

ALGORITMI I STRUKTURE PODATAKA

Izvršavanje građiva iz PPL-ja

Pokazivaci

- uvek treba inicijalizirati vrijednost pokazivača prije upotrebe

Razmjena podataka s funkcijom

call by value

call by reference

lokalna zamjena adrese

Reservacija i oslobadanje memorije #include < malloc.h >

malloc - rezervira blok veličine size bajtova u memoriji, i vraća pokazivač na taj blok
(ako blok tražene veličine nije mogao biti rezervisan, vraća NULL pokazivač)

void * malloc(size_t size); - rezervacija memorije

realloc - ako se prije rezervisani blok može povećati na veličinu size, proširi ga
- ako nema mjesto u memoriji, kopira sadržaj starog bloka na novu

lokaciju na kojoj ima mjesto za size bajtova

(ako nigdje u memoriji nema size bajtova slobodnog mjesto, vraća NULL)

void * realloc(void * block, size_t size); - proširene rezervacije memorije

free - oslobada blok memorije ne koji pokazuje pokazivač block

(pokazivač block smije biti samo jedan od pokazivača nastalih

prethodnim pozivima funkcije malloc ili realloc)

void * free(void * block);

ALGORITAM - precizno opisan način rješenja nekog problema

PROCEDURA - po svakome isto što i algoritam, ali ne mora završiti u koničnom broju koraka

PROGRAM - opis algoritma koji u nekom prog. jeziku jednoznačno određuje što računalo treba napraviti

Analiza složenosti algoritama

- Izbor skupova podataka za ispitnu testiranje algoritma:
 - ponašanje u najboljem slučaju
 - ponašanje u najgorem slučaju
 - projektno (tipično) ponašanje
- a priori - trajanje izvođenja algoritma kao vrijednost funkcije nekih relevantnih argumentata
- a posteriori - statistika dobivena ujerenjem na računalu

Složnost algoritma - (O-notacija)

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$$

Tehnike adresiranja

Osnovni pojmovi - kљuc

primarni kљuc - jednoznačno određuje neki zapis

vezani (kompozitni) kљucni - potrebi za jednoznačno određivanje nekih vrsta zapisa

sekundarni kљuc - ve mada jednoznačno određivati zapis, ali ukazuje na neki atribut

Postupci pretraživanja

Slijedno pretraživanje - slijedeće datoteke

pregleđavanje datoteke zapis po zapis (zapisi ne moraju biti sortirani)

projektno se čita u/2 zapis - složnost $O(n)$

Citanje po blokovima - direktne datoteke (ni zapisi su jednakog duljine!)

zapisi sortirani po primarnom kљucu, pa nije uopšte potrebno pregleđavati sve zapise

kad se ustalovi položaj zapisa s traženim kљucem, tada se slijedno pretraži

Binarno pretraživanje - preduvjet: podaci sortirani!

pretraživanje započinje u polovici datoteke | polja, i nastavlja se staknim

raspolavljanjem intervala u kojem bi se mogao naći traženi zapis

složnost je $O(\log_2 n)$

Raspisano adresiranje - hashing

na raspolaganju nam je M pretinaca

iz vrijednosti ključa pomoći hash funkcije izračunava se pseudoslučajni broj iz intervala 0 do M-1

taj broj je adresa grupe podataka (pretinca) koji svi daju isti pseudoslučajni broj

(ako se učki pretinac popuni, može se u vježbi upisati pokazivač na prelijevnu područje ili se prelazi na susjedni pretinac - parčić)

Variabilu su podložni :

Kapacitet pretinca

gustota pakiranja = $N / (M \cdot C)$

N = broj zapisa koje treba postaviti

M = broj pretinaca

C = broj zapisa u jednom pretincu

Memorijski segmenti

TEXT - za postavljanje izvršnog koda i konstanti

DATA - inicijalizirane globalne i statičke lokalne varijable

BSS - neinicijalizirane globalne i statičke lokalne varijable

pomnila (heap) - dinamički alocirane memorija (malloc)

stog (stack) - privremena memorija dok traje izvođenje funkcije

Sistemski stog - VFP

noviji elementi postavljaju se na više memorijske lokacije

postavljanje na stog PUSH, skidanje sa stoga POP

sjedilica koja se postavlja na stog je okvir stoga (stack frame) :

Povratna adresa na koju će treba vratiti ukon izvršuju pozvane funkcije

lokalne varijable funkcije

argumenti (parametri) funkcije

registri procesora (opisivo o prenodiocu i vježbom opisjane)

Rekurzija

procedura poziva samu sebe (mora postojati završetak!)

za pohranjivanje rezultata i povratak iz rekurzije konsti su struktural podataka stog

Uvod u objektno - orijentisano programiranje u programskom jeziku C++

Proceduralna paradigma

uvode funkcije koje su izgradene kao skup naredbi, i koje imaju dobro definisan ulaz i izlaz

Modularno programiranje

procedure evodue funkcionalnosti grupuju se u module koji mogu imati vlastite podatke.

