1. Alapfogalmak

1.1. Algoritmus

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

1.2. 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 esete: egy olyan (esetleg összetett) adat, amely teljesíti a bemeneti feltételt.
Például, a rendezési probléma a következőt jelenti.
Bemenet: Azonos típusú adatok $H = \{a_1, \ldots, a_n\}$ halmaza, amelyeken értelmezett egy $\leq$ lineáris rendezési reláció.
Kimenet: A H halmaz elemeinek egy $\leq$ rendezéstartó felsorolása, tehát olyan $\langle b_1, \ldots, b_n \rangle$ sorozat, amelyre $b_1 \leq b_2 \leq \ldots \leq b_n$, és $H = \{b_1, \ldots, b_n\}$.

1.3. Specifikáció

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

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

Algoritmus helyessége. 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.

Például, a rendezési probléma specifikációjában a B bemeneti feltétel azt írja elő, hogy a bemeneti halmaz elemein értelmezett a $\leq$ lineáris rendezési reláció. A kimeneti feltétel pedig a következő formulával adható meg:

$$(H = \{b_1, \ldots, b_n\}) \wedge (b_i \leq b_{i+1}, i = 1, \ldots, n-1)$$

Megjegyzés.
1. Ugyanazon bemeneti adatra több olyan kimeneti adat is lehet, amelyre a specifikáció igaz.
2. A kimeneti feltételben hivatkozhatunk bemeneti adatra, illetve bemeneti adatot tartalmazó változóra is. A kimeneti formulában az x változónak a művelet előtti értékére a $Pre(x)$ kifejezéssel hivatkozunk. Pl. a CSERE művelet, amelytől azt várjuk el, hogy az x és y változó tartalmát kicserélje, a következő specifikációval adható meg: $\{Igaz\}$ CSERE $\{x = Pre(y) \wedge y = Pre(x)\}$.

1.4. Algoritmusok futási ideje

Legyen A az $e_1, \cdots, e_m$ 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.

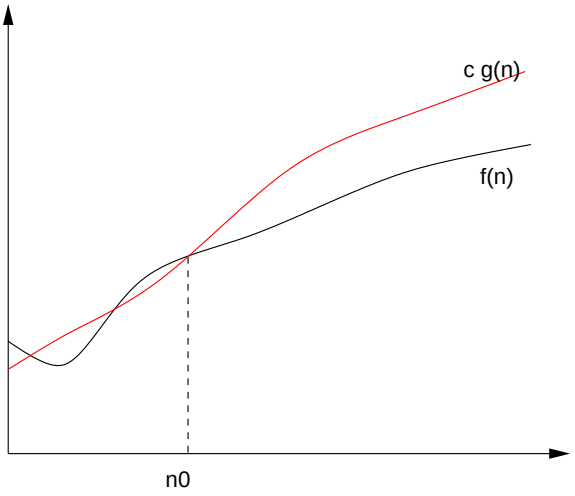
$$T_a(A,n) = \sum_{|x|=n} Pr(x)\,T(A,x)$$

