

Algo jegyzet

Princzes Barnabás

2023. március 21.

Kivonat

Lent hagytam a füzetem és jövő héten doga szóval time to kick ass! Írjunk egy jó kis jegyzetet.

1. Előadás

1.1. Alapok

Van a $\{B\}A\{K\}$ azaz a Bemeneti feltétel (egy logikai állítás) Algoritmus és Kimeneti feltétel (szintén egy logikai állítás).

Legyen A algoritmus. e_1, \dots, e_m elemi műveletek az algoritmusban és t_i az adott e_i -hez tartozó időigény. Az algoritmus tényleges futási ideje $T(A, x)$ ahol x egy bemenet és a bemenet mérete $|x|$ például (tömb, halmaz...) esetében az elemek száma. Az e_i és t_i és $|x|$ együtt a bonyolultság mértéke.

1.2. Esetek

- Legjobb eset:

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

- Legrosszabb eset:

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

- Átlagos eset: Legyen $\Pr(x)$ annak a valószínűsége, hogy épp x lesz A algoritmus bemenete, ekkor:

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

```
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
}
```

Itt $\text{Keres}(A, n, x)$ futási ideje: $c_1 + (i + 1)c_2 + ic_3 + c_4$

$$T_{lj} = c_1 + c_2 + c_4 = O(1)$$

$$T_{lr} = c_1 + (n+1)c_2 + nc_3 + c_4 = (c_2 + c_3)n + c_1 + c_2 + c_4 = O(n)$$

Az átlagos eset számításához vezessünk be pár fogalmat:

Legyen B_0 a legjobb bemeneti eset amikor is egyből megtaláljuk a keresett elemet

Legyen B_n amikor nem találjuk meg.

Minden lehetséges bemenet $\in B_i \mid i = 0, \dots, n-1$

Legyen D az Integer összes lehetséges értéke (nagy szám).

Annak hogy az keresendő számot pont kiszűrjük a valószínűsége: $p = \frac{1}{D}$

Annak, hogy egy olyat találunk amelyik nem a keresett szám: $q = 1 - \frac{1}{D}$

$Pr((A, n, x) \in B_i) = q^i p, i = 0, \dots, n-1$

Itt q^i jelöli, hogy hányszor nem találtuk meg és p amikor megtaláltuk. Az utolsó lehetséges esetnél viszont csak q eset fordul elő az pedig n -szer ugyanis végigmegyünk és nem találjuk meg a keresett elemet. $Pr((A, n, x) \in B_n) = q^n$,

$$T_a(n) = \sum_{i=0}^n Pr((A, n, x) \in B_i)(c_1 + (i+1)c_2 + ic_3 + c_4)$$

Az átlagos esetet úgy kapjuk meg, ha minden lehetséges bemenet futási idejét megszorozzuk az adott bemenet valószínűségével és az eseteket összeadjuk.

A B_n bemenetre vonatkozó eset specifikus így emeljük ki a többi közül:

$$T_a(n) = q^n(c_1 + (n+1)c_2 + nc_3 + c_4) + \sum_{i=0}^{n-1} q^i p(c_1 + (i+1)c_2 + ic_3 + c_4)$$

Mivel $q^n \rightarrow 0$ (0-hoz tart) ezért felfele kerekítjük 1-re. Mivel a Σ alatt mindent beszorzunk $p(\dots)$ -al ahol a zárójelben $i < n$ így azt ki is emelhetjük ugyanúgy felfele kerekítve i -t és $i+1$ -et kicserélve n -re.

$$T_a(n) \lesssim T_a(\hat{n}) = (c_1 + (i+1)c_2 + ic_3 + c_4) + (c_1 + n(c_2 + c_3) + c_4)p \sum_{i=0}^{n-1} q^i$$

A Σ tag egyenlő a mértani sorozat összegképletével, behelyettesítjük:

$$T_a(\hat{n}) = (c_1 + (i+1)c_2 + ic_3 + c_4) + (c_1 + n(c_2 + c_3) + c_4)p \frac{q^n - 1}{q - 1}$$

