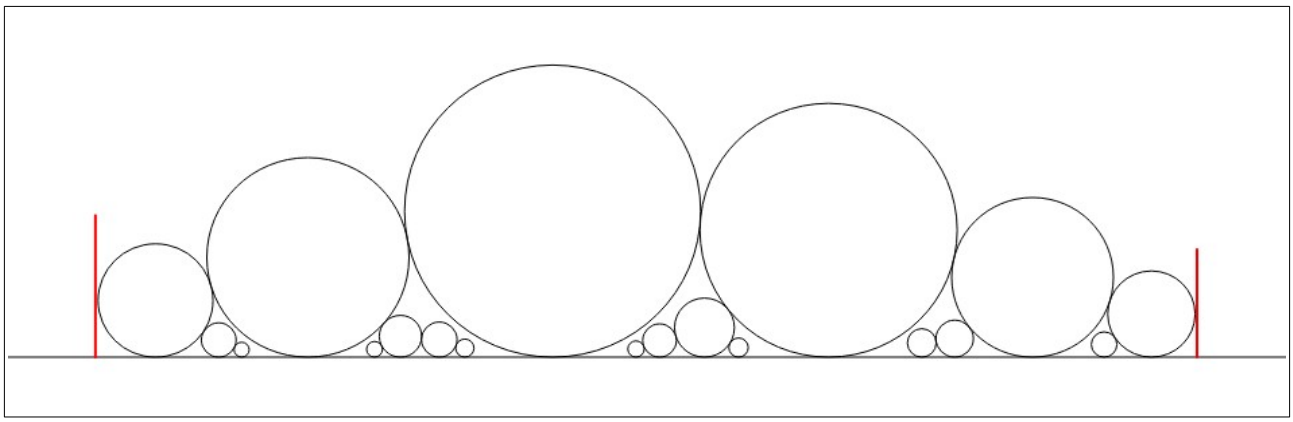
Coins on Shelf

- Projekat za potrebe kursa *Geometrijski algoritmi* -

Boris Karanović, 1113/2020



Validan raspored diskova sa granicama police

- Opis problema -

Na raspolaganju imamo *N* diskova različitih poluprečnika, od kojih neki mogu biti isti. Cilj nam je da rasporedimo sve diskove tako da dodiruju jednu horizontalnu liniju (liniju koja ce biti zajednička tangenta za sve diskove), koju ćemo nadalje nazivati *policom*. Diskovi se ne smeju preklapati, a većina njih će se dodirivati sa levim i desnim susedom. Kada ovako postavimo diskove na policu, povlačimo dve vertikalne linije (na slici su crvene boje) i računamo dužinu police određene sa presecima tih pravi i police. Krajnji cilj nam je da ta dužina bude što manja, tj. da diskovi zauzimaju što manje prostora.

Ulaz algoritma: Multiskup diskova različitih poluprečnika, ukupno *N*.
Izlaz algoritma: Veličina police na koju oni mogu biti smešteni.

