Camp Προετοιμασίας
Πανελλήνιος Διαγωνισμός Πληροφορικής
Αθήνα, 11-16 Απριλίου 2011

Πολυπλοκότητα

Γεράσιμος Αρσένης

Βασικό Ερώτημα

- Ανάμεσα σε δύο αλγόριθμους, πώς μπορούμε να εκτίμησουμε ποιός είναι αποδοτικότερος;

Με τον όρο επίδοση αλγορίθμου εννοούμε:

- **Το χρόνος εκτέλεσης**
- Τις απαιτήσεις σε μνήμη

Απο τί εξαρτάται ο χρόνος εκτέλεσης;

- **Τα δεδομένα εισόδου**
- Υπολογιστικό περιβάλλον(ταχύτητα CPU, λειτουργικό σύστημα κτλ.)
- Υλοποίηση (γλώσσα προγραμματσμού κτλ.)

Μέγεθος εισόδου

- Συνήθως το πλήθος των στοιχείων προς επεξεργασία. (π.χ. πλήθος αριθμών προς ταξινόμηση)
- Ο χρόνος εκτέλεσης εξαρτάται και από τα ίδια τα στοιχεία. (π.χ. αν είναι ήδη ταξινομημένα)
- Για συγκεκριμένο μέγεθος εισόδου (N), μας ενδιαφέρει η χειρότερη, η μέση και η καλύτερη περίπτωση χρόνου εκτέλεσης.

Εκτίμηση χρόνου εκτέλεσης (Πολυπλοκότητα)

Λόγω της εξάρτησης του χρόνου εκτέλεσης από το υλικό, τη γλώσσα προγραμματισμού κτλ είναι δύσκολο να τον υπολογίσουμε επακριβώς και έτσι μας ενδιαφέρει περισσότερο να μπορούμε να κάνουμε μια προσεγγιστική εκτίμηση ως συνάρτηση του μεγέθους της εισόδου.

Υπολογισμός Πολυπλοκότητας

- Στην πράξη, μπορούμε συνήθως να καταλάβουμε την πολυπλοκότητα κατευθείαν από τον κώδικα.
- Υπολογίζουμε μία σχέση που εκφράζει πόσες περίπου πράξεις εκτελεί ο αλγόριθμος για είσοδο μεγέθους N.
- Το συμβολίζουμε O(...) όπου μέσα στην παρένθεση είναι μία πίο απλή μορφή της σχέσης που βρήκαμε (βλ. επόμενη διαφάνεια)

Υπολογισμός Πολυπλοκότητας

Γενικός κανόνας:

- Βρίσκουμε τον αριθμό επαναλήψεων κάθε for (ή while) στη χειρότερη περίπτωση (αγνοώντας σταθερούς όρους).
- **Προσθέτουμε** το πλήθος επαναλήψεων κάθε for αν αυτά είναι στη σειρά.
- **Πολλαπλασιάζουμε** το πλήθος επαναλήψεων εάν βρίσκονται το ένα μέσα στο άλλο.
- Κρατάμε μόνο τους μεγιστοβάθμιους όρους και αγνοούμε τους σταθερούς συντελεστές.

Παραδείγματα

```
//Είσοδος: N, A[N]
sum = 0;
p = 1;
for (i=1; i <= N; i++)
{
    sum = sum + A[i];
    p = p + A[i];
}
```

