Algoritmus

Az algoritmus olyan elemi műveletekből kompozíciós szabályok szerint felépített összetett művelet, amelyet megadott feltételt teljesítő bemeneti adatra végrehajtva, a megkívánt kimeneti adatot eredményezi.

Számítási Probléma

A számítási probléma bemenet/kimenet feltételpárral meghatározott követelmény. Azt írja elő, hogy a bemeneti feltételt teljesítő adatra a kimeneti feltételt teljesítő adatot kell kiszámítani. Probléma bemenete: egy olyan (esetleg összetett) adat, amely teljesíti a bemeneti feltételt. Például, a rendezési probléma a következőt jelenti. Bemenet: Azonos típusú adatok $H = \{a_1, ..., a_n\}$ halmaza, amelyeken értelmezett egy $\leq$ lineáris rendezési reláció. Kimenet: A H halmaz elemeinek egy rendezéstartó felsorolása, tehát olyan $\{b_1, ..., b_n\}$ sorozat, amelyre $b_1 \leq b_2 \leq \cdots \leq b_n$, és $H = \{b_1, ..., b_n\}$.

Specifikáció

$$\{B\}A\{K\}$$

- B logikai formula, a bemeneti feltétel,
- K logikai formula, a kimeneti feltétel,
- A az algoritmus, amelyre az állítás vonatkozik.

Az A algoritmus helyes a $\{B\}A\{K\}$ specifikációra nézve, ha minden X bemeneti adatra, amelyre teljesül a B bemeneti feltétel, az algoritmus végrehajtása véges sok elemi művelet végrehajtása után befejeződik, és a keletkezett kimeneti adatra teljesül a K kimeneti feltétel.

Beszúró Rendezés

```
BESZÚRÓ-RENDEZÉS (A)
 For j= 2 to hossz(A)
        {kulcs:=A[j]
         i:=j-1
         while i>0 és A[i]>kulcs
                 {A[i+1]:=A[i]
                 i:=i-1}
         A[i+1]:=kulcs}
```

Példa

$(5,2,4,6,1,3)$
$(5,2^j,4,6,1,3)$
$(2,5,4,6,1,3)$
$(2,5,4^j,6,1,3)$
$(2,4,5,6,1,3)$
$(2,4,5,6^j,1,3)$
$(2,4,5,6,1^j,3)$
$(1,2,4,5,6,3)$
$(1,2,4,5,6,3^j)$
$(1,2,3,4,5,6)$

További példa:

http://www.youtube.com/watch?v=fdGvEZzYYRcfeature=related

Futási idő elemzése

Legyen A az $e_1, \ldots, e_m$ elemi műveletekből felépített algoritmus, és jelölje t_i az e_i művelet futási idejét (konkrét, vagy hipotetikus gépen). A t_i futási idő függhet az e_i művelet argumentumaitól. A továbbiakban feltételezzük, hogy minden e_i elemi művelet futási ideje c_i konstans. Adott x bemenetre jelölje $T(A,x)$ az A algoritmus tényleges futási idejét, ami az x bemeneti adatra ténylegesen végrehajtott elemi műveletek futási idejének az összege. Jelölje $|x|$ az x bemeneti adat méretét. Ez a méret összetett adat (pl. halmaz, vagy sorozat) esetén általában az adatok száma. Nem összetett adat esetén pedig általában az adat értéke. Az e_i elemi műveletek t_i futási ideje és az $|x|$ méret függvény együttesen adja a bonyolultsági mértéket. Algoritmusnak más költsége is van, nevezetesen a memóriaigény, és elosztott algoritmusok esetén fontos a kommunikációs költség is.