Mivel $p = 1 - q$ ezért átalakítunk:

$$T_a(\hat{n}) = (c_1 + (i+1)c_2 + ic_3 + c_4) + (c_1 + n(c_2 + c_3) + c_4)(1 - q^n)$$

Mivel $(1 - q^n) \rightarrow 1$ megint felfele kerekítünk, majd összevonunk.

$$T_a(\hat{n}) \lesssim T_a(\hat{\hat{n}}) = (2c_2 + 2c_3)n + 2c_1 + c_2 + 2c_4$$

Itt pedig n a domináns tag szóval:

$$T_a(n) = O(n)$$

1.3. Praktikus képletek

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

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

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

$$\sum_{i=1}^n i = \frac{n(n+1)}{2} \quad (a_1 = 1, d = 1 \text{ speciális számtani sorozat összegképlete})$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^n n = n^2$$

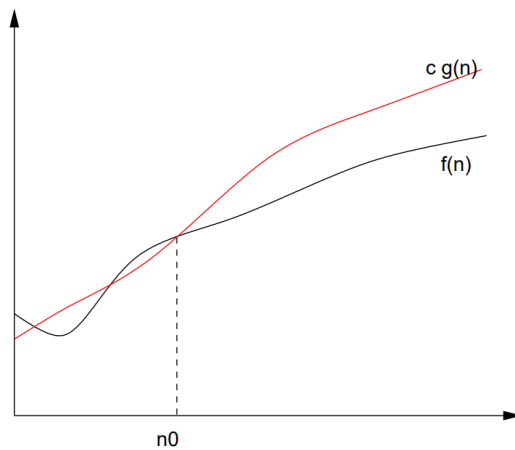
$$\sum_{i=m}^n q^i = \frac{q^m - q^{n+1}}{1 - q} \quad (\text{mértani sorozat összegképlete})$$

2. Függvények növekedési rendje

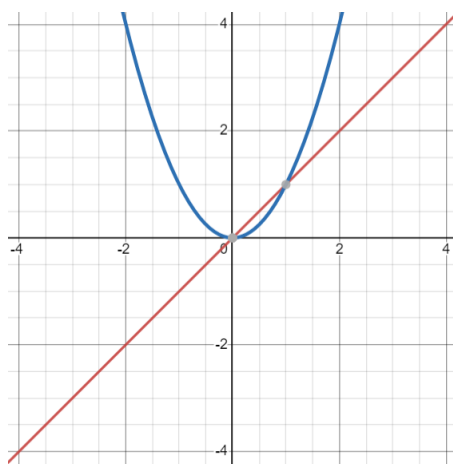
2.1. O - Nagy ordó

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

Nagy ordó egy halmaz amelyekben adott függvények vannak, jelöljünk egy ilyen függvényt $f(n)$ -el. Ezek minden olyan $f(n)$ függvény ahol létezik egy c konstans úgy, hogy lesz egy n_0 küszöbszám amire igaz, hogy minden n_0 -nál nagyobb n -számra $f(n)$ nem negatív és $cg(n)$ alatt van. $f(n) \in O(g(n))$ helyett szoktuk a $f(n) = O(g(n))$ jelölést alkalmazni.



$f(n) = O(g(n))$ mivel $g(n)$ -re tudunk választani olyan pozitív konstansot, hogy mindig $f(n)$ fölött maradjon.



Hasonlítsuk össze x és x^2 függvényt

Nem tudunk olyan c konstans választani, hogy azzal $c \cdot x$ egy n_0 érték fölött mindig nagyobb legyen mint x^2 ezért:

$$O(x) \neq O(x^2)$$

2.2. Ω - Omega

Ugyanaz mint a O csak ez alulról korlátoz.