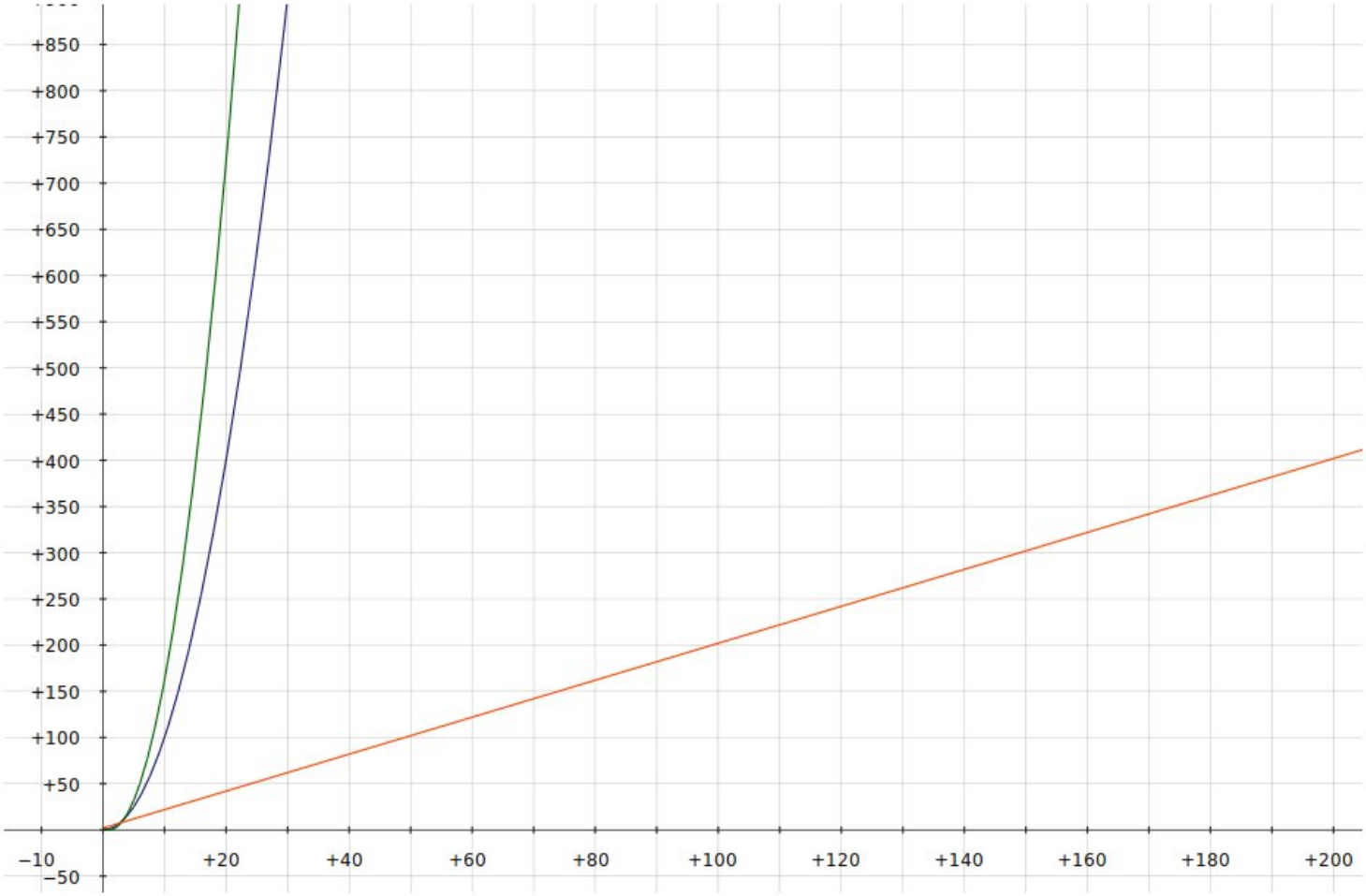
O(N)

Παραδείγματα

```
//Είσοδος: N
for (i=1; i<=N; i++)
    for (j=1; j<=N; j++)
        A[i][j] = i + j;
```

O(N^2)

Παραδείγματα



O(N^2) (Υλοποίηση 1)

O(N^2) (Υλοποίηση 2)

O(N)

Παραδείγματα

- 2N+2 = O(N)
- 10N+1 = O(N)
- 10N^2 = O(N^2)
- N^2+20N+10 = O(N^2)
- N^3 + N + 1 = O(N^3)
- 5 = O(1)
- 2^N + N^5 = O(2^N)

Πολυπλοκότητα

Οι πιό συνηθισμένες πολυπλοκότητες αλγορίθμων είναι:

- O(1) (σταθερός χρόνος)
- O(logN) (λογαριθμικός)
- O(N) (γραμμικός)
- O(NlogN)
- O(N^2) (τετραγωνικός)
- O(N^3)
- ...
- O(2^N) (εκθετικός)
- O(N!)

Πολυπλοκότητα

N \ O()	logN	N	NlogN	N^2	2^N	N!
10	0,003 μs	0,01 μs	0,033 μs	0,1 μs	1 μs	3,63 ms
20	0,004 μs	0,02 μs	0,086 μs	0,4 μs	1 ms	77,1 years
30	0,005 μs	0,03 μs	0,147 μs	0,49μs	1 sec	8,4*10^15
40	0,006 μs	0,04 μs	0,213 μs	1,6 μs	18,3 min	
50	0,006 μs	0,05 μs	0,282 μs	2,5 μs	13 days	
100	0,007 μs	0,1 μs	0,644 μs	10 μs	4*10^13	
1.000	0,010 μs	1 μs	9,966 μs	1 ms		
100.000	0,017 μs	0,10 ms	1,67 μs	10 sec		
1.000.000	0,020 μs	1 ms	19,93 ms	16,7 min		
10.000.000	0,023 μs	0,01 sec	0,23 sec	1,16 days		

Παραδείγματα

```
cnt = 0;
for (i=1; i <= N; i++)
     for (j=1; j <= N; j++)
          for (k=1; k <= N; k++)
               if (i+j+k == 5) cnt++;
```

O(N^3)

Παραδείγματα

```
k = 0;
for (i=1; i <= N; i++)
      for (j=1; j <= i; j++)
      {
            c[i][j] = k;
            k++;
      }
```

O(N^2)

Παραδείγματα

```
for (i=1; i <= (N*N / 2); i++)
     for (j=1; j <= N; j++)
          printf("%d\n", i*j);
```

Θ(N^3)

Παράδειγμα

```
int result=0;                                          //  1
for (int i=0; i<N; i++)                                //  2
  for (int j=i; j<N; j++) {                            //  3
    for (int k=0; k<M; k++) {                          //  4
      int x=0;                                         //  5
      while (x<N) { result++; x+=3; }                  //  6
    }                                                  //  7
    for (int k=0; k<2*M; k++)                          //  8
      if (k%7 == 4) result++;                          //  9
  }                                                    // 10
```

Εδώ η πολυπλοκότητα θα εξάρταται από δύο μεταβλητές (N και M)

O(M N^3)