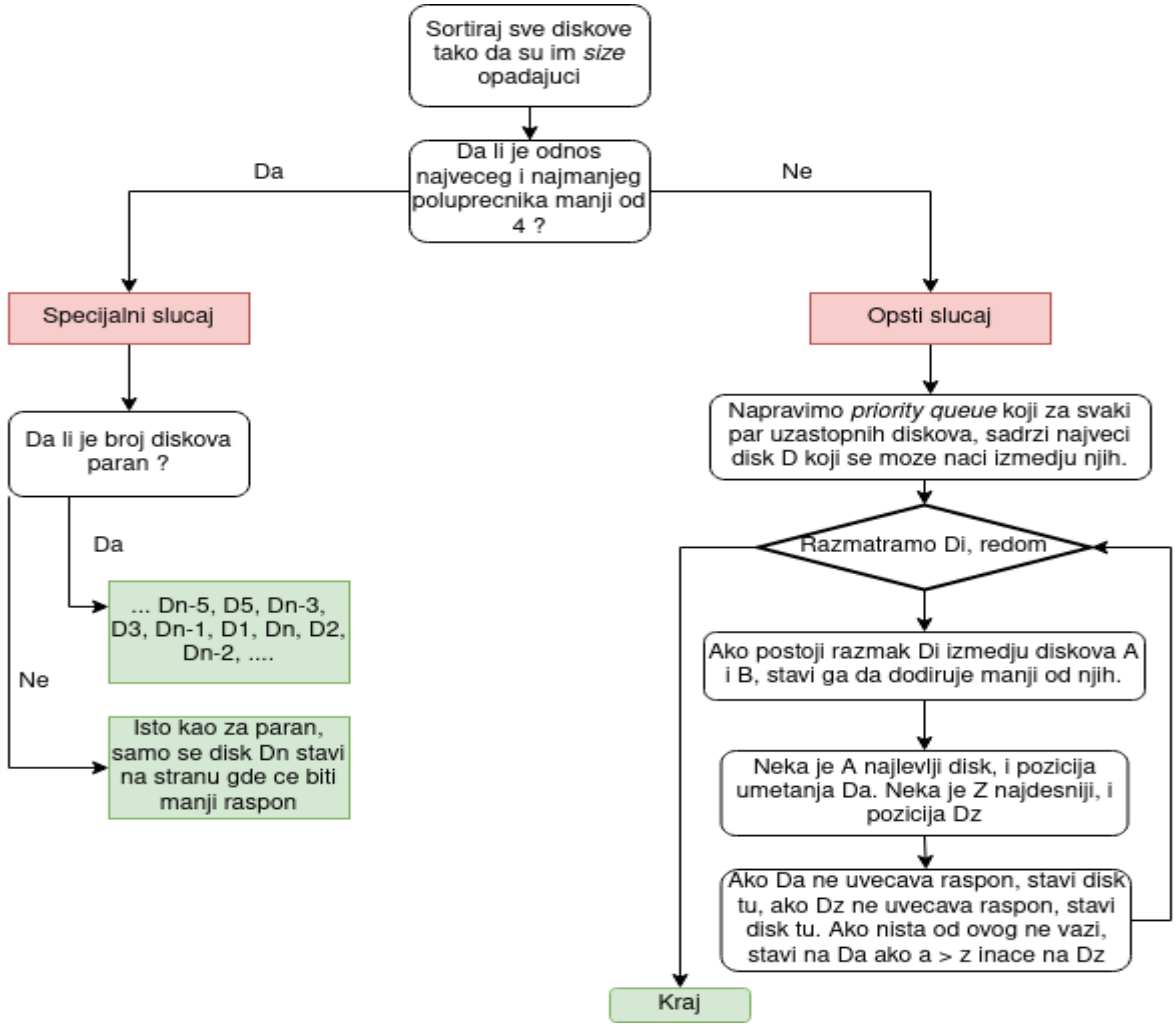
- Razmatranje slučajeva problema -

Problem se može podeliti na dva manja podproblema, sa ciljem rešavanja lakših pojedinačnih problema. Problem *CoinsOnShelf* je NP-težak problem za koji se dokaz težine redukuje na problem *3-Partition*. Osnovna težina problema potiče od činjenice da kada postavimo bilo koja dva diska, oni formiraju prazninu između njih u koju se mogu staviti dodatni diskovi.

Postoji uslov koji skup diskova treba da zadovoljava tako da se između nijedna dva susedna diska ne može postaviti ni jedan drugi disk. Taj uslov je da poluprečnik najvećeg diska ne sme biti više od 4 puta veći od poluprečnika najmanjeg diska. U ovom slučaju, napisani algoritam će dati optimalnu vrednost za veličinu police u slozenosti *O(nlogn)*, a ova slozenost potice od sortiranja diskova po velicini, koji se zatim linearno rasporede. Ovaj slučaj ćemo nadalje nazivati *linearnim*.

Ukoliko prosli uslov nije ispunjen, tada imamo opšti slučaj algoritma za koji je implementiran algoritam koji ne daje optimalno rešenje, ali daje 4/3 aproksimaciju optimalnog rešenja. To znaci da će algoritam u najgorem slučaju dati 4/3 optimalnog rezultata, tj za 1/3 više nego optimalno rešenje. Ovo resenje je takođe dato u složenosti *O(nlogn)*. Ovaj slučaj ćemo nadalje nazivati *opštim*.