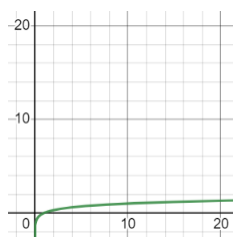
$$\Omega(g(n)) = \{f(n) : (\exists c > 0, n_0 \geq 0)(\forall n \geq n_0)(0 \leq cg(n) \leq f(n))\}$$

2.3. Θ - Theta

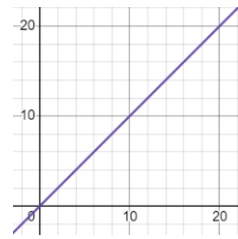
Ez alulról és felülről is korlátoz. Azaz egy adott értéktől a legjobb és a legrosszabb eset is ugyanazt a nagyságrendet követi.

$$\Theta(g(n)) = \{f(n) : (\exists c_1, c_2 > 0, n_0 \geq 0)(\forall n \geq n_0)(0 \leq c_1g(n) \leq f(n) \leq c_2g(n))\}$$

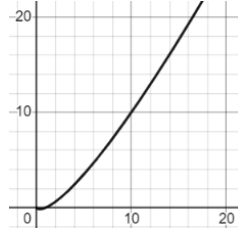
2.4. Nevezetes függvényosztályok



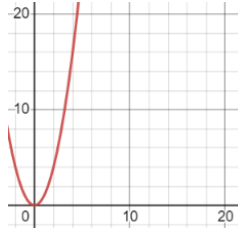
Logaritmikus - $O(\lg(n))$



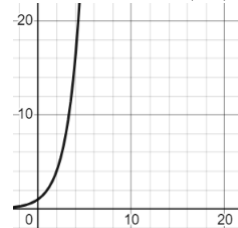
Lineáris - $O(n)$



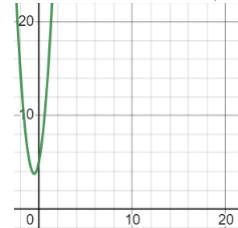
$n \log(n)$ - $O(n \lg(n))$



Négyzetes - $O(n^2)$



Exponenciális - $O(2^x)$



Polinomiális - $O(\sum_{i=0}^k a_i n^i) (a_k > 0)$

Ez utóbbi a legösszetettebb így itt sokkal elővigyázatosabbnak kell lenni mikor azt nézzük, hogy melyiknél nagyobb.

3. Alkalmazás

A fenti definíciók következményei:

- Ha $f(n) = O(g(n))$ azaz $f(n)$ -t felülről határolja $g(n)$ akkor:

$$0 \leq \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

- Ha $f(n) = \Omega(g(n))$ azaz $f(n)$ -t alulról határolja $g(n)$ akkor:

$$0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq \infty$$

- Ha $f(n) = \Theta(g(n))$ azaz $f(n)$ -t alulról és felülről is határolja $g(n)$, azaz növekedési rendjük egyenesen arányos, akkor:

$$0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

3.1. Praktikus képletek

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

4. Rekurzió

Egy P függvény rekurzív, ha utasításai közt előfordul P hívása.

Egy rekurzív összefüggés önmagának hívásából és egy leállási feltételből áll. A leállási feltétel a degenerált esetre vezet.

4.1. Partíciószám probléma: $P(n) = ?$

Legyen n egy szám ennek egy partíciója az a π ahol $\pi = \langle a_1, \dots, a_k \rangle$ olyan a_i számokból álló sorozat amelyekre teljesül:

$$\sum_{i=1}^k a_i = n$$
$$a_1 \geq a_2 \geq \dots \geq a_k > 0$$

Ezek teljesülésekor π nem más mint n -nek 1 partíciója. Jelöljük $P(n)$ -el n szám összes partíciójának számát.

Mondjuk meg n szám partícióinak számát!

- Vezessünk be egy új jelölést:
Legyen $P2(n, k)$ n -szám azon partíciójainak száma ahol $a_i \leq k$.
Ezzel a jelöléssel $P2(n, n) = P(n)$.

