είναι της μορφής **2\*a\*n+b0 (O(n))**, ακολουθεί μια **for** που εκτελείται **Ν φορές** , άρα **Ν\*b4**  και τελειώνουμε με μία printf , που εκτελείται σε σταθερό χρόνο **b5**

## Συνοψίζοντας:

* **sw=0 => f(n)=b1+(n\*b2)+(2\*a\*n+b0)+(n\*b4)+b5=>**

**f(n)=(a+b2+b4)\*n+b1+b3+b5=> O(n)**

* **sw!=0=>f(n,M)=b1+(M\*b3) +(2\*a\*n+b0)+(n\*b4)+b5=>**

**f(n,M)=(2\*a+b4)\*n+(b3 \*M)+(b1+b5)**

Επειδή **n** και **Μ** είναι δύο **διαφορετικές μεταβλητές ίδιου βαθμού,** μπορούμε να καταλήξουμε στο συμπέρασμα ότι η πολυπλοκότητα είναι και αυτή **O(n)**