1.5. Függvények növekedési rendje, aszimptotikus jelölések

O-jelölés

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

„$f(n) = O(g(n))$” jelentése: $f(n) \in O(g(n))$

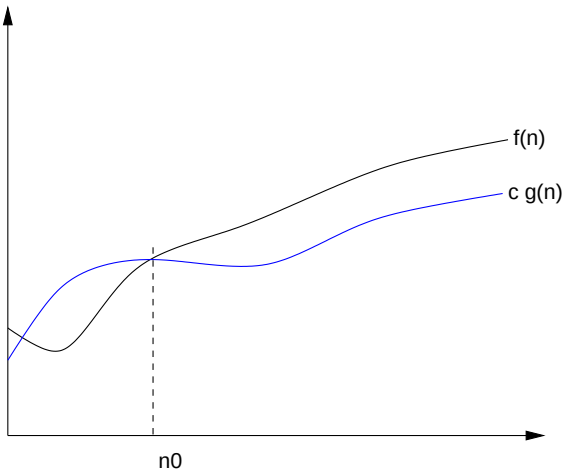


1. ábra. $f(n) = O(g(n))$

Ω-jelölés

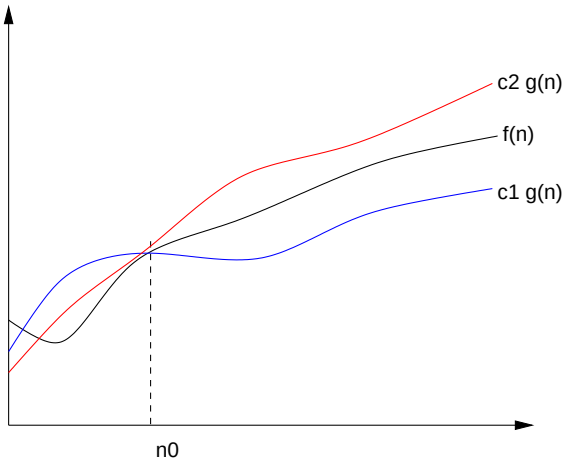
$$\Omega(g(n)) = \{f(n) : (\exists c, n_0 \geq 0)(\forall n \geq n_0)(0 \leq cg(n) \leq f(n))\}$$

Θ-jelölés



2. ábra. $f(n) = \Omega(g(n))$

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



3. ábra. $f(n) = \Theta(g(n))$

Algoritmusok futási idejének elemzésénél előforduló legfontosabb függvényosztályok:
Logaritmikus - $O(\lg n)$
Lineáris - $O(n)$
n-log-n - $O(n \lg n)$
Négyzetes - $O(n^2)$
Köbös - $O(n^3)$
Polinomiális - $O(\sum_{i=0}^{k} a_i n^i)\,(a_k > 0)$
Exponenciális - $O(2^n)$

Minden $f(n)$ és $g(n)$ függvényre $f(n) = \Theta(g(n))$ akkor és csak akkor ha $f(n) = O(g(n))$ és $f(n) = \Omega(g(n))$.
o-jelölés

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

Következmény:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0\ .$$

Például, $2n = o(n^2)$, de $2n^2 \neq o(n^2)$.
ω-jelölés

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

Például, $n^2/2 = \omega(n)$, de $n^2/2 \neq \omega(n^2)$.

Az $f(n) = \omega(g(n))$ relációból következik, hogy

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

1.6. Aszimptotikus egyenlőségek és egyenlőtlenségek

Tranzitivitás:

$$\begin{array}{rcccccl}
f(n) & = & \Theta(g(n)) \text{ és } g(n) & = & \Theta(h(n)) \text{ akkor } f(n) & = & \Theta(h(n)), \\
f(n) & = & O(g(n)) \text{ és } g(n) & = & O(h(n)) \text{ akkor } f(n) & = & O(h(n)), \\
f(n) & = & \Omega(g(n)) \text{ és } g(n) & = & \Omega(h(n)) \text{ akkor } f(n) & = & \Omega(h(n))\ , \\
f(n) & = & o(g(n)) \text{ és } g(n) & = & o(h(n)) \text{ akkor } f(n) & = & o(h(n)), \\
f(n) & = & \omega(g(n)) \text{ és } g(n) & = & \omega(h(n)) \text{ akkor } f(n) & = & \omega(h(n)).
\end{array}$$

Reflexivitás:

$$\begin{aligned} f(n) &= \Theta(f(n)), \\ f(n) &= O(f(n)), \\ f(n) &= \Omega(f(n)). \end{aligned}$$

