S.85

Jákli Aida Karolina, 11. osztály
Zalaegerszeg, Zrínyi Miklós Gimnázium
jakliaida@freemail.hu
CodeBlocks 12.11 (gcc 4.7.1)

Ha egy tömbben minden *i*-re ($i \leq n$) meghatározzuk, hogy a sorozat első *i* eleme között hány olyan van, ami *c*-nél nagyobb van egyenlő, akkor bármely *j*-re és *k*-ra ($j, k \leq n$; $j \leq k$) könnyen ellenőrizhetjük, hogy a *j*-től a *k*. elemig tartó részsorozat megfelelő-e. Így azonban az összes részsorozatot meg kell vizsgálnunk, ezért a futásidő négyzetes lesz, ami a nagyobb tesztesetekre nem fér bele az 1 másodperces időlimitbe.

Határozzuk meg a sorozat minden elemére, hogy hány olyan (nem feltétlenül megfelelő) részsorozat van, amely az adott elemnél végződik. A részsorozatokat csoportosítsuk aszerint, hogy hány darab *c*-nél kisebb kell elemet beletenni ahhoz, hogy végül azonos legyen a *c*-nél kisebb illetve a *c*-nél nagyobb vagy egyenlő elemek száma. Ha több *c*-nél kisebb elem van, akkor beteendő elemek száma negatív lesz, mert igazából ki kéne venni őket.

A példa bemenet alapján (n=4, c=6):
(a táblázat 1. oszlopa: indexek a tömbben, 2. oszlop: *c*-nél nem kisebb elemek száma – *c*-nél kisebb elemek száma)

		10	5	6	2
0	-4	0	0	0	0
1	-3	0	0	0	0
2	-2	0	0	0	0
3	-1	0	1	0	2
4	0	0	1	1	2
5	1	1	0	2	0
6	2	0	0	0	0
7	3	0	0	0	0
8	4	0	0	0	0

- Egy olyan részsorozat van, ami az első elemnél végződik: a {10}, ebbe egy darab 6-nál kisebb elemet tehetünk.
- A második elemre végződő részsorozatokból kettő van, a {10, 5} és az {5}. A {10, 5} részsorozatban egyenlő a 6-nál nem kisebb és a 6-nál kisebb elemek száma, tehát további elem nem tehető bele. A másikban pedig egy darab 6-nál kisebb elem van, tehát –1 darab 6-nál kisebb elemet tehetünk bele.
- A harmadik elemnél 3 részsorozat végződik, ezek közül kettőbe ({10, 5, 6} és {6}) egy darab 6-nál kisebb elem tehető. Az {5, 6} részsorozatban egyenlő a 6-nál nem kisebb és a 6-nál kisebb elemek száma, ezért további elemet nem tehetünk bele.
- A negyedik elemre végződő részsorozatok közül, kettőbe ({10, 5, 6, 2} és {6, 2}) 0, a másik kettőbe ({5, 6, 2} és {2}) pedig –1 darab 6-nál kisebb elem tehető.

Az összes részsorozat közül azok a megfelelőek, amelyekben a *c*-nél nem kisebb elemek száma legalább az elemszámuk felével egyenlő. Tehát a táblázat 4-8. sorában lévő számok összegzésével kapjuk meg a megoldást. (A táblázat összes elemének összegzésével megkapjuk, hogy összesen hány részsorozata van a sorozatnak.)

Ez így önmagában szintén négyzetes futásidejű lenne, és a nagyobb teszteseteknél a táblázat nem férne el a memóriában. Azonban nem szükséges a teljes táblázatot tárolni, elég, ha csak az éppen aktuális oszlopot tároljuk.
Egy adott elemre végződő részsorozatot úgy kapunk, hogy az előző elemre végződő részsorozatok végére berakjuk az adott elemet. Ezáltal a betehető elemek száma, az adott elem értékétől függően nő vagy csökken. Tehát a példa bemenet alapján az első elemre végződő részsorozat a {10}, ebbe egy darab 6-nál kisebb elem tehető, ezért ezt a részsorozatot a tömb 5. ele-

S.85
Jákli Aida Karolina, 11. osztály
Zalaegerszeg, Zrínyi Miklós Gimnázium
jakliaida@freemail.hu
CodeBlocks 12.11 (gcc 4.7.1)

mében fogjuk számon tartani. Ha ennek a végére berakjuk az 5-öt, akkor a {10, 5} sorozatba további elemet már nem tehetünk, ezt a részsorozatot a tömb 4. elemében tartjuk számon. Ezen kívül van még egy részsorozat, az {5}, ebbe –1 darab elemet tehetünk (tömb 3. eleme).

		10	5	6	2
0	-4	0	0	0	0
1	-3	0	0	0	0
2	-2	0	0	0	0
3	-1	0	1	0	2
4	0	0	1	1	2
5	1	1	0	2	0
6	2	0	0	0	0
7	3	0	0	0	0
8	4	0	0	0	0

Tehát lényegében olyan, mintha „eltoltuk” volna a tömb elemeit, majd ezután a megfelelő elemet növeljük. Ha az adott elem kisebb, mint *c*, akkor a tömb eleje felé kell tolni az elemeket és a –1-nek megfelelő elemet kell növelni, ha az adott elem nem kisebb, mint *c*, akkor a tömb vége felé kell tolni a tömbelemeket és az 1-nek megfelelő elemet kell növelni. Elképzelhető, hogy a sorozat csak *c*-nél kisebb, vagy csak *c*-nél nem kisebb elemeket tartalmaz, ekkor az elemeket vagy az egyik vagy a másik irányba kéne *n*-nel eltolni, ezért $2 \cdot n + 1$ hosszúságú tömbre van szükség. Ebben a tömbben az *n*. elem jelenti azokat a részsorozatokat, amelyekben a kétféle elemek száma megegyezik.

A tömb elemeit nem szükséges másolgatni, úgy is elképzelhetjük, hogy nem a tömbelemeket, hanem a 0 helyét tologatjuk (a nulla helye kezdetben a tömb *n*. eleme). Pl.: ha a {10} részsorozatba beletesszük az 5-öt, akkor a tömb eleje felé kéne eltolni az elemeket, tehát a tömb 5. eleme kerülne a 4. helyére, ezért a nulla helye 4 helyett 5 lesz. Tehát ha az adott elem kisebb *c*-nél, akkor eggyel növeljük, egyébként pedig eggyel csökkentjük a nulla helyét. Ezután a –1 illetve az 1 helye is meghatározható.

Mivel a tömb minden lépésben az *i*. elemnél végződő részsorozatok számát adja meg, ezért minden lépésben meg kell határozni a megfelelő részsorozatok számát, tehát összegezni kell a tömb azon elemeit, amelyek a nulla helyén vagy hátrébb vannak. Ha a tömb elemeit csak önmagukban tároljuk, akkor az összeg meghatározása miatt négyzetes lesz a futásidő. Ha a tömb *i*. eleme az *i*-nél kisebb vagy egyenlő elemek összegét jelenti, akkor pedig az elemek növelése fog sok időt igénybe venni. Ha az elemeket Fenwick-fában (KöMaL, S.73 megoldás) tároljuk, akkor hatékonyabb programot kapunk. Így könnyen meghatározhatjuk a megfelelő részsorozatok számát, az összes, *i*. elemnél végződő részsorozat számából levonjuk a tömbben a nulla helye előtti elemek összegét (a nem megfelelő részsorozatok száma).