**Strukture podataka i algoritmi, deo prvi**

**Koja su tri nacina apstrahovanja podataka?**

* Klasifikacija
* Generalizacija
* Agregacija

**Sta je tip podatka?**

Tip podatka se definise kao skup objekata i operacija nad njima.

**Sta je struktura podatka?**

Struktura podatka je slozeni tip podatka. Nastaju primenom strategije apstrakcije agregacije, tj. Objedinjavanjem vise jednostavnijih tipova podataka.

**Kada se kaze da je neka struktura linearna to znaci da?**

Svakom element odgovara najvise jedan prethodnik i najvise jedan sledbenik.

**Sta je nelinearna struktura?**

Stabla i grafovi su nelinearne structure.

**Kada je data struktura mreza, onda to znaci da?**

Graf (mreza) G predstavlja uredjeni skup (V,E) gde je V skup cvorova, a E skup ivica, u kome svaka ivica predstavlja par (x,y) cvorova iz V.

**Razlika izmedju steaka i reda?**

Stek ubaci i izbaci sa istog kraja. LIFO, Push, Pop, Peek

Red ubaci na jednom, izbaci sa drugog kraja. FIFO, Enqueue, Dequeue

**Ako se redom ubace D,C,B,A u stek, sta ce uraditi metoda izbaci iz steak?**

Izbacice A,B,C,D

**LIFO?**

LIFO Last In First Out

**FIFO?**

FIFO First In First Out

**Pretpostavimo da imamo stek kapaciteta 40, u njemu imamo 10 ubacenih elemenata na pozicijama (niz[0] do niz[9]), sa koje pozicije ce pop metoda izbaciti element?**

niz[9]

**Imamo red implementiran preko niza, kapaciteta 42 elemenata i u njega ubacenih 10 elemenata (niz[0] do niz[9]).**

**Sa koje pozicije ce metoda dequeue() izbaciti element?**

niz[0]

**Na koju poziciju ce metoda enque() ubaciti element?**

niz[10]

**Dat je niz brojeva, implementiran preko reda (2,5,7,8,12) kapaciteta 7. Pokazivaci P na 2 i K na 12. Koji niz komandi daje niz (4,17,7,8,12,34,11) a pokazivaci P na 7 i K na 17?**

1. **ubaci(34), izbaci(),ubaci(11),izbaci(),ubaci(4),ubaci(17)**
2. ubaci(25), izbaci(),ubaci(12),izbaci(),ubaci(9),ubaci(17)
3. ubaci(37), izbaci(),ubaci(17),izbaci(),ubaci(5),ubaci(17)
4. ubaci(65), izbaci(),ubaci(16),izbaci(),ubaci(7),ubaci(17)

Red-Ubaci na kraj-Izbaci sa pocetka!!!

P K

2 5 7 8 12

4 17 7 8 12 34 11

 K P

Ubaci(34) 2 5 7 8 12 **34**

Izbaci() 5 7 8 12 34

Ubaci(11) 5 7 8 12 34 **11**

Izbaci() 7 8 12 34 11

Ubaci(4) 7 8 12 34 11 **4**

Ubaci(17) 7 8 12 34 11 4 **17**

 P K

**Brojevi ulaze redosledom 5,8,2,6,4 koji ce izaci drugi po redu ako primenimo FIFO?**

Drugi po redu izlazi 8.

**Ako je red implementiran preko niza, i ako je s index poslednjeg ubacenog elementa u niz, koja je formula za dobijanje sledeceg indexa za ubacivanje?**

(s+1)%kapacitetNiza

**Linearna struktura je najpogodnija da bi se predstavila:**

1. Organizaciona struktura
2. **Rang lista**
3. Porodicni odnosi
4. Sastav proizvoda

**Data je ciklicna DS lista. Ako je L pokazivac na prvi element, koji pokazuje na poslednji?**

1. L.sledeci.prethodni
2. L.prethodni.prethodni
3. **L.sledeci.prethodni.prethodni**
4. L.prethodni.sledeci

**Dati su elementi DS liste (1,2,3,4,5) sta radi sledeca metoda?**

**X=prvi.sledeci.sledeci;**

**X.sledeci=x.prethodni.sledeci;**

**x.prethodni=x.sledeci.prethodni;**

1. **1 2 4 5**
2. 1 3 4 5
3. 1 2 3 5
4. Nista od navedenog

Trojka treba da nestane.