Legjobb eset:
$T_{lj}(A,n) = \min\{T(A,x) : |x| = n\}$

Legrosszabb eset:
$T_{lr}(A,n) = \max\{T(A,x) : |x| = n\}$

Átlagos eset:
Jelölje $Pr(x)$ annak valószínűségét, hogy x bemeneti adata lesz az A algoritmusnak.
$T_a(A,n) = \sum_{|x|=n} Pr(x)T(A,x)$

Elemzés

Legyen t_j a for ciklus j-edik végrehajtásában a while ciklus tesztelések száma.

```
BESZÚRÓ-RENDEZÉS (A)                költség végrehajtások száma
 For j= 2 to hossz(A)                 c1     n
        {kulcs:=A[j]                  c2     n-1
         i:=j-1                       c3     n-1
         while i>0 és A[i]>kulcs      c4     t2+t3+...+tn
                 {A[i+1]:=A[i]        c5     (t2-1)+...+(tn-1)
                 i:=i-1}              c6     (t2-1)+...+(tn-1)
         A[i+1]:=kulcs}               c7     n-1
```

A teljes költség:

$$c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1)$$

Legjobb eset:
Ha a tömb már rendezett, akkor minden j-re $t_j = 1$, tehát

$$T_{lj}(A,n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1)$$

Ez n lineáris függvénye.

Legrosszabb eset:
Ha a tömb már fordított sorrendben rendezett, akkor minden j-re $t_j = j$, tehát

$$T_{lr}(A,n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4(n(n+1)/2 - 1) +$$

$$c_5((n-1)n/2-1)+c_6((n-1)n/2-1)+c_7(n-1),$$

mivel $\sum_{j=2}^{n} j = n(n+1)/2-1$.
A legrosszabb eset n négyzetes függvénye.

Aszimptotikus jelölések

O-jelölés

$$O(g(n)) = \{f(n) : (\exists c, n_0 \geq 0)(\forall n \geq n_0)(0 \leq f(n) \leq cg(n))\}$$

Az $f(n) = O(g(n))$ jelentése: $f(n) \in O(g(n))$.

Ω-jelölés

$$\Omega(g(n)) = \{f(n) : (\exists c, n_0 > 0)(\forall n \geq n_0)(cg(n) \leq f(n))\}$$

Az $f(n) = \Omega(g(n))$ jelentése: $f(n) \in \Omega(g(n))$.

Θ-jelölés

$$\Theta(g(n)) = \{f(n) : (\exists c_1, c_2, n_0 > 0)(\forall n \geq n_0)(c_1 g(n) \leq f(n) \leq c_2 g(n))\}$$

Az $f(n) = \Theta(g(n))$ jelentése: $f(n) \in \Theta(g(n))$.

Aszimptotikus jelölések

o-jelölés

$$o(g(n)) = \{f(n) : (\forall c > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)(0 \leq f(n) \leq cg(n))\}$$

Az $f(n) = o(g(n))$ jelentése: $f(n) \in o(g(n))$, ekkor $f(n)/g(n) \to 0$.

ω-jelölés

$$\omega(g(n)) = \{f(n) : (\forall c > 0)(\exists n_0 \geq 0)(\forall n \geq n_0)(cg(n) \leq f(n))\}$$

Az $f(n) = \omega(g(n))$ jelentése: $f(n) \in \omega(g(n))$, akkor $f(n)/g(n) \to \infty$.

Példák

- $2n+20 = \Theta(n)$
- $n^2 = \omega(n \log n)$
- $\log_2(n) = \Theta(\log_{10} n)$
- $n^2 + 20n + 10 = \Theta(n^2)$
- $2^n = \omega(n^{100000})$

A konstans költségű elemi műveletek mindegyikét számolhatjuk egységnyi költségűnek.

Fő tulajdonságok

- A Θ reláció tranzitív, reflexív és szimmetrikus, így egy osztályozás tartozik hozzá.

- A Θ, O, Ω, o, ω relációk mindegyike tranzitív.
- Ha $f(n) = O(g(n))$, akkor $g(n) = \Omega(f(n))$.
- Ha $f(n) = o(g(n)$, akkor $f(n) = O(g(n))$.
- Ha $f(n) = \omega(g(n))$, akkor $f(n) = \Omega(g(n))$.

