

## ALGORITMI I STRUKTURE PODATAKA ZA POČETNIKE

Dušan Marjanski<sup>1</sup>, Olivera Hrnjaković<sup>2</sup>, Aleksandar Kupusinac<sup>3</sup>, Rade Doroslovački<sup>4</sup><sup>1,2,3,4</sup>Univerzitet u Novom Sadu, Fakultet tehničkih nauka, Novi Sad, Srbija<sup>1</sup>marjanskid@uns.ac.rs, <sup>2</sup>olja.hrnjakovic@uns.ac.rs, <sup>3</sup>sasak@uns.ac.rs, <sup>4</sup>rade.doroslovacki@uns.ac.rs

**Kratak sadržaj:** Izučavanje informacionih tehnologija predstavlja popularnu i perspektivnu pojavu današnjice. Algoritmi i strukture podataka, neretko, predstavljaju oslonac informacionim tehnologijama i jedni su od prvih pojmova koje je neophodno savladati u procesu učenja određenog programskog jezika. U cilju pisanja kvalitetnog programskog koda, potrebno je postaviti dobre osnove iz oblasti poznavanja struktura podataka. Takođe, važno je steći utisak o zavisnosti koja postoji između algoritama i struktura podataka. U radu će biti reči o prednostima i manama određenih struktura podataka, kao i o algoritmima sortiranja i pretrage. Na kraju će biti predstavljena zavisnost algoritma od strukture podataka i obrnuto.

**Ključne reči:** Osnove programiranja, algoritmi, strukture podataka

## ALGORITHMS AND DATA STRUCTURES FOR BEGINNERS

**Abstract:** Studying information technology is a popular and promising phenomenon nowadays. Algorithms and data structures are often the mainstay of information technology and one of the first concepts that must be mastered in the process of learning a particular programming language. In order to write quality programming code, it's necessary to lay good foundations in the field of data structures knowledge. Also, it's important to get an impression of the dependency that exists between algorithms and data structures. This paper will discuss the advantages and disadvantages of certain data structures, as well as sorting and searching algorithms. Finally, it will present the dependencies of the algorithm on data structure and vice versa.

**Key Words:** Basics of programing, algorithms, data structures

## 1. UVOD

Da bi se stekla osnovna znanja iz oblasti informacionih tehnologija ili, konkretno, nekog programskog jezika, neophodno je, najpre, proučiti pojmove kao što su algoritmi i strukture podataka. U daljem tekstu rada biće objašnjeni pojmovi algoritma i strukture podataka, njihovi predstavnici. Biće, takođe, iznete prednosti i mane određenih algoritama i struktura podataka. Važnost razumevanja algoritama i struktura podataka, njihovih prednosti i mana, ogleda se i u međusobnom odnosu koji oni grade u cilju rešavanja određenog problema.

## 2. ALGORITMI

Algoritam predstavlja postupak za rešavanje nekog problema. Najčešće za problem postoji više načina, odnosno algoritama, koji mogu dovesti do njegovog rešavanja. Algoritam predstavlja skup instrukcija i pravila pomoću kojih je, korak po korak, opisano rešavanje zadatog problema. Iako algoritmi imaju veoma značajnu ulogu u oblasti računarstva, postoji posebna naučna disciplina koja se bavi teorijom algoritama.

Kada je reč o osobinama koje svaki algoritam treba da zadovolji, potrebno je naglasiti da algoritam treba najpre da bude ispravan, odnosno tačan. Pored tačnosti, algoritam mora da zadovolji sledeće uslove: diskretnosti, determinisanost, efektivnost, rezultativnost i generičnost.[1] Diskretnost podrazumeva da se algoritam sastoji od konačnog broja koraka, dok svaki korak može podrazumevati obavljanje jedne ili više operacija. Determinisanost podrazumeva da svaki korak algoritma mora biti jasno i precizno definisan. Takođe, determinisanost podrazumeva da je jednoznačno definisano koji korak u algoritmu sledi nakon trenutnog koraka. Efektivnost je osobina da je vreme izvršavanja algoritma konačno, odnosno da se on završi u prihvatljivo vreme. To znači da se i sve operacije koje čine algoritam moraju se izvršiti za konačno vreme. Rezultativnost podrazumeva osobinu algoritma da posle konačnog broja koraka dovede do traženog rezultata u vidu jedne ili više izlaznih vrednosti. Algoritam može imati više ulaznih podataka, a ne mora niti jedan. Generičnost podrazumeva da algoritam nije namenjen za rešavanje pojedinih slučajeva nekog problema, već više srodnih. Optimalnost je osobina algoritma koja nije podrazumevana, ali je poželjna. Optimalnost je ono što pravi razliku između više algoritama koji rešavaju sličan ili identičan problem. Optimalnost se može ogledati u broju koraka u izvršavanju algoritma, u brzini njegovog izvršavanja, u količini resursa koji se koriste, itd. U nekim slučajevima optimalnost nije ostvariva ili prioritet, dok je u nekim slučajevima od izuzetnog značaja.

