Maksimum drsečega okna, Timen Bobnar

OSNUTEK

# **Povzetek \*\*\* Končno\*\*\***

Na povezavi ([1]) je predstavljena naloga maksimuma drsečega okna. Zanimajo nas vsi maksimumi v okencu dolžine **k** ko se sprehajamo po seznamu števil. Ta problem bi radi rešili z implementacijo vrste, ki ima le nekaj osnovnih operacij (vstavi, vrh, odstrani, prazen). *Najprej si bomo ogledali način predstavitve vhodnih podatkov problema ter s pomočjo primera predstavili, zakaj pri problemu točno gre. V naslednjem razdelku si bomo ogledali algoritem za reševanje našega problema. Algoritem prikažemo še na primeru. Nalogo končamo s prikazom ključnih delov naloge.*

# **Problem**

*Na spletni strani [1] je podan problem maksimumi drsečega okna. Poglejmo, zakaj gre.*

*Naši vhodni podatki so seznam števil, kjer je seznam, dolžine med 1 in 10^5 ter vsak element omejen med -10^4 in 10^4. Dobimo tudi podatek* ***k****, ki je omejen med 1 in dolžino seznama, in predstavlja velikost našega drsečega okna.*

*Ta problem bomo reševali z vrstami, ki imajo 4 osnovne funkcije, in sicer vstavi, beri, odstrani in prazna.*

# **Ideja rešitve**

*Najprej moramo iz seznama narediti vrsto.*

*Sedaj zares pridemo do našega problema. Sprehajali se bomo čez vrsto in vsakič, ko iščemo maksimum v kosu naše vrste dolžine* ***k****, si**bomo prvi element katerega preberemo zapomnili kot maksimum in odstranili. Sedaj preverimo še nadaljnjih* ***k-1*** *elementov, in če je, kakšen večji si ga zapomnimo in naprej primerjamo z njim, vse elemente prepisujemo v pomožno vrsto, da jih ne pozabimo. Če naše okence dolžine k, ne more v celoti ''sesti'' v našo vrsto potem smo zaključili. V primeru da ''okence'' v celoti ''sede'' v vrsto potem naše število, ki smo si ga zapomnili, kot maksimum damo v neko vrsto recimo ji maksimi. Vse preostale elemente v osnovni vrsti prepišemo v pomožno vrsto. In iz pomožne vrste nazaj v osnovno. Sedaj imamo enako osnovno vrsto kot na začetku z eno razliko, in sicer da smo odstranili prvi element vrste.*

*Ko se to zaustavi našo vrsto maksimi prepišemo v seznam, in vrnemo kot rezultat.*

# **Uporaba na primeru \*\*\* Končno \*\*\***

*Naj bodo naši osnovni podatki ( [ 4, 5, 32, 42, 7, 38, 5, -5, -3, -2, -15, 3 ] , 4 ). Seznam najprej prevedemo v vrsto, in sicer da je videti tako:*

***vrsta\_osnovna****=****začetek*** *: 4, 5, 32, 42, 7, 38, 5, -5, -3, -2, -15, 3 :* ***konec.***

*Začnimo sprehajanje. Preberemo prvi element in si ga zapomnimo kot maksi=4 nato ga odstranimo. Sedaj naša vrsta izgleda tako vrsta\_osnovna=****začetek*** *: 5, 32, 42, 7, 38, 5, -5, -3, -2, -15, 3 :* ***konec.*** *Pogledamo naslednje 3 element in jih primerjamo z maksi-mom. Elemente prepišemo v novo vrsto pomožna.*

*Sedaj naši podatki izgledajo tako:*

1. *vrsta\_osnovna=****začetek*** *: 7, 38, 5, -5, -3, -2, -15, 3 :* ***konec,***
2. ***vrsta\_pomožna=začetek: 5,32,42:konec,***
3. ***maksi=42.***

*Sedaj maksi vstavimo v prazno vrsto* ***vrsta\_maksi zacetek: 42:konec*** *in ga pozabimo. Vse elemente v osnovni vrsti prepišemo v pomožno in nato iz pomožne nazaj v osnovno. Sedaj naši podatki izgledajo tako.*

1. ***vrsta\_osnovna=začetek : 4, 5, 32, 42, 7, 38, 5, -5, -3, -2, -15, 3 : konec***
2. ***vrsta\_pomožna=začetek : konec***
3. ***maksi=None***
4. ***vrsta\_maksi zacetek: 42: konec***

*Ta postopek ponavljamo dokler ne pridemo do položaja:*

1. ***vrsta\_osnovna=začetek : -2, -15, 3 : konec***
2. ***vrsta\_pomožna=začetek : konec***
3. ***maksi=None***
4. ***vrsta\_maksi zacetek: 42,42,42,42,38,38,5,-2,3:konec***

*V tem koraku si zapomnimo najprej -2 in izbrišemo. Nato preberemo -15 in prepišem v pomožno vrsto. Sedaj preberemo 3, si zapomnimo in 3 prepišemo v pomožno vrsto. Ker je k = 4 in smo mi trenutno prebrali le 3 elemente moramo še enega, ki ga ni zato končamo.*

**Naš rezultat: [42,42,42,42,38,38,5,-2,3].**

# **Programska rešitev**

Koda kot taka ne vsebuje veliko posebnosti. Edina stvar ki se jo splača omeniti je uporaba booleanov.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava

Opis je samodejno ustvarjen

tu smo uporabili boolean **bol**, zato da lahko zaključimo našo zanko ob pravem času, in sicer ko je naša osnovna vrsta prekratka.

**Analiza časovne zahtevnosti:**

Naj bo naša operacija vstavi, odstrani, beri.

Naj bo n dolžina vhodnega seznama(n) in števila(k). Potem je naša časovna zahtevnost=O(n^3-k^3)

Saj je število operacij približno n+(n-k)sum(i=2,n-k)(3+3i+3i-3) + len(rešitve). Kjer je vsota enaka

3\*(k^2-k(2n+1)+n^2 +n-2). Če to napademo z O-jem dobimo n^3+k^3.

# **Viri**

[1] LeetCode, (17.11.2023), Sliding Window Maximum(239)

https://leetcode.com/problems/implement-stack-using-queues/.

­­­