

I. Gyakorlat (2019.02.11)

1. Polinom¹ helyettesítési értékének kiszámítása. Adott egy n -ed fokú polinom, határozzuk meg egy adott x helyen felvett értékét:

$$a_n * x^n + a_{n-1} * x^{n-1} + \dots + a_1 * x + a_0$$

Tegyük fel, hogy nagyon sok polinomunk van és nagyon sok helyen kell kiszámítani az értékét, ezért törekedjünk arra, hogy minél hatékonyabb megoldást készítsünk.

A polinom együtthatóit egy 0-tól indexelt, $n+1$ méretű tömbben helyezzük el. (Megállapodás: ha a tömb neve „Z” vagy „Z”-re végződik, akkor 0-tól indexelünk.) A tömb mérete $Z.M = n+1$.

A megoldásoknál írjuk fel, hogy az egyes lépések hányszor hajtódnak végre. Vizsgáljuk meg a szorzások $S(n)$ és az összeadások $\ddot{O}(n)$ számát, a polinom fokszámának függvényében.

Feltehető, hogy az $n \geq 0$, azaz $Z.M > 0$

Első megoldás, az összegzés tételéből származik:

Polinom1(Z:R[]; x:R) :R	<i>Hányszor fut le</i>
y := Z[0]	1
i = 1 to Z.M-1	n+1
h := x	n
j = 1 to i-1	1+2+3+...+n-1+n
h := h*x	1+2+3+...+n-1
y := y+h*Z[i]	n
return y	1

$$S(n) = \frac{n * (n + 1)}{2} = \frac{n^2 + n}{2}$$

$$\ddot{O}(n) = n$$

¹ – olyan többtagú algebrai kifejezés, melyben csak számok és változók nem negatív egész kitevőjű hatványainak szorzata illetve összegei szerepelnek (Pl.: $q(x) = 2x^2 + 6x + 9$)

Második megoldás, x hatványait rekurzívan számoljuk a h változóban:
 $x^i = x^{i-1} * x$, ha $i > 0$, $x^0 = 1$

Rekurzív(Z:R[]; x:R) :R	<i>Hányszor fut le</i>
y:=Z[0] h:=1	1
i = 1 to Z.M-1	n+1
h:=h*x	n
y:=y+h*Z[i]	n
return y	1

$$S(n) = 2n$$

$$\ddot{O}(n) = n$$

Harmadik megoldás, a Horner séma:

$$y = (((\dots(a_n * x + a_{n-1}) * x + a_{n-2}) * x + \dots + a_1) * x + a_0$$

Horner(Z:R[]; x:R) :R	<i>Hányszor fut le</i>
y:=Z[Z.M-1]	1
i= Z.M-2 downto 0	n+1
y:=y*x+Z[i]	n
return y	1

$$S(n) = n$$

$$\ddot{O}(n) = n$$

A három megoldást szemléltetjük egy táblázatban a Θ aszimptotikus korlát segítségével. (Aszimptotikánál a konstans tagok és az alacsonyabb kitevőjű tagok elhagyhatók, ezért fog eltűnni a rekurzív megoldásnál a szorzatból a 2-es.) $T(n)$ a futási időt jelöli.

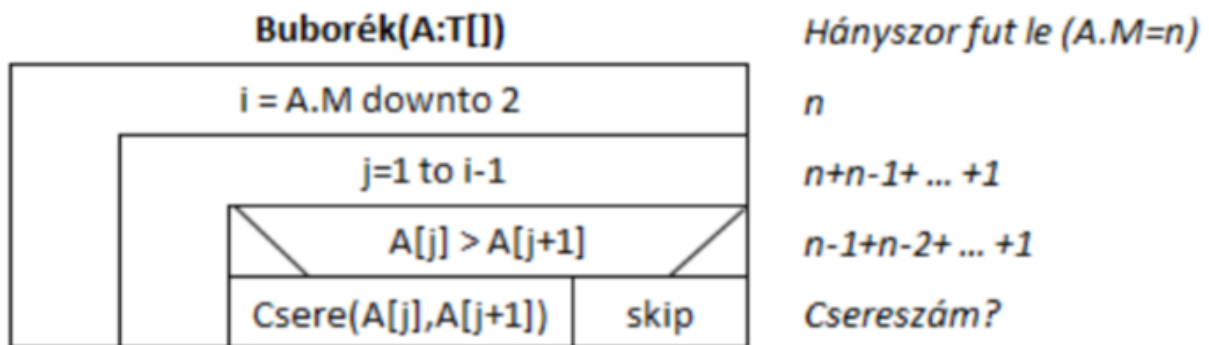
	Polinom1	Rekurzív	Horner
$S(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$
$\ddot{O}(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
$T(n)$	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$

2. **Buborék rendezés** (Elméletileg ez kellett, hogy legyen programozáson.)
 Készítsük el az alap algoritmust, majd a javított változatot (ez utóbbi szorgalmi HF). Elemezzük itt is, hogy a struktorgram egyes lépései hányszor hajtódnak végre. Nézzük meg az összehasonlítások $\bar{O}h(n)$ és cserék $Cs(n)$ számát. Cserék elemzésénél használjuk az $mCs(n)$ (minimális cserék száma), $MCs(n)$ (maximális cserék száma) illetve az $ACs(n)$ (átlagos csere szám) jelöléseket. Átlagos csere számot nem kell pontosan kiszámolni, elég csak a „megérzés”-re támaszkodni.