## 2.1 ALGORITMI SORTIRANJA

Sortiranje je u oblasti računarstva jedan od problema koji je najrasprostranjeniji. Upravo su iz tog razloga algoritmi sortiranja nešto sa čim se početnici neretko najpre susreću. Zadatak algoritma sortiranja je da posloži elemente u redosledu koji je zadati određenim kriterijumom. Problem sortiranja može se rešiti primenom više različitih algoritama. Neki od njih su intuitivni i jednostavni, dok su drugi kompleksni.[2] Algoritmi za sortiranje koji se najčešće koriste su:

- *bubble sort*,
- *merge sort*,
- *quick sort*,
- *selection sort*,
- *insertion sort* i
- *heap sort*.

### 2.1.1 BUBBLE SORT

Da bi se početnicima pojasnio princip sortiranja, dobra praksa je početi od *bubble sort*-a. Prednost ove metode je u njenoj jednostavnosti - smatra se jednom od najjednostavnijih metoda sortiranja. Ukoliko je potrebno sortirati relativno mali broj elemenata, *bubble sort* može biti dobar izbor. Ukoliko je potrebno sortirati veći broj elemenata, neefikasnost ove metode dolazi do izražaja. Upravo se iz tog razloga ovaj algoritam najčešće koristi u edukativne svrhe.

Ovaj algoritam sortiranja u svakom prolazu kroz niz podrazumeva poređenje susednih elemenata i zamenu njihovih pozicija, ukoliko je to u skladu sa kriterijumom sortiranja. Prolasci kroz elemente niza ponavljaju se sve dok se ne desi prolaz kroz niz u kome nije došlo ni do jedne zamene pozicije susednih elemenata. Navedeni kriterijum završetka algoritma ukazuje na to da je niz elemenata sortirani u skladu sa zadatim kriterijumom. Na slici 1. prikazan je primer u kom se korišćenjem *bubble sort*-a od nesortiranog niza dobija niz sortirani po rastućem kriterijumu.

```
Prvi prolaz:
(6 1 4 3 9) → (1 6 4 3 9), razmena jer je 6 > 1
(1 6 4 3 9) → (1 4 6 3 9), razmena jer je 6 > 4
(1 4 6 3 9) → (1 4 3 6 9), razmena jer je 6 > 3
(1 4 3 6 9) → (1 4 3 6 9)

Drugi prolaz:
(1 4 3 6 9) → (1 4 3 6 9)
(1 4 3 6 9) → (1 3 4 6 9), razmena jer 4 > 3
(1 3 4 6 9) → (1 3 4 6 9)
(1 3 4 6 9) → (1 3 4 6 9)

Treći prolaz:
(1 3 4 6 9) → (1 3 4 6 9)
(1 3 4 6 9) → (1 3 4 6 9)
(1 3 4 6 9) → (1 3 4 6 9)
(1 3 4 6 9) → (1 3 4 6 9)
```

Slika 1. Primer sortiranja niza korišćenjem *bubble sort*-a

## 2.2 ALGORITMI PRETRAGE

Pod pojmom algoritma pretrage podrazumeva se algoritam koji daje rešenje nekog problema nakon što izvrši evaluaciju nad svojim prostorom pretraživanja. Pod pojmom prostor pretraga podrazumeva se skup svih mogućih rešenja problema. Pretraga je proces koji obično ima za cilj prinalaženje elemenata koji zadovoljava određene kriterijume. To je obično element nekog niza, liste, stabla ili neke druge strukture podataka. Algoritme pretrage možemo podeliti na:

- "slepe" (*blind, uninformed search*) i
- "usmerene" (*directed, informed search*).

### 2.2.1 SLEPA PRETRAGA

U algoritmima slepe pretrage ne uzima se u obzir specifična priroda problema. Ovo ujedno predstavlja i prednost i manu ove vrste pretrage. Prednost se ogleda u tome da se ovakvi algoritmi mogu primeniti na širok