U nastavku se biti prikazana shema po kojoj se izvrsava nas algoritam:



Shema izvršavanja algoritma

- Naivni algoritmi -

Kao i u naprednim algoritmima koji su navedeni, naivni algoritmi su implementirani u skladu sa podelom problema na dva podproblema u zavisnosti odnosa najvećeg i najmanjeg diska u skupu. Naivni algoritmi su zasnovani na činjenici da su sve pozicije diskova permutacije brojeva koji bi predstavljali redne brojeve diskova kako smo ih navodili. Cilj je ispitati sve te permutacije I pronaći najbolje rešenje.

Za razliku od naprednog algoritma, naivni algoritam će uvek dati optimalno rešenje. Naravno cena ovog rešenja je višestruko veća u odnosu na napredniji algoritam: složenost potiče od svih permutacija, a to je *O(n!)* za razliku od *O(nlogn)* u naprednijem algoritmu.

- Pokretanje I koričćenje programa -

Program se moze koristiti sa unapred zadatim brojem diskova i njihovim veličinama preko ulazne datoteke, kao I sa pseudoslučajnim odabirom veličina diskova.

Ulazne datoteke moraju biti sledećeg oblika: u prvom redu se nalazi broj diskova koji ce zatim slediti – *N*. Zatim u svakom od *N* redova se nalazi po jedna veličina diska. Za potrebe testiranja i pokretanja programa, kreirano je nekoliko različitih datoteka koje demonstriraju sve podslučajeve problema.

test1.general	Diskovi su proizvoljnih veličina, algoritam radi u opštem slucaju, daje brzu optimizaciju. Naivni algoritam se moze pokrenuti, ali posto imamo 8 diskova u ovom slucaju, potrebno je pregledati 8! = 40320 različitih permutacija.
test2.general	Diskovi su proizvoljnih veličina, ali sada imamo dosta manjih diskova koji se mogu smestiti između vecih. Animacija izgleda vrlo zanimljivo u ovom slučaju. Naivni algoritam je bespotrebno pokretati u ovom slučaju jer postoji 19! permutacija
test3.special.even	Specijalan slučaj kada je odnos poluprecnika najvećeg i najmanjeg diska maksimalno 4. *N* je paran broj.
test4.special.odd	Isti slučaj kao *test3*, samo je ovog puta *N* neparan broj pa se diskovi malo drugačije rasporedjuju, u skladu sa shemom izvršavanja algoritma
test5.naive.special	Ovo je ulazna datoteka nad kojom ima smisla pokretati naivni algoritam, posto imamo ukupno 24 permutacije koje se pregledaju u razumnom roku.

Algoritam se takodje može pokrenuti sa nasumičnim biranjem veličina diskova. Preporučujem da se pokreće za manje brojeve, recimo do 5 ili 6 diskova, kada se može posmatrati razlika izmedju naprednog i naivnog algoritma.

- Vizuelizacija algoritama -

Prilikom izvršavanja naprednog algoritma, slika se ažurira nakon svake postavke diskova, tako možemo da pratimo I samo izvršavanje. U slucaju naivnog algoritma, posto je već navedeno da su u pitanju samo permutacije, iscrtava se samo

krajnji raspored diskova za datu permutaciju, u cilju ustede vremena prilikom prikaza.

- Detalji implementacije -

Radi preglednosti, svi opisi korišćenih struktura podataka, klasa, metoda, funkcija će biti navedeni kao komentari u samom kodu.

- Testovi -

Implementirana su tri testa koji demonstriraju upotrebu Google Test framework-a. Prvi test proverava da li su nakon izvršavanja algoritma, napredna i naivna polica sadrze sve diskove koji su učitani, tj da li su svi diskovi raspoređeni. Drugi test proverava da li su konačne veličine polica jednake, naravno u slučaju kada su odnos poluprečnika najvećeg i najmanjeg diska iznosi maksimalno 4. U opštem slučaju, napredni algoritam daje rešenje koje nije optimalno. Treći test proverava da li su *footprint* tačke diskova u sortiranom poredku, sto je takodje neophodno da bi rešenje bilo validno.

Zbog osobine da napredni algoritam ne daje optimalan rezultat u velikoj većini slučajeva, kao i da naivni algoritam zahteva izuzetno mnogo vremena, raspon testova koji se mogu napisati nije toliko širok koliko se to moze činiti bez zalaženja u detalje.