Buborék példa:					Csere
3	5	2	4	1	0
3	5	2	4	1	1
3	2	5	4	1	1
3	2	4	5	1	1
3	2	4	1	5	1. menet vége, 5 a helyén van
3	2	4	1	5	1
2	3	4	1	5	0
2	3	4	1	5	1
2	3	1	4	5	2. menet vége
2	3	1	4	5	0
2	3	1	4	5	1
2	1	3	4	5	3. menet vége
2	1	3	4	5	1
1	2	3	4	5	4. menet vége, rendezett a tömb

Csere összesen: 7
 Összehasonlítás összesen: 10

A rendező kulcsokat (és a hozzájuk tartozó adatokat) egy A nevű tömbben helyeztük el. $A.M = n$, a rendező kulcsok darabszáma.



Összehasonlítások száma $\ddot{O}(n) = \sum_{i=1}^{n-1} i = \frac{n*(n-1)}{2} = \frac{n^2-n}{2} = \Theta(n^2)$

Cserék számát hogyan tudjuk meghatározni?

A cserék száma a rendezendő adatsorban található inverziók számával egyenlő. Lásd a példában 7 inverzió van: 3,2 3,1 5,2 5,4 2,1 4,1

Ebből adódik, hogy $mCs(n) = 0$ (nincs inverzió, azaz növekvően rendezett a bemenet). $MCs(n) = \ddot{O}(n)$ (minden összehasonlítást csere követ, azaz fordítottan rendezett a tömb). $ACs(n) = \frac{n*(n-1)}{4} = \Theta(n^2)$. Ezt nem kell pontosan levezetni az alábbi linken megtalálható dr. Fekete István jegyzetében

https://people.inf.elte.hu/fekete/algorithmusok_jegyzet/01_fejezet_Muveletigeny.pdf

Aszimptotika – egy görbének (lehet az függvény görbe is) végtelenbe vesző vége tetszőlegesen megközelít egy egyenest, de azt soha el nem éri, akkor a görbének ez a vége aszimptotikus tulajdonságú és az egyenes, amihez közelít a görbe aszimptotája.

Ω = aszimptotikus alsó korlát – adott $f(n)$ függvényhez, akkor mondjuk, hogy $\Omega(g(n)) = f(n)$, ha létezik c, n_0 állandó, hogy $n > 0$ esetén $0 \leq cg(n) \leq f(n)$

O = aszimptotikus alsó korlát - adott $f(n)$ függvényhez, akkor mondjuk, hogy $O(g(n)) = f(n)$, ha létezik c, n_0 állandó, hogy $n > 0$ esetén $0 \leq f(n) \leq cg(n)$

Használjuk Ω és O a csere számokra: $mCs(n) = 0$, $MCs(n) = \Theta(n^2)$, azaz $mCs(n) = \Omega(1)$, $MCs(n) = O(n^2)$

Buborék rendezés javítási módszerei:

- figyelhetjük egy logikai változóval, hogy volt-e csere, ha nem volt akkor a külső ciklus álljon le
- megjegyezhetjük az utolsó csere helyét, ha ez u és $u+1$ indexen történt, akkor $u+1$ -től már a tömb rendezett, a külső ciklus változót u -ra lehet csökkenteni. Legkedvezőbb és legrosszabb esetek: $m\ddot{O}(n) = \Theta(n)$, $M\ddot{O}(n) = \Theta(n^2)$. Futási idő: $mT(n) = \Theta(n)$, $MT(n) = \Theta(n^2)$.

Példa:

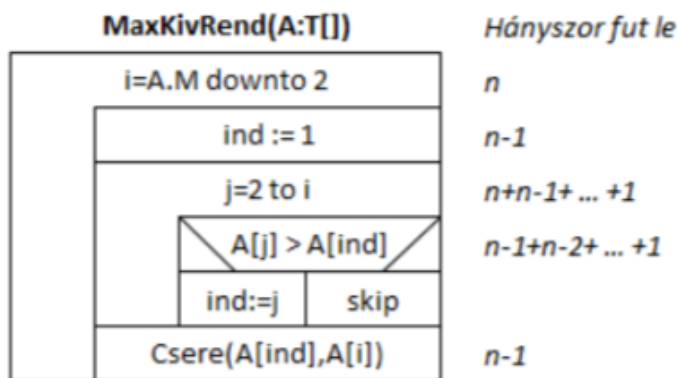
Javított buborék példa:					Csere	
2	3	1	4	5	0	
2	3	1	4	5	1	u=2
2	1	3	4	5	0	
2	1	3	4	5	0	
2	1	3	4	5		1. menet vége 3,4,5 rendezett
2	1	3	4	5	1	u=1
1	2	3	4	5		kész

Csere összesen: 2
 Összehaonlítás összesen: 5

Struktogramja (házi feladat, később bekerül)

3. Maximum kiválasztásos rendezés.

Maximum értéket azért nem használunk, mert rendezésnél mindig feltesszük, hogy nem csak a kulcs, hanem a kulcshoz tartozó rekord is tárolva van a tömbben, melynek mozgatása költséges lenne.



