1. gyakorlat

Téma:

Algoritmusok műveletigényének meghatározása, hatékonyság, hatékonyság jellemzése (aszimptotikus korlátok bevezetése).

Javasolt feladatok:

1. **Polinom helyettesítési értékének kiszámítása**. Adott egy n-ed fokú polinom, határozzuk meg egy adott x helyen felvett értékét: $a_n*x^n+a_{n-1}*x^{n-1}+ \ldots. a_1*x+a_0$
   (Tfh nagyon sok polinomunk van, és nagyon sok helyen kell kiszámítani az értékét, ezért készítsünk minél hatékonyabb megoldást.)

A polinom együtthatóit egy nullától indexelt, n+1 méretű tömbben helyezzük el. (Megállapodás: ha a tömböt nem nullától indexeljük, a deklarációnál és a specifikációnál jelezzük, pl. A/1:T[n]. Most tehát Z:R[] ugyanaz, mint Z/0:R[].) A Z tömb mérete: Z.length (hangsúlyozzuk, hogy: Z.length=n+1).

A megoldásoknál írjuk fel, hogy az egyes lépések hányszor hajtódnak végre. Vizsgáljuk meg a ciklusiterációk *it(n)*, a szorzások *S(n)* és az összeadások *Ö(n)* számát, a polinom fokszámának függvényében.

Feltehető, hogy n≥0, azaz Z.length>0

Első megoldás, az összegzés tételéből származik:

Polinom1(Z:R[]; x:R) :R	*Hányszor fut le (Z.lenght=n+1)*
y:= Z[0]	*1*
i = 1 to Z.length-1	*n+1 (ciklusfeltétel kiértékelés)*
h:=x	*n*
j= 1 to i-1	*1+2+3+…+n-1+n*
h:=h*x	*0+1+2+3+…+n-1*
y:=y+h*Z[i]	*n*
return y	*1*

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

$$Ö(n) = n \qquad it(n) = S(n)$$

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	*Hányszor fut le (Z.lenth=n+1)*
y:=Z[0] h:=1	*1*
i = 1 to Z.length-1	*n+1*
h:=h*x	*n*
y:=y+h*Z[i]	*n*
return y	*1*

$$S(n) = 2*n \qquad it(n) = n$$

$$Ö(n) = n$$

Harmadik megoldás, a Horner séma:

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

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

$S(n) = n$

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

it(n) = n

Jellemezzük a három megoldást a Θ aszimptotikus korlát segítségével.

Megj: it(n) a futási idő nagyságrendjét általában, minden nemrekurzív program esetében is megadja:

	Polinom1	Rekurzív	Horner
S(n)	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$
Ö(n)	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
it(n)	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$

2. **Buborék rendezés** Készítsük el az alap algoritmust, majd a javított változatot. Elemezzük itt is, hogy a struktogram egyes lépései hányszor hajtódnak végre. Nézzük meg az összehasonlítások Öh(n) és cserék számát Cs(n). Cserék elemzésénél használjuk a mCs(n), MCs(n) ACs(n) jelöléseket. Átlagos csere számot nem kell pontosan kiszámolni, elég csak a „megérzés”-re támaszkodni.

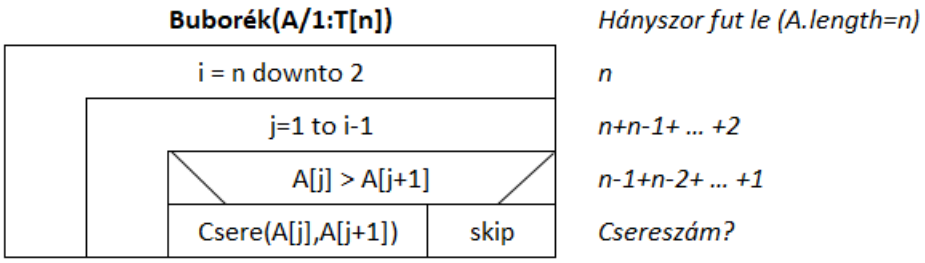
Nézzük meg a rendezés menetét egy rövid példán, majd írjuk fel a struktogramot.

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

Összehaonlítás összesen: 10

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



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

Cserék számát hogyan tudjuk meghatározni?

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

Ebből adódik, hogy mCs(n)=0 (nincs inverzió, azaz növekvően rendezett a bemenet)

MCs(n)= Ö(n) (minden összehasonlítást csere követ, azaz fordítottan rendezett a tömb)

ACs(n)= $\frac{n*(n-1)}{4} = \Theta(n^2)$ Ennek a levezetése a lejjebb megadott linken megtalálható.