- Ez azért segít nekünk mert így fel tudunk írni egy rekurzív feltételt:

$$P2(n, n) = P2(n, n - 1) + 1$$

Példa:

$$P2(4, 3) = |\{ \{3, 1\}, \{2, 2\}, \{2, 1, 1\}, \{1, 1, 1, 1\} \}| = 4$$

$$P2(4, 4) = |\{ \{4\}, \{3, 1\}, \{2, 2\}, \{2, 1, 1\}, \{1, 1, 1, 1\} \}| = 5$$

Egy esettel több mert abban az esetben ha a_i tartalmazza n -et akkor a partíció fogalma miatt nem tartalmazhat mást.

- Egy rekurzív összefüggésnek viszont szükséges meghatározunk a degenerált esetet/eseteket:

$$P2(n, 1) = P2(1, k) = 1$$

Példa:

$$P2(4, 1) = |\{ \{1, 1, 1, 1\} \}| = 1$$

$$P2(1, 4) = |\{ \{1\} \}| = 1$$

A 4-et egyesek összegéből 1 féle képpen tudjuk kirakni. Az 1-et megpróbálhatjuk bármilyen nagy számokból kirakni, de csak egy darab 1-esből tudjuk.

- A fenti példából következik, hogy nem csak az egyest hanem a többi számot is feleslegesen próbálnánk meg nálánál nagyobból kirakni:

$$P2(n, k) = P2(n, n), \text{ ahol } n < k$$

Példa:

$$P2(4, 5) = |\{ \{4\}, \{3, 1\}, \{2, 2\}, \{2, 1, 1\}, \{1, 1, 1, 1\} \}| = 5$$

Nem tudunk 5 -öst használni mert a definíció gátolja.

- Mivel rájöttünk, hogy n generálja a speciális eseteket elgondolkodhatunk miben más $P2(n, k)$ és $P2(n, k - 1)$:

$$P2(n, k) = P2(n, k - 1) + P2(n - k, k), \text{ ahol } k < n$$

Példa:

$$P2(5, 3) = |\{ \{3, 2\}, \{3, 1, 1\}, \{2, 2\}, \{2, 1, 1\}, \{1, 1, 1, 1\} \}| = 5$$

$$P2(5, 2) = |\{ \{2\}, \{1, 1\} \}| = 2$$

$$P2(2, 3) = |\{ \{2, 2\}, \{2, 1, 1\}, \{1, 1, 1, 1\} \}| = 3$$

Az első partícióban fellelhető az utolsó partíció összes eleme, ezentúl k mellé azért, hogy n legyen az összeg a második partícióval előállítható eseteket használhatjuk fel.

Felírtunk 4 szabályt:

$$P2(n, k) = P2(n, k - 1) + P2(n - k, k), \text{ ahol } k < n$$

$$P2(n, k) = P2(n, n), \text{ ahol } n < k$$

$$P2(n, n) = P2(n, n - 1) + 1$$

$$P2(n, 1) = P2(1, k) = 1$$

Ezek alapján egy programmal így számolnánk:

```

public class Particio{
    public static long P(int n){
        return P2(n,n);
    }
    private static long P2(int n, int k){
        if (k==1 || n==1)
            return 1;
        if (k>=n)
            return P2(n,n-1)+1;
        return P2(n, k-1)+P2(n-k, k);
    }
}

```

Viszont nekünk egyszerűbb ha egy táblázatot csinálunk. A táblázat sorindexe legyen k , oszlop-indexe n , cellái pedig $P(n, k)$:

... k							
6							
5							
4							
3							
2							
1							
	1	2	3	4	5	6	... n

Először töltsük ki ahol $n = 1$ vagy $k = 1$ ugyanis $P2(n, 1) = P2(1, k) = 1$:

... k	1						
6	1						
5	1						
4	1						
3	1						
2	1						
1	1	1	1	1	1	1	
	1	2	3	4	5	6	... n