Házi feladatok

1. Legendre algoritmus. Az algoritmus az a^k hatványát számolja ki.

Struktogram házi feladat.

Útmutatás:

Érdemes lejátszani egy példán

Menet	a	s	k
Kezdés	3	1	11
1. k páratlan	3	3	10
2. k páros	3^2	3	5
3. k páratlan	3^2	3^3	4
4. k páros	3^4	3^3	2
5. k páros	3^8	3^3	1
6. k páratlan	3^8	3^{11}	0
k=0, vége az algoritmusnak, s –ben a megfelelő hatvány van			

- legkedvezőbb eset, amikor a k 2 hatványa, ekkor a bináris alakja: 100..00, azaz a legutolsó menet kivételével mindig páros ágon fut az algoritmus, k mindig feleződik. A legutolsó menetben $k=1$ esetében egyszer fut le a páratlan ág. Ilyenkor a ciklus meneteinek száma: $T(k) = \log_2 k + 1$.
- legkedvezőtlenebb az az eset, amikor felváltva fut a páratlan, majd páros ágon. Ilyenkor 2 hatvány-1 alakú a k , aminek bináris alakja csupa 1-es: 111..11. Amikor a „ $k := k-1$ ” ágon fut páros lesz a szám, majd kettővel osztva ismét páratlan, ilyenkor a ciklus meneteinek száma: $T(k) = 2 \cdot \lceil \log_2 k \rceil + 1$. (ha mindig osztunk 2-vel $\rightarrow \log_2$)
- Mivel ugyanazt a nagyságrendet kaptuk, azaz $mT(k) = MT(k) = \Theta(\log_2 k) = \Theta(\log k)$, így az általános eset is e kettő közé kell, hogy essen.
- egy kis kitérő: $\log_a n = \Theta \log_b n$ és fordítva, azaz ha $a, b > 1$, akkor a logaritmusok ugyanazt a nagyságrendet képviselik, így a logaritmus alapszáma elhanyagolható
- a $k := k/2$ nem jelent osztást, ez csak a bitek shiftelését jobbra, így ez a Horner sémához hasonlóan egy nagyon hatékony algoritmus az egész kitevőjű hatványok kiszámításához.

- Adott egy n hosszú, egész számokat tartalmazó tömb. Keressük a tömb azon szakaszát, melynek összege a lehető legnagyobb. Legyen a tömb neve A és adjuk meg a két indexet: $1 \leq ind1 \leq ind2 \leq n$, melyre $\sum_{i=ind1}^{ind2} A[i]$ a maximális. 3 megoldás létezik: $\Theta(n^3)$, $\Theta(n^2)$, $\Theta(n)$.

Egy adott k kezdőindextől (kezdetben 1-től) elkezdjük összeadni a tömbben lévő számokat. Mindig a következő (i -edik) elemmel növeljük az összeget. Ha az így kapott összeg nagyobb, mint az $A[i]$, akkor megyünk tovább az összeg számolással. Amikor az összeg negatívvá válik (lásd példában 4. elem), akkor a következő körben $s + A[i] < A[i]$ teljesül, így nem érdemes a részösszeget folytatni, egy új részösszeget kezdünk s -ben, és megjegyezzük az új kezdőindexet k -ban. Ha az adott körben s értéke nagyobb lesz, mint az

eddig részösszeg maximuma, akkor max-ot, és ind1, ind2 változókat is megfelelően átállítjuk.

Tulajdonképpen ez egy rekurzív függvényen végzett maximum keresés², ahol a rekurzív függvény értéke két komponensből áll, egy összegből (s), és egy kezdőindexből (k). A rekurzív függvény egyenlete pedig:

$$szum(1) = (A[1], 1)$$

$$i > \text{esetben: } szum(i) = \begin{cases} (A[i], i), & \text{ha } A[i] > szum(i-1)_1 + A[i] \\ (szum(i-1)_1 + A[i], szum(i-1)_2), & \text{egyébként} \end{cases}$$

A ciklusmag elsőként meghatározza a rekurzív függvény i-edik értékét, majd a maximum kiválasztás tétel ezeken keresi a maximumot.

Illusztráció:

A	1	2	3	4	5	6	7	8	9	10
	4	5	-3	-7	4	-1	2	3	-1	10
s	4	9	6	-1	4	3	5	8	7	17
k	1				5					
max	4	9								17
ind1	1	1								5
ind2	1	2								10