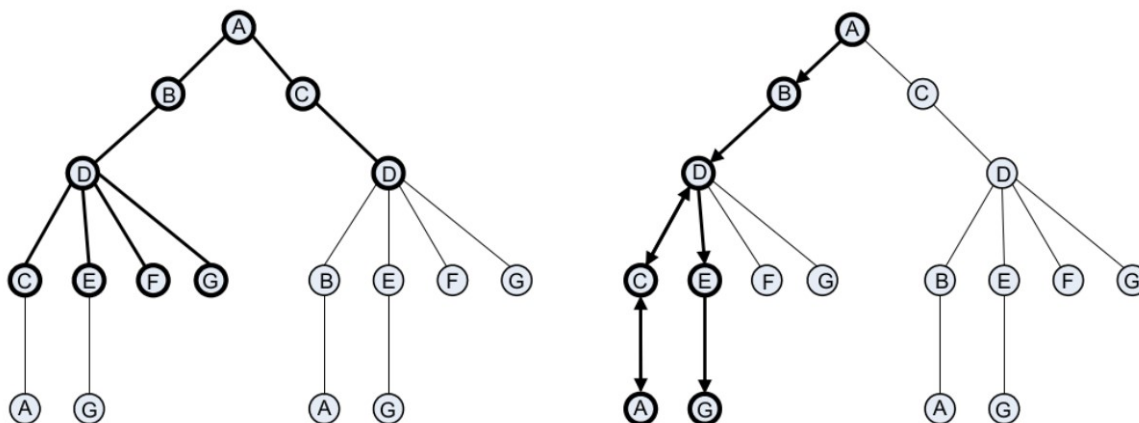
spektar problema. Imajući u vidu veličinu većine prostora za pretragu, može se zaključiti da algoritmi slepe pretrage ne zadovoljavaju kriterijum efikasnosti. Kada su u pitanju primeri koji se koriste u edukativne svrhe, odnosno primeri sa malim prostorom pretrage, opravdano je koristiti algoritme slepe pretrage. Kod ovih algoritama proces pretrage napreduje po nekom unapred definisanom šablonu, sistematski kroz prostor pretrage ili nasumičnim odabirom redosleda evaluacije.[3]

Pretraga liste ili niza pripada grupi najjednostavnijih algoritama pretrage. Linearno pretraživanje, koje spada u ovu grupu algoritama, podrazumeva sekvencijalno ispitivanje jednog po jednog elementa iz liste u potrazi za traženim elementom. Vreme izvršavanja je neretko predugačko i predstavlja manu ovog algoritma. Prednost se ogleda u činjenici da lista nad kojom se vrši pretraga ne mora biti „obrađen“, odnosno unapred sortirana kao što zahtevaju neki drugi algoritmi. Algoritam binarne pretrage moguće je primeniti na unapred sortiranoj listi, a pritom mora biti omogućen i direktan pristup svakom od elemenata liste pojedinačno. Algoritam binarne pretrage započinje od člana koji se nalazi na polovini liste, a u zavisnosti od odnosa u kom su tekući i traženi element pretraga se završava, ukoliko je baš to traženi element, ili nastavlja u odgovarajućoj polovine prostora pretrage, odnosno liste. Na taj način se u svakom koraku smanjuje interval u kom se nalazi potencijalno rešenje.

Algoritmi koji se baziraju na pretrazi strukture tipa stabla čine jezgro algoritama pretrage. Postoje dva načina na koje je moguće vršiti pretragu u okviru strukture tipa stabla:

- pretraga u dubinu (*depth-first search*) i
- pretraga u širinu (*breadth-first search*).

Pretraga u dubinu ima za cilj da dođe nekog lista stabla u što manjem broju koraka. Pretraga u širinu obilazi svaki od čvorova koji se nalaze na tekućoj visini stabla pre nego što pređe na čvorove koji su sledbenici onih na tekućoj visini.



Slika 2. Pretraga strukture stabla po širini i po dubini

### 2.2.2 USMERENA PRETRAGA

Usmerene pretrage, za razliku od slepih, koriste informacije o specifičnoj prirodi problema što može značajno uticati na brzinu pronalaženja rešenja. Informacije o specifičnosti problema nazivaju se heuristikama. Bitno je naglasiti da se heuristike temelje na pravilima i tehikama koje mogu pomoći pri pronalaženju rešenja, ali ne garantuju nužno njegovo pronalaženje. Većina algoritama usmerene pretrage namenjena je, pre svega, za strukture podataka kao što je stablo. Predstavnik ove vrste pretraga, *best-first search*, prolazi kroz stablo pretrage tako što bira najbolje čvorove za nastavak pretrage na osnovu heuristika čvorova.

## 3. STRUKTURE PODATAKA

Pojam strukture podatka predstavlja način organizovanja i čuvanja podataka u računaru tako da se oni mogu efikasno koristiti.[4] Može se reći da je struktura podataka logički model određene organizacije podataka. Osnovne operacije koje se primenjuju nad strukturama podataka su:

- pristup i obrada – pristup svakom elementu strukture da bi se obradio željeni podatak,
- pretraživanje – pronalaženje jednog ili više elemenata strukture koji ispunjavaju zadat uslov,
- dodavanje – pridruživanje novog elementa strukturi i
- brisanje – uklanjanje određenog elementa iz strukture.

