Complexitate timp

Facultatea de Matematică şi Informatică

Lecţii de pregătire pentru admitere

2017

Algoritmi şi complexitate

- Enunţ
- Algoritmi de rezolvare
  - Care algoritm este mai bun?
- Datele de intrare
  - Forma datelor de intrare influenţează performanţele unui algoritm
- Implementare
- Cum evaluăm un algoritm?
  - Timp de execuţie. Timpul depinde de maşina pe care rulează programul. Ne trebuie o metodă de evaluare care să nu depindă de maşină => Complexitate timp
  - Spaţiu de execuţie. De foarte multe ori la fel de important ca timpul de execuţie şi în strânsă legătură cu acesta => Complexitate spaţiu

Dimensiunea datelor de intrare

# of records	10	20	50	100	1000	5000
algorithm 1	0.00s	0.01s	0.05s	0.47s	23.92s	47min
algorithm 2	0.05s	0.05s	0.06s	0.11s	0.78s	14.22s

- Valoarea ***M*** pentru care trebuie formulat răspunsul
- Număr de elemente: vector cu ***N*** elemente
- Dimensiunea propriu-zisă a datelor: int, long, long long
- Stabilim dimensiunea datelor de intrare după un set de variabile $N_1, N_2, \ldots, N_k$
- Determinăm complexitatea timp ca funcţie de aceste variabile

$$f(N_1, N_2, \ldots, N_k)$$

Numărarea pas-cu-pas

C

```
int result=0,N,M;
for (int i=0; i<N; i++)
 for (int j=i; j<N; j++)
  {
    for (int k=0; k<M; k++)
     {
      int x=0;
      while (x<N)
       {
        result++;
        x+=3;
       }
     }
    for (int k=0; k<2*M; k++)
     if (k%7 == 4) result++;
  }
```

PASCAL

```
var result,i,j,k,x,N,M:integer;
result:=0;
for i := 0 to n-1 do
 for j := i to n-1 do
  begin
    for k := 0 to M-1 do
     begin
      x:=0;
      while x<N do
       begin
        inc(result);
        x:=x+3;
       end;
     end;
    for k := 0 to 2*M-1 do
     if k mod 7 = 4 then inc(result);
  end;
```

Calculați funcția de complexitate ***f(N,M)*** a programului de mai sus.

Definiţii formale

- Big-*O*
  - Spunem că ***f (N)*** este ***O(g(N))*** dacă există ***c*** şi N_0 astfel încât: pentru orice $N > N_0$ avem $f(N) < c*g(N)$
  - *Exemplu 1*: dacă $f(N) = 3N(N-1)/2 + N = 1{,}5N^2 - 0{,}5N$, spunem că *f (N)* este $O(N^2)$
- Big-Ω
  - Spunem că ***f (N)*** este ***Ω(g(N))*** dacă ***g(N)*** este ***O(f(N))***
- Big-*Θ*
  - Spunem că ***f (N)*** este ***Θ(g(N))*** dacă ***f (N)*** este ***O(g(N))*** *şi* ***g(N)*** este ***O(f(N))***

Spunem că un algoritm (sau un program) are complexitatea ***O(g(N))*** dacă funcția sa de complexitate ***f (N)*** este ***O(g(N))***.

- Programul prezentat anterior are complexitatea $O(N^3M)$.

Funcţii uzuale de complexitate

- *f (N) este **O(log N)**: complexitate logaritmică (nu contează baza!)*
- *f (N) este **O(N)**: complexitate liniară*
- *f (N) este **O(N^2)**: complexitate pătratică*
- *f (N) este **O(N^3)**: complexitate cubică*
- *f (N) este **O(N^k)**, k* oarecare*: complexitate polinomială*
- *f (N) este **O(2^N)**: complexitate exponenţială (nu contează baza!)*

- *f (N) este **O(1)**: complexitate constantă*

Alt exemplu

C

```
int j=0,N,D;
for (int i=0; i<N; i++)
 {
  while((j<N-1)&&(A[i]-A[j]>D))
   j++;
  if (A[i]-A[j] == D)
   return 1;
 }
```

PASCAL

```
var i,j,N,D:integer;
j:=0;
for i := 0 to N-1 do
  begin
   while (j<N-1)&&(A[i]-A[j]>D) do
    inc(result);
   if (A[i]-A[j] == D) then exit;
  end;
```

Calculați funcția de complexitate ***f(N)*** a programului de mai sus.

Bibliografie

- Această prezentare este o prelucrare după articolul “Computational Complexity”, autor Michal Forišek (misof), care se găseşte pe Topcoder.

https://www.topcoder.com/community/data-science/data-science-tutorials/computational-complexity-section-1/