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**

\*\*\* V tem razdelku natančno predstavim, zakaj gre pri problemu. Problem bom ilustriral z več slikami konkretnega dogajanja \*\*\*

# **Ideja rešitve \*\*\* Končno \*\*\***

Da se izognemo nepotrebnemu premetavanju podatkov oziroma iskanjem maksimumov, bomo v problem vpeljali vrsto. Vrsta ima nekaj lepih lastnosti, ki v našem primeru pridejo prav, te lastnosti so vstavi, odstrani in preberi. Potrebovali bomo tudi števec, ki bo shranjeval vrednost o tem, kolikokrat se je pojavim maksimum v našem trenutnem oknu.

Za optimalno rešitev našega algoritma imamo 5 pogojev, ki nam veliko olajšajo naše iskanje maksimuma.

V prvem koraku prestavimo k(širina drsečega okna) podatkov v našo pomožno vrsto ter med prelaganjem poiščemo maksimum. Trenutno pomožna vrsta predstavlja naše drseče okno. Sedaj vzamemo element iz začetnih podatkov ter pogledamo kakšna je njegova vrednost. Iz pomožne vrste odstranimo podatek ter pogledamo kakšna je njegova vrednost. V primeru ko je odstranjen podatek maksimum, števec zmanjšamo za 1. Sedaj nastopi par pogojev, ki jih je potrebno pregledati. V primeru ko vstavimo podatek večji od sedanjega maksimumam vemo da je ta podatek novi maksimum, hkrati moramo tudi števec resetirati na 1. V primeru ko vstavimo podatek majši od maksimuma in je števec večji od 0 (torej se še en maksimum nahaja v vrsti) vemo , da je naš novi maksimum enak staremu. V primeru ko vstavimo element enak maksimumu števec povečamo za ena in maksimum ostane isti. Sedaj imamo le še eno situacijo in sicer ko vstavimo element manjš od maksimuma in je števec enak 0(odstranili smo maksimum). Taka situacija je zelo neugodna saj sedaj ne vemo koliko je maksimum in moramo preverit vse podatke. Ta postopek ponavljamo dokler je še kakšen podatek v začetnih podatkih.

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

Naj bodo naši osnovni podatki ( [ 3, 2, 1, 2, 4, 3, 5, 5, 3, 2] , 3).

Po prvem koraku imamo situacijo:

Imamo situacijo [2, 4, 3, 5, 5, 3, 2], max=3 ,števec=1 in vrste= zac:3,2,1:konec rezultat=[3]

Sedaj v naslednjem koraku izločimo iz vrste 3 ter ustavimo 2. Ko smo izločili 3 se je tudi števec postavil na 0. Dobimo vrsto zac:2,1,2:konec ker v tem koraku ne vemo že v naprej kaj je maksimum moramo pregledati celotno vrsto tako dobimo da je maksimum enak 2 ter števec enak 2. rezultat=[3,2]

V naslednjem koraku vstavimo 4 ker je 4>2 je maksimum enak 4 stevec enak 2 in rezultat enak [3,2,4].

Trenutno naši podatki izgledajo tako [3, 5, 5, 3, 2], max=4, števec=1 in vrsta=zac:1,2,4:konec.

Sedaj bi v vrsto dodali 3 ki je manjša od 4, vemo tudi da 4 nismo izločili torej je maksimum 4:

Trenutno naši podatki izgledajo tako [5, 5, 3, 2], max=4, števec=1 in vrsta=zac:2,4,3:konec.

Sedaj sledi četrti in peti korak kjer vstavimo število pet in tako kot ko smo vstavljali 4 sedaj 5 postane maksimum in imamo sedaj podatke [3,2], max=5, števec=2 vrsta=zac:3,5,5:konec rezultat je do sedaj [3,2,4,4,5,5]. Sedaj vstavimo 3 , ker je 3<od 5 in pet nismo odstranili je spet 5 maksimum.

Sedaj imamo še zadnji korak. V zadnjem koraku odstranimo eno petico torej števec postane 1 in vstavimo 2 ker nam števce pove da je vsaj ena 5 še notri in je 5>2 sledi da je zadnji maksimum enak 5.

# **Programska rešitev**

\*\*\*Tukaj bomo pokazali kako izgleda koda\*\*\*

# Analiza časovne zahtevnosti:

\*\*\*malo bomo povedali o časovni zahtevnosti\*\*\*

# **Viri**

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

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

­­­