Szimmetria:

$$f(n) = \Theta(g(n)) \text{ akkor és csak akkor ha } g(n) = \Theta(f(n)).$$

Antiszimmetria:

$$\begin{array}{lcll} f(n) & = & O(g(n)) & \text{akkor és csak akkor ha} \quad g(n) = \Omega(f(n)), \\ f(n) & = & o(g(n)) & \text{akkor és csak akkor ha} \quad g(n) = \omega(f(n)). \end{array}$$

Példa algoritmus futási idejének számítására.
Probléma: Keresés véges halmazból származó, egész számokból álló rendezetlen sorozatban.
Bemenet: $\langle a_0, \ldots, a_{n-1} \rangle$ sorozat és x keresett elem.
Kimenet: Olyan i hogy $0 \leq i < n$ és $x = a_i$ vagy $i = n$ és akkor $(\forall i)(0 \leq i < n \Rightarrow x \neq a_i)$

```
public static int Keres(int[] A, int x){
  int i=0;                          //c1
  while (i<A.length && A[i]!=x){    //c2
    i++;                            //c3
  }
  return i;                         //c4
}
```

Legyen $B_i = \{(A,n,x) : A[i] = x \wedge (\forall j)(0 \leq j < i \Rightarrow A[j] \neq x)\}$ $i = 0, \ldots, n-1$, és
legyen $B_n = \{(A,n,x) : (\forall j)(0 \leq j < n \Rightarrow A[j] \neq x)\}$
Minden (A,n,x) bemeneti adat pontosan egy B_i halmazba esik ($i = 0, \ldots, n$).
Ha $(A,n,x) \in B_i$, akkor KERES(A,n,x) futási ideje

$$c_1 + (i+1)c_2 + ic_3 + c_4$$

Legjobb eset: Ha $(A,n,x) \in B_0$, tehát
$T_{lj}(n) = c_1 + c_2 + c_4 = O(1)$
Legrosszabb eset: Ha $(A,n,x) \in B_n$, tehát
$T_{lr}(n) = c_1 + (n+1)c_2 + nc_3 + c_4 = (c_2 + c_3)n + c_1 + c_2 + c_4 = O(n)$. **Átlagos eset:**
Legyen D az Integer típus elemeinek a száma. Tegyük fel, hogy minden bemenet egyformán valószínű, azaz annak valószínűsége, hogy $x = a$ valamely adott a egész számra $\frac{1}{D}$.
Vezessük be a $p = \frac{1}{D}$ és $q = 1 - \frac{1}{D}$ jelöléseket (Nyilvánvaló, hogy $p < 1$ és $q < 1$).
Jelölje $Pr((A,n,x) \in B_i)$ annak valószínűségét, hogy az (A,n,x) bemenet a B_i halmazba esik.
$Pr((A,n,x) \in B_i) = q^i p$, $i = 0, \ldots, n-1$, és
$Pr((A,n,x) \in B_n) = q^n$
Tehát a KERES algoritmus átlagos futási ideje:

$$\begin{aligned} T_a(n) &= \sum_{i=0}^{n} Pr((A,n,x) \in B_i)(c_1 + (i+1)c_2 + ic_3 + c_4) \\ &= \sum_{i=0}^{n-1} q^i p(c_1 + (i+1)c_2 + ic_3 + c_4) + q^n(c_1 + (n+1)c_2 + nc_3 + c_4) \\ &\leq (c_1 + n(c_2 + c_3) + c_4)p \sum_{i=0}^{n-1} q^i + (c_1 + (n+1)c_2 + nc_3 + c_4) \\ &= (c_1 + n(c_2 + c_3) + c_4)p\frac{q^n - 1}{q - 1} + (c_1 + (n+1)c_2 + nc_3 + c_4) \\ &= (c_1 + n(c_2 + c_3) + c_4)(1 - q^n) + (c_1 + (n+1)c_2 + nc_3 + c_4) \\ &\leq (c_1 + n(c_2 + c_3) + c_4) + (c_1 + (n+1)c_2 + nc_3 + c_4) \\ &= (2c_2 + 2c_3)n + 2c_1 + c_2 + 2c_4 \end{aligned}$$

Tehát $T_a(n) = O(n)$.