Osnovna podela struktura podataka je na linearne i nelinearne strukture. Predstavnicima linearnih struktura podataka su niz, lista, stek i red. U slučaju niza elementi strukture upisani su u susednim memorijskim lokacijama, dok je kod liste linearna veza između elemenata strukture ostvarena posredstvom pokazivača. Predstavnicima nelinearnih struktura podataka su stablo i graf.

Strukture podataka možemo, takođe, podeliti na statičke i dinamičke strukture. Kod statičkih struktura podataka veličina, odnosno memorija koju zauzimaju, je fiksna i unapred se određuje u fazi kompajliranja programa. Prednost statičkih struktura ogleda se brzini njihove obrade. S druge strane, mana je to što se unapred mora definisati koliko će memorijskih lokacija biti rezervisano za potrebe strukture. Kasnije se može ispostaviti da je zauzeto premalo ili previše memorijskih lokacija. Kod dinamičkih struktura podataka veličina se ne određuje u fazi kompajliranja programa već se može menjati u toku njegovog izvršavanja. Prednost dinamičkih struktura ogleda se u optimalnom korišćenju memorije jer se zauzima tačno onoliko memorijskih lokacija koliko je neophodno. Kod dinamičkih struktura potrebno je voditi računa kako o zauzimanju tako i o oslobađanju memorije. Posledica lošeg rukovanja može biti *memory leak*, odnosno „curenje memorije“.

### **3.1 NIZ**

Kada se početnicima objašnjavaju strukture podataka, dobra je praksa početi od niza najpre zbog jednostavnosti ove strukture. Niz predstavlja strukturu podatka čiji se elementi čuvaju na uzastupnim memorijskim lokacijama.[4] Ideja je da se više elemenata istog tipa podatka skladišti zajedno. Time se izračunavanje položaja svakog od elemenata strukture svodi na operaciju dodavanja rednog broja elementa na adresu memorijske lokacije prvog elementa niza. Pored jednodimenzionalnih nizova, postoje i višedimenzionalni nizovi. Predstavnik višedimenzionalnih nizova je matrica, odnosno dvodimenzionalni niz. Prednost nizova, kao linearnih struktura podataka, ogleda se u podudaranju fizičke i logičke reprezentacije strukture. Nedostatak predstavljaju složeni postupci realizacije pojedinih operacija nad strukturom kao što su brisanje ili umetanje elementa na željenoj poziciji.

## **4. ODNOSI ALGORITMA I STRUKTURE PODATKA**

Vrlo često dobar odabir strukture za reprezentaciju podataka omogućava upotrebu efikasnijeg algoritma za njihovu obradu. Treba imati na umu koje se to prednosti određene strukture podataka mogu iskoristiti cilju rešavanja određenog problema. U razvoju složenih programskih rešenja, pokazalo se da je izbor najpogodnijih struktura podataka za zadati problem direktno uticao na kvalitet i performanse rešenja.[3] Nakon što je odabrana struktura, sledi odabir algoritma koji će na najbolji način iskoristiti prednosti te strukture u cilju rešavanja problema. Što je struktura podataka složenija, algoritam za rešavanje problema biće kraći i jednostavniji. Ukoliko je struktura odabrana za rešavanje problema jednostavna, algoritam će svojom složenošću morati da nadomesti sve njene nedostatke.

## **5. ZAKLJUČAK**

Pri savladavanju osnova određenog programskog jezika, algoritmi i strukture podataka imaju značajnu ulogu. Nije dovoljno samo vladati osnovnim pojmovima, već je potrebno detaljnije spoznati prirodu određenog algoritma i strukture podataka, kako zaobići njihove mane, a iskoristiti prednosti. To je osnovni preduslov pri izboru struktura podataka koje će se koristiti u realizaciji određenog rešenja. Da bi se kvalitet rešenja podigao na viši nivo, potrebno je pažnju posvetiti elementima koji rešenje čine efikasnijim. To se može postići analizom odnosa između algoritma i struktura podataka.

## **6. LITERATURA**

- [1] D. Dinu, prezentacije sa predavanja na predmetu Teorija algoritama, Fakultet tehničkih nauka, Novi Sad, 2019.
- [2] P. Janičić, F. Marić, *Osnove programiranja kroz programski jezik C*, Matematički fakultet, Beograd, 2017
- [3] N. Pavković, D. Marjanović, N. Bojčetić, *Programiranje i algoritmi*, skripta II, Zagreb, 2005.
- [4] Online: <https://www.geeksforgeeks.org/data-structures>