Problem implementacije sklada z uporabo vrste, Urša Kumelj

OSNUTEK

# **Povzetek \*\*\* Končno\*\*\***

*V prispevku je predstavljen problem implementacije sklada z uporabo vrst* (LeetCode, 2023)*. Razred Sklad želimo predstaviti z uporabo razreda Vrsta. To pomeni, da konstruiramo razred Sklad, ki se bo obnašal kot ta podatkovna struktura tako, da podatke hrani v vrsti (ali več vrstah). Metode odstrani (odstrani vrhnji element v skladu), vstavi (vstavi element na vrh sklada), vrh (vrne element na vrhu sklada) in prazen (pove, ali je sklad prazen ali ne) moramo torej izvesti s pomočjo operacij nad vrstami. Najprej si ogledamo primer in na podalgi le-tega opšemo, kako k problemu sploh pristopimo. V naslednjem razdelku predstavimo možne načine za reševanje tega problema. Prav tako predstavimo tistega, s katerim smo se odločili problem rešiti in objasnimo zakaj taka odločitev. Algoritem prikažemo še na primeru. Prispevek zaključimo s prikazom ključnih delov kode rešitve in obrazložitvijo časovne zahtevnosti. Na koncu dodamo še nekaj besed o samih težavah pri reševanju tovrstnega problema.*

# **Problem\*\*\*končno\*\*\***

Na spletni strani (LeetCode, 2023) je podan problem implementacije sklada z uporabo vrst. Poglejmo, za kaj gre.

*Implementiraj sklad (LIFO last in first out) z uporabo le dveh vrst. Implementirani sklad naj podpira vse funkcije običajnega sklada (vstavi, vrh, odstrani, prazen).*

*Implementiraj razred Sklad:*

1. *Vstavi(x): vstavi element x na vrh sklada*
2. *Odstrani(): odstrani element z vrha sklada in ga vrne*
3. *Vrh(): vrne element na vrhu sklada*
4. *Prazen(): vrne True, če je sklad prazen in False sicer*

*Zaznamek: Uporabljati smete samo običajne operacije vrste, kar pomeni, da so veljavne le operacije vstavi (vstavi element na konec vrste), odstrani (odstrani element na začetku vrste) in prazna (preveri ali je vrsta prazna ali ne).*

Če imamo sklad

A diagram of a number

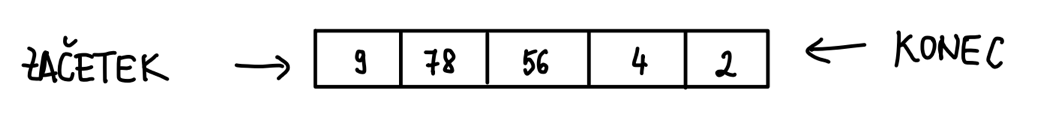
Description automatically generated

Imamo dve "očitni" možnosti predstavitve z vrsto. Prvi, kjer je vrsta sledeča

A black rectangular with numbers

Description automatically generated with medium confidence

recimo ***enaka predsavitev,*** drugi, kjer je vrsta



pa ***obratna predstavitev.***

# **Ideja rešitve**

Naj bo primer za sklad enak

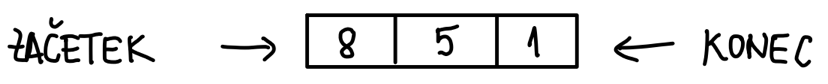
A number grid with arrows and numbers

Description automatically generated with medium confidence

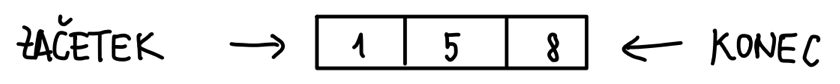
in se nanj sklicujmo v nadaljevanju.

Ko se lotimo naloge, imamo dva očitna načina predstave sklada z vrsto in sicer lahko ga predstavimo kot:

1. Enaka predstavitev



1. Obratna predstavitev



Če se odločimo za 1.način, bo naša implementacija bolj zapletena, ker:

1. Za vrh sklada bi se potrebovali sprehoditi skozi celotno vrsto in na koncu shraniti zadnji element. V primerjavi z drugim primerom se sliši težje, saj je vrh drugega načina ravno prvi element, za katerega poznamo metodo začetek iz vrste.
2. Za odstranitev iz sklada pa bomo prav tako rabili nekako odstraniti vrhnji element, pri tem pa ne smemo ustvariti nove vrste brez vrhnjega elementa in jo potem vrniti, ker ta metoda ne vrača. Zveni precej zakomplicirano, saj pri uporabi drugega načina le odstranimo prvi element v vrsti.
3. Za vstavitev elementa na vrh sklada pa ne bomo imeli preveč dela, saj bomo podatek samo vstavili na konec vrste.

Če pa se odločimo za 2. način, bomo prav tako imeli težave, ker bi se za vstavitev elementa na vrh sklada potrebovali sprehoditi skozi celotno vrsto, kar je težje kot pri prvem načinu.