**Ako imamo ljude koji cekaju red u banci, i radnica na salteru svakog momenta moze da prozove bilo koga ko stoji u redu (ili proziva onoga ko je na redu) da pridje salteru za to je onda najbolje primeniti?**

1. Stek
2. Red
3. **DS lista**
4. Sve je jednako dobro

**Sta predstavlja O(n)?**

Efikasnost/efikasna kompleksnost

**Kada se kaze da algoritam ima vremensku kompleksnost O(n) onda to znaci da?**

1. **Vreme izvrsavanja algoritma je proporcionalno sa n**
2. Vreme izvrsavanja algoritma je manje od n sekundi
3. Algoritam ima n ugnjezdenih petlji
4. Algoritam je n puta sporiji od standardnog algoritma

**Optimalna kompleksnost najboljeg moguceg algoritma za pretrazivanje JS liste koja ima n elemenata je?**

O(1)

**Kolika je (najgora) kompleksnost algoritma za pretrazivanje JS liste?**

Posto je jedino moguca sekvencijalna pretraga kod listi, onda je O(n).

**Kakva je kompleksnost algoritma za pretrazivanje DS liste koja ima n elemenata?**

Posto je jedino moguca sekvencijalna pretraga kod listi, onda je O(n).

**Sekvencijalno pretrazivanje ima prosecnu kompleksnost?**

O((n+1)/2)

**Koliko se koraka izvrsava u slucaju niza od n elemenata kod sekvencijalnog pretrazivanja?**

n+1/2

**Za binarno pretrazivanje je potrebno?**

Za binarno pretrazivanje je potrebno da je niz sortiran.

**Dat je opadajuci niz i ma binarno pretrazivanje, jedna linija koda je boldirana, treba proveriti da li je tacna?**

Ovako je tacno:

If(A[s]>k)

gg=s-1;

else (A[s]<k)

dg=s+1;

**Dat je neki niz celih brojeva i iterativni algoritam za interpolaciono pretraživanje.**

int[] b = { 7, 8, 15, 23, 36 };

public static int interpolaciono(int podatak, int[] n) {

 int l = 0;

 int d = n.length - 1;

 while (l <= d) {

 int index = l + (podatak - n[l])/(n[d] - n[l])\*(d - l);

 if (n[index] == podatak)

 return index;

 if (n[index] > podatak)

 d = index - 1;

 else

 l = index + 1;

 }

 return -1;

 }

**Na kojoj poziciji se nalazi element u nizu kome pristupa algoritam u trećoj iteraciji, ako se traži broj 36?**

1. 3
2. 4
3. 5
4. **nacice broj pre trece iteracije**

**Najgori slucaj kompleksnosti kod interpolacionog pretrazivanja?**

Sa foruma: O(log2n) Skripta: O((log2N))2

**Interpolaciono pretrazivanje u kojoj interaciji pronalazi element?**

O(n)

**Navedite slucaj kada interpolaciono pretrazivanje ima bolje preformanse od binarnog?**

Kada se trazeni element nalazi “slepljen” uz desnu ivicu skupa koji se pretrazuje, jer se tada binarno pretrazivanje degenerise u sekvencijalno.

**Navedite slucaj kada interpolaciono pretrazivanje ima losije preformanse od binarnog?**

Kada raspodela elemenata nije uniformna.

**Dat je niz elemenata** 9 **4** 12 **2** 6 8 18**. Kako ce izgledati niz nakon prvog prolaza kroz niz algoritmom selection sort?**

1. 9 4 12 2 6 8 18
2. 4 9 12 2 6 8 18
3. **2 4 12 9 6 8 18**
4. 2 4 12 9 8 6 18

**Sta je zajednicko kod merge i quick sorta?**

Podeli pa vladaj

**Selection sort i quick sort spadaju u kategoriju?**

Interchange sortiranja

**Kad je u pitanju selection sort nad n elemenata koliko se najvise puta izvrsi premestanje elemenata do kompletnog izvrsavanja algoritma?**

n-1

**Objasni insertion sort?**

Prolazak kroz nesortiran deo i ubacivanje sledeceg elementa u sortiran deo.

**Kod O(n) sta znaci n?**

n je gornja granica

10**n** + 6 = O(**n**)

6**n2**+ 3n + 12 = O(**n2**)

**Koja vrsta liste ce najbrze dati odgovor na pitanje “Koji element je na poziciji n?”?**

Lista implementirana kao niz

**Analiza efikasnosti algoritma**

Najbolji slucaj: 1

Najgori slucaj: n

Prosek: (n+1)/2

**Asimptotska kompleksnost**

P(n) je asimptotski vece od q(n) akko:

lim q(n)/p(n) = 0 za n->beskonacno

**Veliko O notacija (Big O notation)**

Notacija f(n)=(O(g(n)) cita se f(n) je veliko o od g(n) i znaci da je f(n) asimptotski manje ili jednako od g(n). U asimptotskom smislu g(n) je gornja granica za f(n).