Ha algoritmus futási idejének nagyságrendjét (O, Ω, Θ) akarjuk kifejezni, akkor a számítást egyszerűsíthetjük:

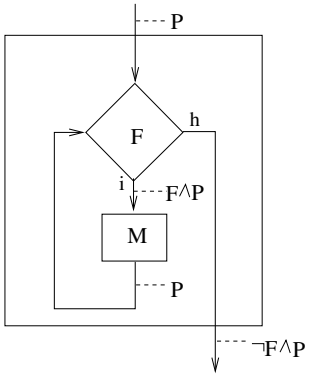
1. Minden elemi művelet költségét 1-nek vehetjük.
2. Elhagyhatjuk azokat az elemi műveleteket, amelyek végrehajtási száma konstans, azaz nem függ a bemeneti adattól.

1.7. Bizonyítási (következtetési) szabályok

$$\frac{H_1, \cdots, H_k}{H}$$

Ha $H_1, \cdots, H_k$ igaz állítások (specifikációk), azaz vagy axiómák, vagy már bizonyított állítások, akkor H is igaz állítás.

1.7.1. Kezdőfeltételes ismétléses vezérlés.

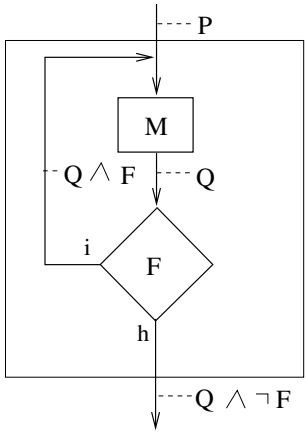


4. ábra. while ciklus bizonyítási szabálya

$$\frac{\{F \wedge P\}\, M\, \{P\}}{\{P\}\, \textbf{while}\, (F)\, M\, \{\neg F \wedge P\}}$$

A P logikai formulát *ciklusinvariánsnak* nevezzük.

1.7.2. Végfeltételes ismétléses vezérlés.



5. ábra. repeat ciklus bizonyítási szabálya

$$\frac{\{P\}\, M\, \{Q\},\ Q \wedge F \Rightarrow P}{\{P\}\, \textbf{do}\, M; \textbf{while}\, (F)\, \{Q \wedge \neg F\}}$$

1.8. A KERES algoritmus helyességének bizonyítása

```
public static int Keres(int[] A, int x){
  int i=0;                          //c1
  while (i<A.length && A[i]!=x){    //c2
    i++;                            //c3
  }
  return i;                         //c4
}
```

Jelölje F az ismétlési feltételt, $F = (i < n) \wedge (A[i] \neq x)$.
1. Megmutatjuk, hogy a $P = (\forall j)(1 \leq j < i \Rightarrow A[i] \neq x)$ formula ciklusinvariáns lesz.
Nyilvánvaló, hogy ha a while ciklus magja, az i++ értékadás előtt teljesül az $F \wedge P$ formula, akkor az értékadás után teljesül P.
2. Meg kell mutatni, hogy a while ciklus előtt, azaz az i=0; értékadás után teljesül a P formula. Ez azért igaz, mert nincs olyan j, hogy $0 \leq j < i = 0$.
A while ciklus biztosan terminál, mert a ciklusmag minden végrehajtása 1-el növeli az i változó értékét. A kezdőfeltételes ismétlés bizonyítási szabálya szerint a while ciklus után $\neg F \wedge P$ teljesül.
$\neg F = (i \geq n) \vee (A[i] = x)$. Ha $i \geq n$ teljesül, az azt jelenti, hogy a keresett x nem eleme a sorozatnak, hisz P is teljesül. Ha a terminálás után $i < n$, akkor $A[i] = x$ és mivel P is teljesül, i a legkisebb olyan index, hogy $A[i] = x$.