Da se o tej odločitvi prepričamo, si poglejmo še tabelo časovne zahtevnosti za posamezno metodo. Podrobnejša razlaga sledi na koncu.

|  |  |  |
| --- | --- | --- |
| METODA | 1.NAČIN | 2.NAČIN |
| Vstavi(x) | O(1) | O(n) |
| Odstrani() | O(n) | O(1) |
| Vrh() | O(n) | O(1) |
| Prazen() | O(1) | O(1) |

## **Obratna predstavitev \*\*\* Končno \*\*\***

Da se izognemo nepotrebnim zapletom, opisanim zgoraj, se je bolj smiselno odločiti za obratno predstavitev (2. način), ker bomo še vseeno imeli manj težav kot pri enaki predstavitvi (1. način). Prav tako pa iz tabele vidimo, da bomo imeli samo enkrat časovno odvisnost linearno, sicer pa konstantno.

Opišimo sedaj, kako bi se v splošnem lotili reševanja za posamezno metodo:

1. prazen\_sklad(): tukaj nimamo posebnosti. V primeru, da je vrsta prazna, bo prazen tudi sklad. Tako vrnemo True v primeru, ko to drži in False sicer.
2. odstrani\_sklad(): kot smo že opisali zgoraj v ideji rešitve, bomo odstranili prvi element v vrsti.

A black and red text

Description automatically generated with medium confidence

1. vrh\_sklad(): tudi tukaj bomo vrnili ravno prvi element vrste.

A red circle with black text

Description automatically generated

1. vstavi\_sklad(x): imamo dve možnosti in sicer, da uporabimo metodo stražarja ali pa pomožno vrsto. To bo nekoliko bolj podrobneje razloženo v razdelku Programska rešitev.

A white board with black text and red x

Description automatically generated

# **Uporaba na primeru\*\*\*končno\*\*\***

Poglejmo si, kako bi zgoraj opisane metode delovale na konkretnem primeru.

Naj bo sklad s1 enak

A black and white image of a number

Description automatically generated

Kaj dobimo po klicu posamezne metode na zgornji primer vrste v1?

1. s1.prazen\_sklad(): sklad sedaj ni prazen in bomo dobili False
2. s1.vrh\_sklada(): dobili bomo 500

A red circle with numbers and a number in a row

Description automatically generated

1. s1.odstrani\_sklad(): dobili bomo

A red x and black text

Description automatically generated

1. s1.vstavi(9): dobili bomo

A black and white image of a number

Description automatically generated with medium confidence

Lahko pa bo nekdo želel poklicati metodi vrh in odstrani na že praznem skladu, takrat bomo sprožili napako.

# **Programska rešitev\*\*\*končno\*\*\***

Poglejmo si morda najtežji del kode. To je metoda vstavi\_sklad(x). Pri tej je ključna ideja uporaba stražarja. Povedali smo, da je potrebno na začetku v prvotno vrsto vstaviti stražarja (vrstica 90). Kaj je vrednost našega stražarja, je vseeno. Naj bo to recimo None. Nato vstavimo še podatek x (vrstica 91). Potem se sprehajamo skozi vsak element v vrsti, dokler ponovno ne pridemo do stražarja (vrstica 92). V zanki vsak element iz začetka, prestavimo na konec (vrstici 93 in 94). Ne smemo pa pozabiti elementa iz začetka tudi izbrisati (vrstica 95). Zanka se nato ustavi, saj smo prišli do stražarja. Vendar stražar je še vedno v vrsti in sicer na začetku. Tega pa ne želimo, zato ga na koncu odstranimo (vrstica 96).

A computer screen with text

Description automatically generated

# **Časovna zahtevnost\*\*\*končno\*\*\***

Pri analizi se moramo sprva dogovoriti, kaj je velikost našega problema in kaj je karakterisitična operacija. Glede na to, da imamo 4 metode, potrebujemo to določiti pri vsaki posebej. Naredimo tabelo, kjer bomo to določili.

|  |  |  |
| --- | --- | --- |
| METODA | KARAKTERISTIČNA OPERACIJA | VELIKOST PROBLEMA |
| Vstavi(x) | prestavljanje elementov iz začetka na konec vrste | poljubno velika vrsta |
| Odstrani() | odstranjevanje | poljubno velika vrsta |
| Vrh() | pregledovanje | poljubno velika vrsta |
| Prazen() | pregledovanje | poljubno velika vrsta |

Tabelo časovne zahtevnosti smo naredili že pri ideji rešitve. Razložimo lahko še zakaj O(1) in O(n):

* O(1): vedno potrebujemo pregledati/odstraniti le prvi element, zato je to delo vedno konstantno
* O(n): potrebujemo se sprehoditi skozi celotno vrsto

Na tem mestu bi se morda vprašali, ali obstaja kakšen način, da bi povsod dobili konstantno časovno zahtevnost. Recimo, da bi uporabili dve vrsti ena, ki opisuje 1. način in druga, ki opisuje 2. način. Tudi tukaj bi se nam zataknilo in ne bi prišli do boljših časovnih zahtevnosti. Morda pa obstaja kakšen način, ampak jaz zanj ne vem.

**Težave\*\*\*končno\*\*\***

Največ težav sem imela, ker sem se razlage svojega problema lotila tako, da sem izhajala iz vrste in ne iz sklada. Po konzultacijah me je profesor na te stvari opozoril, zato sem jih popravila. Prav tako mi je predlagal razlago načinov reševanja s prikazom tabele časovne odvisnosti, ki sem jo tako vključila v predstavitev in opis. Predlagal je tudi razmislek, če obstaja še kakšen boljši način poleg omenjenih. Vse pripombe po konzultacijah sem vključila v predstavitev in v opis.

# **Viri**

LeetCode. (brez datuma). *Implement Stack using Queues.* https://leetcode.com/problems/implement-stack-using-queues/.

­­­