Сложност на алгоритми

гл.ас. д-р.Нора Ангелова

Оценка на програма

- Времева сложност — оценка на време за изпълнение
- Пространствена (обемна) сложност — оценка на използвана памет

Пример

```
int n=10;
int sum=0;
for(int i=0; i<n; i++)
    sum++;
```

- За инициализацията е необходимо константно време a+b
- За i=0 времето е c
- За изпълнение на цикъла - p*n
  Общо време = a+b+... = p*n+q, където p,q-const

Асимптотична нотация

- n – размер на входните данни
- Формалното оценяване на сложността на алгоритмите при "достатъчно голямо" *n*, т.е. клонящо към безкрайност.

- **O(*F*)** - определя множеството от всички функции *f*, които нарастват **не** по-бързо от *F*, т.е. съществува константа c > 0 такава, че $f(n) <= cF(n)$.
- **θ(*F*)** - определя множеството от всички функции *f*, които нарастват толкова бързо, колкото и *F* (с точност до константен множител), т.е. съществуват константи $c_1 > 0$ и $c_2 > 0$ такава, че $c_1F(n) <= f(n) <= c_2F(n)$.
- **Ω(*F*)** - определя множеството от всички функции *f*, които нарастват **не** по-бавно от *F*, т.е. съществува константа c > 0 такава, че $f(n) >= cF(n)$.

$n \to \infty$

O(F), свойства

- Елементарна операция (не зависи от размера на обработваните данни) - O(1)
- Рефлексивност: $f \in O(f)$;
- Транзитивност: ако $f \in O(g)$, $g \in O(h)$, то $f \in O(h)$;
- Транспонирана симетрия: ако $f \in \Omega(g)$, то $g \in O(f)$ и обратно;
- За всяко $k > 0$, $kf \in O(f)$;
- $n^r \in O(n^s)$, за $0 < r < s$.
- Нарастването на сума от функции: $f + g \in \max(O(f), O(g))$;
- Композиция на оператори - $f * g \in O(f * g)$;
- Условни оператори - определя се от асимтотично най-бавния между условието и различните случаи;
- Цикли, вложени цикли - $O(n)$, $O(n^p)$

Алгоритмична сложност

```
for (sum = 0, i = 0; i < n; i *= 2)
    sum++;
```

i =1, 2, 4, ..., 2^k, ... докато **i < n**.
Цикълът се изпълнява [log *n*] пъти.

Сложността е $O(\log n)$.

Двоично търсене

- Броим обръщенията към елементите на масива.
- В рекурсивната функция се разглежда средния елемент и се прави едно рекурсивно извикване с два пъти по-малък масив.
- Нека $T(n)$ е функцията, която задава броя на обръщенията

Следователно:

$T(n) = T(n/2) + 1 = T(n/4) + 2 = T(n/2^k) + k$
При $n = 2^k \Rightarrow T(n) = T(1) + \log n$

Сложността на алгоритъма е $O(\log n)$.

Недостатъци на асимптотична нотация

- Оценен алгоритъм да е по-бавен от алгоритъм с по-лоша оценка
- Два алгоритъма с еднаква сложност може да са различно бързи.

Quick sort

Сложност:

- В най-лошия случай - $O(N^2)$
- В средния случай - $O(N \cdot log(N))$
- В средния случай - $O(N)$

Оценка

- Най-често използвани функции, подредени по скорост на нарастване

`c, log n, n, n.log n, n^2, n^3, 2^n, n!, n^n`

Благодаря за вниманието