SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

PROJEKT

ALGORITMI U PRIMJENI

POHLEPNI ALGORITMI

Profesorica: Student:

Divna Krpan Tomislav Damjanović

Split, srpanj 2020.

Sadržaj

[Uvod 1](#_Toc46792485)

[Problem postavljanja polica (Fitting shelves problem) 3](#_Toc46792486)

[PROJEKT 4](#_Toc46792487)

[Primjer 1. 5](#_Toc46792488)

[Primjer 2. 5](#_Toc46792489)

[ZAKLJUČAK 6](#_Toc46792490)

# Uvod

Općenito, algoritam je niz koraka kojih se trebamo držati kako bi riješili određeni zadatak ili problem. Nakon konačnog broja koraka dolazimo do rješenja, ili saznajemo da rješenje uopće ne postoji. Drugim riječima, algoritam je skup svih operacija potrebnih za rješavanje određenog zadatka po točno određenom rasporedu. Algoritam također mora imati i ispravan postupak. Na samom ulazu se mora definirati problem, a na izlazu riješenje. Ako smo dobro definirali i postavili problem, i ako se ne pronađe nijedna greška-smatramo da je algoritam (postupak) uspješan.

“Pohlepnost” je algoritamska paradigma koja gradi rješenje pojedinačno, uvijek birajući sljedeći dio koji nudi najočitiju i neposrednu korist. Dakle, u pitanju su problemi u kojima odabir lokalno optimalnog rješenja vodi i globalnom rješenju.

Na primjer, ako razmotrimo problem sa “frakcijskim ruksakom” [[1]](#footnote-1). Lokalna optimalna strategija je odabrati predmet koji ima omjer maksimalne vrijednosti prema težini. Ova strategija dovodi i do optimalnog globalnog rješenja jer smo dozvolili uzeti frakcije predmeta.

A close up of a sign

Description automatically generated

**Učinkovito rješenje** je da koristimo pohlepni pristup. Osnovna ideja pohlepnog pristupa je izračunati vrijednost omjera / težine za svaku stavku i sortirati predmet na temelju ovog omjera. Zatim uzmemo stavku s najvećim omjerom i dodamo je dok sljedeću stavku ne možemo dodati u cjelini, a na kraju sljedeću stavku dodamo koliko možemo. Ovo će uvijek biti optimalno rješenje ovog problema.

Postoje 4 vrste pohlepnih algoritama, odnosno algoritama sa pohlepnim pristupom:

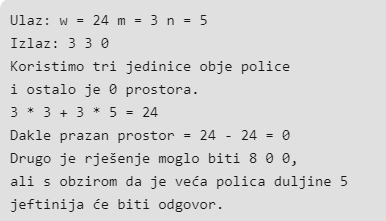
* Standardni algoritmi sa pohlepnim pristupom
* Algoritmi sa pohlepnim pristupom u grafovima
* Algoritmi sa pohlepnim pristupom u operacijskim sustavima
* Algoritmi sa pohlepnim pristupom u nizovima

U ovom projketu obrađena je implementacija jednog standardnog algoritma sa pohlepnim pristupom.

# Problem postavljanja polica (Fitting shelves problem)

S obzirom na duljinu zida w i dvije police duljine m i n, treba pronaći broj svake vrste polica koje se koriste i preostali prazan prostor u optimalnom rješenju, tako da prazan prostor bude minimalan. Veća od dvije police je jeftinija pa ima prednost. Međutim, trošak je sporedan i prvi prioritet je minimizirati prazan prostor na zidu.

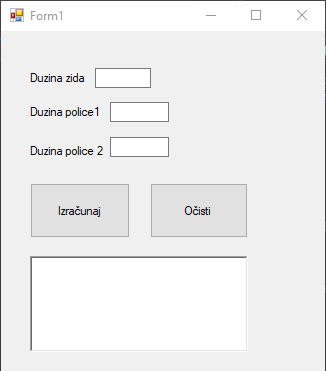
Primjer:



Jednostavan i učinkovit pristup bit će isprobati sve moguće kombinacije polica koje se uklapaju u duljinu zida. Da bismo implementirali ovaj pristup zajedno s ograničenjem da veća polica košta manje od one manje, počevši od 0, povećavamo broj većih polica dok one ne budu prikladne. Također, izračunamo prazan prostor i na kraju spremimo tu vrijednost koja minimizira prazan prostor. Ako je prazan prostor isti u dva slučaja, preferiramo onaj s većim brojem većih polica.

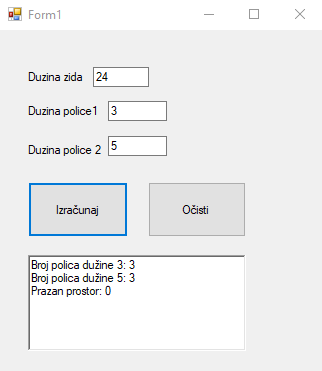
# PROJEKT

Projekt je zamišljen tako da korisnik unosi vrijednosti koje su potrebne da se dobije konačni rezultat, odnosno dužina zida, te dužina veće i manje police.

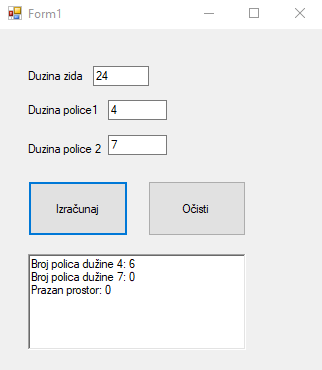


Osim textBox-ova koji služe za unos podataka, na formi se nalaze i dva botuna. Prvi-„Izračunaj“ služi za pokretanje našeg algoritama i ispis podataka na RichTextBox, a drugi-„Očisti“ briše sve iz RichTextBox-a te iz textBox-ova i omogućuje korisniku ponovni upis podataka.

## Primjer 1.



## Primjer 2.



# ZAKLJUČAK

Zašto je ovo pohlepni pristup? Kako je svaki korak koji algoritam poduzme u ciklusu optimalan, a slijedeći korak je manji od prethodnog, kao što možemo vidjeti, algoritam se uvijek dijeli i smanjuje dužinu zida. Drugim riječima, sastoji se od toga da uvijek donese najbolju odluku od svih onih koje možemo donijeti odmah, nadajući se da ćemo zajednicom svih ovih malih najboljih odluka, odlučiti o najboljem rješenju problema općenito.

1. S obzirom na težinu i vrijednosti n predmeta, stavke moramo staviti u ruksak kapaciteta W da bismo dobili maksimalnu ukupnu vrijednost u ruksaku. U **„frakcijskom ruksaku“** možemo razbiti predmete za maksimiziranje ukupne vrijednosti ruksaka. Taj problem u koji možemo razbiti predmet naziva se i problemom frakcijskog ruksaka. [↑](#footnote-ref-1)