Rekurzív algoritmusok

Egy objektum definícióját rekurzívnak nevezünk, ha a definíció tartalmazza a definiálandó objektumot.

Egy P eljárást (vagy függvényt) rekurzívnak nevezünk, ha P utasításrészében előfordul magának a P eljárásnak a hívása.

```
Faktor(n)
  If n=1 return 1
  Else return n*Faktor(n-1)
```

Példa:
Faktor(5)=5*Faktor(4)
Faktor(4)=4*Faktor(3)
Faktor(3)=3*Faktor(2)
Faktor(2)=2*Faktor(1)
Faktor(1)=1

Euklideszi algoritmus

- Ha b=0, akkor $lnko(a,b) = a$.
- Egyébként ha $a \geq b$, akkor $lnko(a,b) = lnko(b, a \mod b)$.

Tehát, ha $a \geq b$, akkor az alábbi algoritmus kiszámolja a legnagyobb közös osztót.

```
Euklidesz(a,b)
  If b=0 return a
  Else return Euklidesz(b, a mod b)
```

$Euklidesz(64,28) = Euklidesz(28,8) = Euklidesz(8,4) = Euklidesz(4,0) = 4$

Partíció probléma

Az n természetes szám egy partíciója olyan $\pi = (a_1, \ldots, a_k)$ sorozat, amelyre teljesül:
- $a_1 \geq a_2 \geq \cdots \geq a_k > 0$
- $\sum_{i=1}^{k} a_i = n$

Jelölje $P(n)$ az n szám összes partíciójának számát.

- P(1)=1 (1=1)
- P(2)=2 (2=2, 2=1+1)

- P(3)=3 (3=3, 3=2+1, 3=1+1+1)
- P(4)=5 (4=4, 4=3+1, 4=2+2, 4=2+1+1, 4=1+1+1+1)

Partíciószám számító algoritmus

Jelölje $P2(n,k)$ n azon partícióinak számát, amelyben minden elem legfeljebb k. Például $P2(4,3) = 4$, $P2(4,2) = 3$.

Ekkor a következő összefüggések teljesülnek:

- $P2(1,k) = 1, P2(n,1) = 1$
- $P2(n,n) = 1 + P2(n,n-1)$ ($n = n$ és a többiek)
- $P2(n,k) = P2(n,n)$ ha $n < k$
- $P2(n,k) = P2(n,k-1) + P2(n-k,k)$ ha $k < n$ (k benne van vagy nincs)
- A megoldás: $P(n) = P2(n,n)$

```
PARTÍCIÓ(n)
  Return P2(n,n)
P2(n,k)
  If (k=1) Or (n=1)  return 1
  If k>=n  return  P2(n,n-1)+1
  return P2(n, k-1)+P2(n-k, k)
```

Rekurziós fa: Olyan fa, amelynek minden pontja egy eljáráshívást jelent adott aktuális paraméterekre, úgy, hogy a pont fiai megfelelnek azoknak az eljáráshívásoknak, amelyek végrehajtódnak az aktuális paraméterek esetén.

Futási idő elemzése

Legyen $E(n,k)$ a $P2(n,k)$ eljáráshívás hatására végrehajtott eljárásutasítások száma.
Ekkor $P2(n,k) \leq E(n,k) \leq 2P2(n,k) - 1$.

Lemma $P(n) = \Omega(2^{\sqrt{n}})$

Bizonyítás Tekintsük az összes olyan $[a_1, \ldots, a_k]$ sorozatot, ahol $\lfloor \sqrt{n} \rfloor \geq a_1$ és $\sqrt{n} \geq a_1 > a_2 > \cdots > a_k > 1$. Ezek száma pontosan $2^{\lfloor \sqrt{n} \rfloor - 1}$, mivel minden ilyen sorozat egyértelműen megadható egy $b_{\lfloor \sqrt{n} \rfloor - 1}, \ldots, b_1$ bitvektorral, ahol $b_x = 1$ akkor és csak akkor, ha $x+1$ eleme a sorozatnak.