10**n** + 6 = O(**n**)

6**n2**+ 3n + 12 = O(**n2**)

**Asimptotska udaljenost**

1 < log n < n < n log n < n2 < n3 < 2n < n!

**Pretrazivanje linearnih struktura**

**Sekvencijalno pretrazivanje**

Porede se elementi redom pocev od pocetka pa do kraja structure.

Najbolji slucaj: O(1) element na pocetku

Najgori slucaj: O(n) element na kraju

Prosek: **O((n+1)/2)**

**Binarno pretrazivanje**

Primenjiv samo u slucaju sortiranih nizova. Ispitati element na sredini i na osnovu toga odrediti deo niza koji treba dalje pretrazivati. Dalje pretrazivanje polovine niza se nastavlja rekuzivno.

Najgori slucaj: O(log n)

Prosek: **O((log n)-1)**

**Interpolaciono pretrazivanje**

Primenjivo samo u slucaju sortiranih nizova. Simulira postupak koji ljudi koriste kod pretrazivanja telefonskog imenika. Binarno sa formulom.

Prosek: **O(log2log2N)** ukoliko su kljucevi uniformno rasporedjeni a ako nisu losije O(N)

Robusno interpolaciono pretrazivanje:

Najgori slucaj: O((log2N))2

Prosek: O(log2(log2N))

**Sortiranje**

**Selection sort**

Sortiranje element po element. Prolazak kroz nesortiran deo i pronalazenje najmanjeg/najveceg elementa. Efikasan na malom broju podataka.

Efikasnost O(n2)

Poredjenja: n2/2

Zamena: n

**Metoda zamene**

Uporedjuje susedne elemente i zamenjuje ih ukoliko nisu u odgovarajucem redu. Bubble i sink sort.

Efikasnost O(n2) naj sporiji od svih algoritama.

Poredjenja: n2/2

Zamena: n2/2 i u proseku i u najboljem slucaju

**Bubble sort**

Pocev od kraja niza uporedjuje susedne elemente i zamenjuje ih ukoliko nisu u odgovarajucem redosledu. Najmanja vrednost isplivava na povrsinu. Mehurici u vodi.

**Sink sort**

Pocev od pocetka niza uporedjuje susedne elemente i zamenjuje ih ukoliko nisu u odgovarajucem redosledu. Najveca vrednost potone na dno.

**Insertion sort**

Prolazak kroz nesortiran deo i ubacivanje sledeceg elementa u sortiran deo. Jednostavan. Efikasan na malom broju podataka.

Kompleksnost O(n2)

Poredjenja: n2/4

Pomeranja: n2/4 u proseku a duplo vise u najgorem slucaju

**Shell sort**

Metoda umetanja (insertion sort) je spor jer se zamene odnose samo na elemente koji su fizicki susedi. Ako je najmanji element na kraju niza potrebno je n koraka da dodje na pocetak. Shellova metoda dozvoljava da se udaljeni elementi zamenjuju. Ideja je da svi elementi koji su medjusobno k udaljeni budu sortirani. Ceo niz se sastoji od k podnizova koji su sortirani, ali su medjusobno izmesani.

**Merge sort**

Pristup “podeli pa vladaj”. Podeli niz na pola u dva niza. Podnizovi se sortiraju na isti nacin rekuzivno. Spoj podnizove u sortirani niz.

Kompleksnost O(n logn)

**Quick sort**

“Podeli pa vladaj” strategija sa sortiranjem u mestu. Podeli niz na dva dela L i D po “pivotu”. Rekuzivno sortirati podnizove L i D.

Poredjenje algoritma za sortiranje:

**O(n2) algoritmi**

* Selection sort
* Bubble sort
* Insertion sort
* Shell sort

**O(n log n) algoritmi**

* Merge sort
* Heap sort
* Qick sort

**O(n log n)** je teoretski najbolja moguća prosecna efikasnost algoritma za sortiranje

|  |  |  |
| --- | --- | --- |
| **Vrsta** | **Prosek** | **Najgori slucaj** |
| **Selection sort** | n2/2 | n2/2 |
| **Buble sort** | n2/2 | n2/2 |
| **Insertion sort** | n2/4 | n2/2 |
| **Shell sort** | / | log2n |

|  |  |  |
| --- | --- | --- |
| **Vrsta** | **Prosek** | **Najgori slucaj** |
| **Merge sort** | n log n | n log n |
| **Heap sort** | n log n | n log n |
| **Quick sort** | n log n | n2/2 |