Vezessük be az Ω és O aszimptotikus korlátokat, és használjuk a csere számra:
mCs(n)=0, MCs(n)=$\Theta(n^2)$ azaz Cs(n)=$O(n^2)$

Az átlagos futási idő kiszámítása részletesen megtalálható dr Fekete István jegyzetében:
https://people.inf.elte.hu/fekete/algoritmusok_jegyzet/01_fejezet_Muveletigeny.pdf

A 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. Itt elég csak a legkedvezőbb és legrosszabb esetet vizsgálni mÖ(n)∈Θ(n), MÖ(n)∈Θ(n^2). Megemlíthetjük a futási idő jelölést: mT(n)∈Θ(n), MT(n)∈Θ(n^2); azaz mT(n),MT(n)∈Ω(n), mT(n),MT(n)∈O(n)

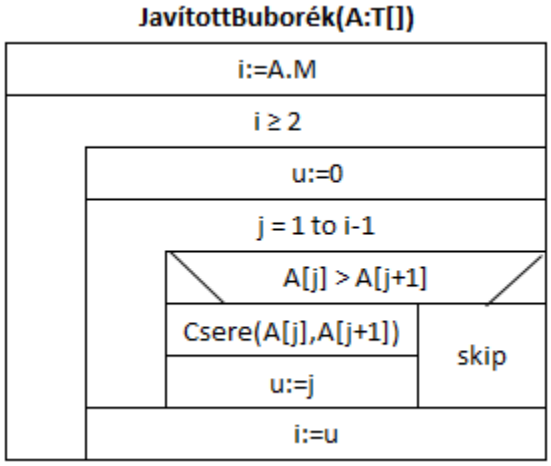
Példa:

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**

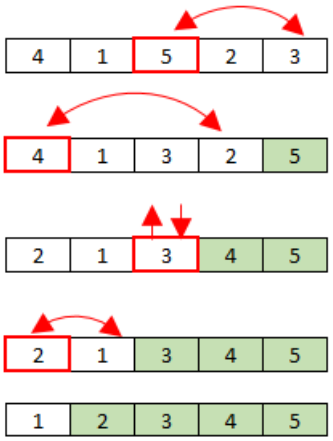
Struktogramja:

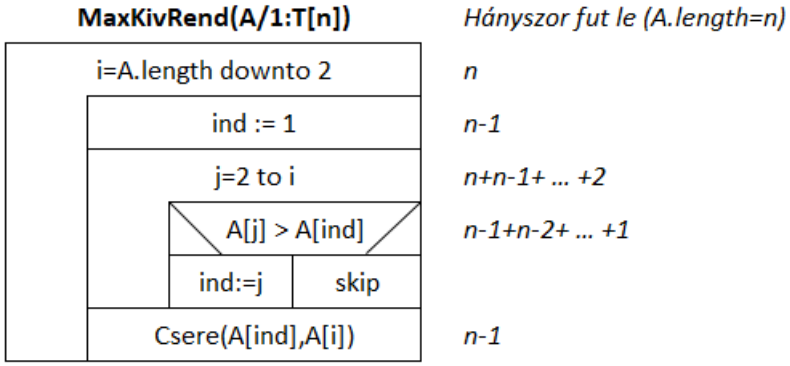


3. **A maximum kiválasztásos rendezés** struktogramjának elkészítése, elemzése.

Mutassuk be a rendezést egy rövid példán, majd írjuk fel a struktogramot. Miért nem érdemes a cserét egy elágazásba tenni? (Maximum értéket azért nem használunk, mert rendezésnél mindig feltesszük, hogy nem csak kulcs, hanem a kulcshoz tartozó rekord is tárolva van a tömbben, melynek mozgatása költséges lenne.)

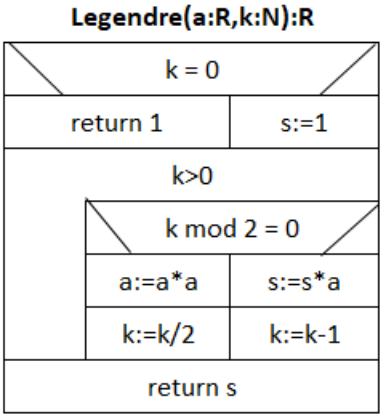
Példa:





Házi feladatok (ezek megoldását a következő gyakorlaton megbeszéljük):

1. Legendre algoritmus műveletigényének meghatározása *k* függvényében:
   Mit számol ki az algoritmus? (Érdemes lejátszani egy példán)



2. Adott egy *n* hosszú, egész számokat tartalmazó tömb. Keressük a tömb azon szakaszát, melynek összege a lehető legnagyobb. (Legyen a tömb neve: A, adjuk meg az a két indexet: 1≤ind1≤ind2≤n, melyre a $\sum_{i=ind1}^{ind2} A[i]$ a maximális.). Elemezzük a megoldás műveletigényét, készítsünk minél hatékonyabb algoritmust! A „Brute-Force" megoldás $\Theta(n^3)$, könnyen javítható $\Theta(n^2)$-re, de van $\Theta(n)$-es megoldás is!