Mivel $k \leq \sqrt{n}$, ezért $\sum_{i=1}^{k} a_i \leq n$. Minden ilyen sorozathoz adjunk hozzá a végén annyi 1-est, hogy a számok összege n legyen, tehát n egy partícióját kapjuk. Ha két kiindulási sorozat különböző volt, akkor a kiegészítéssel különböző partíciót kapunk.

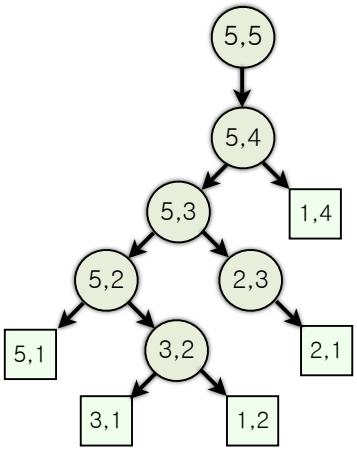
Következmény. A P algoritmus futási ideje $\Omega(2^{\sqrt{n}})$.

Rekurzív algoritmus helyességének bizonyítása

I. Terminálás bizonyítása

Bebizonyítandó, hogy minden eljáráshívás végrehajtása véges lépésben befejeződik. (Az eljárás terminál.)

Megállási feltétel: Legyen $P(x_1, \ldots, x_n)$ egy n-paraméteres rekurzív eljárás. Az $M(x_1, \ldots, x_n)$ egész értékeket felvevő kifejezés megállási feltétele a P rekurzív eljárásnak, ha



1. ábra.

- 1. $M(a_1,\ldots,a_n) \geq 0$ minden megengedett $a_1,\ldots,a_n$ aktuális paraméterre.
- 2. Ha $M(a_1,\ldots,a_n) = 0$ akkor nincs rekurzív hívás $P(a_1,\ldots,a_n)$ végrehajtásakor.
- 3. Ha van rekurzív hívás $P(a_1,\ldots,a_n)$ végrehajtásakor valamely $b_1,\ldots,b_n$ paraméterekre, akkor $M(b_1,\ldots,b_n) < M(a_1,\ldots,a_n)$.

Példa: $M(n,k) = (n-1) \times (k-1)$ megállási feltétel P2-re.

II. Helyesség bizonyítása

1. Alaplépés. Annak bizonyítása, hogy az eljárás helyes eredményt számít ki, ha az aktuális paraméterek esetén nincs rekurzív hívás.

2. Rekurzív lépés. Feltéve, hogy minden rekurzív hívás helyes eredményt ad, annak bizonyítása, hogy a rekurzív hívások által adott értékekből az eljárás helyes eredményt számít ki.

Összefésülő Rendezés

Oszd meg: A rendezendő sorozatot felosztjuk két $n/2$ méretű részsorozatra.

Uralkodj: A két részsorozatot összefésülő rendezéssel rekurzív módon rendezzük.

Egyesít: Összefésüljük a két rendezett részsorozatot, így kapjuk a rendezett sorozatot.

```
Összefésülő-rendezés(A,p,r)
 If p<r
      Then {q:=[(p+r)/2]
           Összefésülő-rendezés(A,p,q)
           Összefésülő-rendezés(A,q+1,r)
           Összefésül(A,p,q,r)}
```

A teljes tömb rendezését a Összefésülő-rendezés$(A,1,hossz(A))$ kezdőhívás adja meg.

Futási idő elemzés

Legyen $T(n)$ az n darab szám Összefésülő-rendezéssel történő rendezésének legrosszabb eset korlátja.
Ekkor $T(1) = \Theta(1)$ és $T(n) = 2T(n/2) + \Theta(n)$, ha $n > 1$

Állítás $T(n) = O(n \log n)$

Bizonyítás: Teljes indukcióval. Az indukciós lépés:

$$T(n) \leq 2(c \cdot n/2 \log(n/2)) + d \cdot n \leq c \cdot n(\log n - \log 2) + d \cdot n \leq c \cdot n \log n,$$

ha $c \log 2 > d$.