Mivel $P2(n, n) = P2(n, n-1) + 1$ így $P(2, 2)$ -öt ki tudjuk tölteni úgy, hogy az alatta lévő számhoz hozzáadunk egyet:

... k	1						
6	1						
5	1						
4	1						
3	1						
2	1	2					
1	1	1	1	1	1	1	
	1	2	3	4	5	6	... n

Innen pedig 2 lehetőség áll fent, kezdjük azzal amikor $n < k$. Erre a szabályunk, hogy $P2(n, k) = P2(n, n)$, ahol $n < k$:

$\dots k$							
6	1	2					
5	1	2					
4	1	2					
3	1	2					
2	1	2					
1	1	1	1	1	1	1	
<hr/>							
	1	2	3	4	5	6	$\dots n$

Amikor $n > k$ akkor is megvan a szabályunk:

$$P2(n, k) = P2(n, k-1) + P2(n-k, k), \text{ ahol } k < n$$

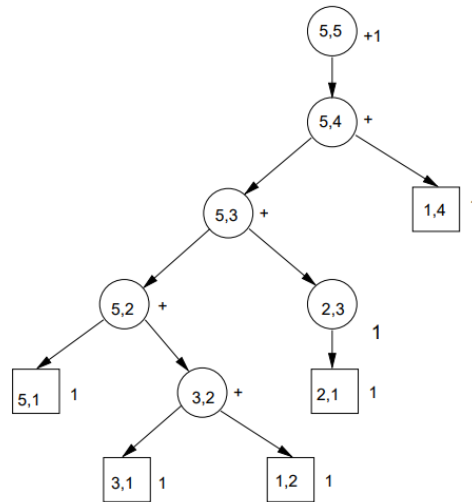
Ez a kitöltés során úgy fog kinézni, hogy (n, n) -től jobbra indulva összeadjuk a kitöltendő cella alatt lévő számot és azt a számot ami tőle balra van k -lépésre:

$\dots k$							
6	1	2					
5	1	2					
4	1	2					
3	1	2					
2	1	2	2	3	3	4	
1	1	1	1	1	1	1	
<hr/>							
	1	2	3	4	5	6	$\dots n$

Majd a fenti lépéseket ismételve folytatjuk:

$\dots k$							
6	1	2	3	5	7	11	
5	1	2	3	5	7	10	
4	1	2	3	5	6	9	
3	1	2	3	4	5	7	
2	1	2	2	3	3	4	
1	1	1	1	1	1	1	
<hr/>							
	1	2	3	4	5	6	$\dots n$

Rekurziós fa fogalma. 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.



1. ábra. A $P2(5,5)$ eljáráshívás rekurziós fája

5. Struktúrák, nagyrészt mutatókkal

5.1. halmazok

Matekban sokszor használjuk a halmazokat. Programozásban viszont az algoritmusoknak kell tudni módosítaniuk a halmazokat, az ilyen módosítható halmazokat dinamikus halmazoknak mondjuk.

Egy halmazban az adat általában ilyen tulajdonságokat tárol:

- kulcs - a manipulációhoz és azonosításhoz
- Satellite mezők - olyan adattagok amik nem érdekesek a halmaz szempontjából

A halmazon lehet módosítani:

- $INSERT(S, x)$
- $DELETE(S, x)$

Továbbá a halmaz adatait is le lehet kérni:

- $SEARCH(S, k)$ - $k = key[x]$ vagy ha nincs a keresett kulccsal elem a halmazban akkor NIL -t ad vissza.
- $MINIMUM(S)$ - Sorba rendezi az elemeket kulcs szerint és visszadobja a legkisebbet.
- $MAXIMUM(S)$ - Rendez és a legnagyobb kulccsal rendelkezőt dobja vissza.
- $SUCCESSOR(S, x)$ - x ekkor a rendezett S halmaz egy eleme