

Java prgramozás 0. házifeladat

Boyer - Moore majority vote

A feladat Boyer - Moore majority vote algoritmus ellenőrzéssel kiegészített változatát kell implementálni néhány megkötéssel. Az algoritmus célja, hogy megtalálja egy adatsorban a többségi elemet (azt az elemet ami az esetek több mint 50%-áért felelős) $O(n)$ időben és $O(1)$ helyigénnyel.

Az eredeti algoritmus:

- Initialize an element m and a counter i with $i = 0$
- For each element x of the input sequence:
 - If $i = 0$, then assign $m = x$ and $i = 1$
 - else if $m = x$, then assign $i = i + 1$
 - else assign $i = i - 1$
- Return m

Az algoritmus maga nem garantálja azt, hogy a visszaadott m érték ténylegesen a többségi elem lesz ezért a végén még ellenőrizni kell, hogy valóban többségi elem lesz. Amódosított algoritmus pseudo kódja:

- Initialize an element m and a counter i with $i = 0$
- For each element x of the input sequence:
 - If $i = 0$, then assign $m = x$ and $i = 1$
 - else if $m = x$, then assign $i = i + 1$
 - else assign $i = i - 1$
- $i = 0$
- For each element x of the input sequence
 - If $x = m$, then $i = i + 1$
- If $i \leq n/2$, where n is the length of the sequence, then $m = -1$
- Return m

A feladat során pozitív egész számokból álló sorozatokon kell többségi elemet keresni így a -1 jelentése az, hogy nem található többségi elem.

A feladat a kiadott keretrendszerben implementálni ezt az algoritmust a *static int majorityVote(int[] sequence)* függvényben.

Heavy Hitters (Opcionális)

A fenti algoritmus egy általánosított verziója a Heavy Hitters algoritmus. Ennek célja, hogy $O(n)$ idő és $O(\epsilon)$ tárhely komplexitással keressen olyan elemeket egy sorozatban, amelyek a sorozat $1/k$ -ad részéért felelősek. Az algoritmus ugyanazt a számlálás technikát alkalmazza mint a fenti esetben, azonban nem 1 számláló van, hanem $k - 1$ darab és akkor csökkentjük egyet az összes számlálót, amikor a k -adik különböző elemet dolgozzuk fel.

Az algoritmus idő komplexitásához fontos feltétel, hogy $O(1)$ idő alatt tudjuk adott elemről eldönteni, hogy éppen számon tartjuk-e és ha igen akkor a számláló elérése is $O(1)$. Ehhez legegyszerűbben valamilyen *HashMap*-et kéne alkalmaznunk. Azonban ha $O(kn)$ -es időkomplexitást engedélyezünk, akkor az egyszerűség kedvéért használhatunk $k - 1$ hosszú tömböket az értékek és a számlálók tárolására és követésére (természetesen a *HashMap** alkalmazása jobb megoldás).

Itt is fontos az algoritmus végén annak ellenőrzése, hogy a talált értékek ténylegesen többségi elemek-e.

Az algoritmusról egy jól érthető összefoglalót és magyarázatot találhattok a következő videó az első 12 percében:

videó