Gyorsrendezés

Oszd meg: Az $A[p,\ldots,r]$ tömböt két nem üres $A[p,\ldots,q]$ és $A[q+1,\ldots,r]$ résztömbre osztjuk úgy, hogy $A[p,\ldots,q]$ minden eleme kisebb vagy egyenlő $A[q+1,\ldots,r]$ minden eleménél.

Uralkodj: Az $A[p,\ldots,q]$ és $A[q+1,\ldots,r]$ résztömböket a gyorsrendezés rekurzív hívásával rendezzük.

Egyesít: A rendezés helyben rendez, így nincs szükség egyesítésre.

```
GYORSRENDEZÉS(A,p,r)
 If p<r
      Then {q=FELOSZT(A,p,r)
            GYORSRENDEZÉS(A,p,q)
            GYORSRENDEZÉS(A,q+1,r)}
```

A teljes tömb rendezését a GYORSRENDEZÉS$(A,1,hossz(A)$ kezdőhívás adja meg.

Feloszt eljárás (Hoare féle)

```
Feloszt(A,p,r)
  x:=A[p]
  i:=p-1
  j:=r+1
  while(Igaz)
       {repeat j:=j-1
              until A[j]<=x
        repeat i:=i+1
              until A[i]>=x
         if i<j
             then csere(A[i],A[j])
             else return j}
```

Példa:

$i[5,3,2,6,1,4,3,7]\,j$ x=5
$[5^i,3,2,6,1,4,3^j,7]$
$[3^i,3,2,6,1,4,5^j,7]$
$[3,3,2,6^i,1,4^j,5,7]$
$[3,3,2,4^i,1,6^j,5,7]$
$[3,3,2,4,1^j,6^i,5,7]$
Return 5

Feloszt eljárás (Lomuto féle)

```
Feloszt(A,p,r)
  x:=A[r]
  i:=p-1
  for j=p to r-1
        {if A[j]<=x
         then {i:=i+1
               csere(A[i],A[j])}}
  csere(A[i+1],A[r])
  return i+1
```

Példa:

$i[2^j,8,7,1,3,5,6,|4]$ x=4
$[2^i|,8^j,7,1,3,5,6,|4]$
$[2^i|,8,7^j,1,3,5,6,|4]$
$[2,1^i|,7,8,3^j,5,6,|4]$
$[2,1,3^i|,8,7,5,6,|4]$
$[2,1,3^i|,4,7,5,6,8]$

Megjegyzés: A Lomuto féle felosztásnál a gyorsrendezést a (A,p,q-1) (A,q+1,r) paraméterekre hívjuk meg.

Futási idő elemzés

Legrosszabb eset

A felosztás mindig egy egyelemű tömbre és a többi elemre oszt. Ekkor

$$T_{lr}(n) = T_{lr}(n-1) + \Theta(n)$$

$$T(n) = \sum_{k=1}^{n} \Theta(k) = \Theta(\sum_{k=1}^{n} k) = \Theta(n^2).$$

Legjobb eset

A felosztás mindig felezi a tömböt. Ekkor

$$T_{lj}(n) = 2T(n/2) + \Theta(n)$$

$$T_{lj}(n) = \Theta(n \log n)$$

Átlagos eset

- Legyen X a eljárás végrehajtása során a FELOSZT által végrehajtott összehasonlítások száma n elemű bemenetre. Ekkor a teljes futási idő $O(n+X)$.
- Legyen z_i az i-edik legkisebb eleme a bemenetnek, és $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$.
- Legyen p_{ij} annak a valószínűsége, hogy z_i-t és z_j-t valamelyik feloszt eljárás összehasonlítja.
- Ekkor $p_{ij} = 2/(j-i+1)$, mivel akkor lesznek összehasonlítva, ha a Z_{ij} halmazból vagy z_i vagy z_j az első felosztó elem.
- Az összehasonlítások várható száma $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2/(j-i+1) = O(n \log n)$. Itt kihasználjuk, hogy a várható érték additív.