Objektno - orijentirana paradigma

uvodi se pojam članске funkcije!

sintaktično - deklaraciju funkcije stavlja u unutar deklaracije strukture / funkcija postaje "član" strukture

terminološka promjena - element strukture postaje članska varijabla strukture

mena sukoba imena - možemo imati funkcije istog imena, sve dok su one elementi različitih struktura

poziv funkcija koje su "dio" strukture se sintaktično jasno razlikuju od poziva "običnih" funkcija

instanca - strukture . ime-funkcije (parametri)

funkcije koje izjavimo djeleju na podatima u strukturi imaju izravnu vezu s konceptom

strukture (jer su ujen integralni dio)

funkciji se implicitno prenosi pokazivač this, koji pokazuje na varijablu strukture za

koju je članska funkcija pozvana

("lokalna varijabla" za tiju se "deklaracija" i inicijalizaciju brine prevedilac)

:: je operator određivanja dosega - scope resolution operator

("mjeva" vidljivosti identifikatora - varijable, funkcije, klase - u programu)

uvode potvrdi pristupa - klijunske riječi

private - privatnim dijelovima se može pristupiti samo unutar definicije (tjela) članske fun-

public

protected

Razredi (klase)

- ključna riječ class | popravlja pojam strukture iz C-a)
- kod strukture je sve podrazumijevano public, dok je kod razreda sve podrazumijevano private
- struktura predstavlja agregatni skup podataka nad kojima operiraju vanjski elementi programa (funkcije)
- vanjske funkcije nisu dio strukture, te zbog toga struktura nije potpuna
- struktura nije zatvorena - me upravlja sama svojim ponašanjem i stanjem

Dva osnovna elementa OO paradigmе

- apstrakcija - razredi / objekti predstavljaju koncepte iz domene problema koji rješavamo
- enkapsulacija - miti jedan dio sustava me bi smio poslati u unutrašnjim detaljima drugog dijela | što postižemo deklariranjem unutrašnjih detalja razreda kao private)

Pojam objekta

- kod strukture koristimo termin varijabla ili instance strukture
- kod razreda se konkretna instance naziva objektom
- razred je jedan, a u njega se može instancirati proizvoljan broj objekata
- svi ti objekti će biti "isti" u smislu da svi imaju isti skup članskih varijabli i članskih funkcija
- objekti se razlikuju po vrijednostima koje imaju njihove članske varijable | stanje objekta)

Kreiranje objekta

- alocira se prostor u memoriji za članske varijable razreda - dva načina:
 - smještanje objekta na stog - objekt se deklariše kao lokalni objekt unutar funkcije
 - smještanje objekta na gomilu (heap) - operatori new (kreiranje) i delete (brisanje objekta)
("životni vijek" objekta nije vezan uz kontekst izvođenja funkcije, već se objekt eksplicitno mora "uništiti" - izbrisati iz memorije)

Konstruktor objekta / razreda

- posebna članska funkcija namijenjena inicijalizaciji stanja objekta kod njegova kreiranja
- prepoznaće se po imenu funkcije (mora biti isto kao i ime razreda)
- konstruktor nemas parametra po defaultu - me vraca i ne može vratiti nikakav podatak makoje je učitavaju

Podrazumijevani konstruktor

- prevodilac za svaki razred za koji nije eksplicitno definiran konstruktor sam dodaje podr. konstruktor (konstruktor bez parametara koji članove varijable inicijalizira na meke defaultne vrijednosti)

Destruktori razreda

- članska funkcija koja će se pozivati prilikom uništavanja objekta
- ako je u razredu definiran destruktur, prevodilac ga/ automatski poziva u trenutku uništavanja objekta

Curenje memorije (memory leak)

- zauzeli smo resurse racunala, ali ih nismo oslobodili, iako ih više ne koristimo
- prednost destruktora - vrjedi samo za objekte kreirane na stogu (kod heapa se mora pozvati delete!)
(prevodilac će se pobrinuti da se pozove funkcija za uništavanje objekta - stog!)

Copy-constructor

- primać referencu na objekt istog tipa koji će poslužiti kao osnova za kreiranje novog objekta
(referenca = sakenivni pokazivač)

Kopiranje objekata

- deep copy - copy-constructor se implementira tako da kreira u potpunosti novu kopiju objekta
- shallow copy - kreira se novi objekt, ali on nastavlja "dijeliti" određen dio stavljaći objektom na temelju kojeg je nastao

Postupci sortiranja

Sortira je bisanjem (selection sort) $O(n^2)$

- pronađi najmanji element niza, i zamjeni ga s prvim elementom niza
- ponavlja s ostatkom niza, smanjivajući nesortirani dio

Bubble - sort $O(n^2)$

- zamjena susjednih elemenata ako nisu u dobrom redoslijedu :

kroni od početka niza prema kraju

zamjeni dva elementa ako je prvi veći od drugog

Sortira je umetanjem (insertion sort)

- postoji dva dijela niza : sortirani i nesortirani
- u svakom koraku sortirani dio se proširuje tako da se u njega na ispravno mjesto ubaci prvi element iz nesortiranog dijela niza

Shell sort - modificirani sort umetanjem (majstariji biti algoritam) $O(n^2)$

- za k -sortirano polje A vrijedi $A[i] \leq A[i+k]$, $\forall i, i+k$ indeksi
- ako je polje k -sortirano, i dodatno se t -sortira ($t < k$), ostaje i dalje k -sortirano
- potpuno sortirano polje je n -sortirano
- konisti se inkrementalni niz brojeva $h_1, h_2, h_3, \dots, h_t$

Mergesort - "podijeli pa vladaj" uz rekurziju $O(n \log n)$

- nesortirani niz podijeli se na dva niza podjednake veličine
- svaki podniz sortira se rekurzivno, dok se ne dobije niz od jednog elementa
- spoje se dva sortirana podniza u sortirani niz

Quicksort - najbrži do sada, rekursija: "podijeli pa vladaj" $O(n \log n)$

① ako je broj članova polja S jednak 0 ili 1, povratak u pozivnu funkciju

② odabratи broj koji član x u polju S \Rightarrow stožer (pivot)

③ podijeli preostale članove polja S , $S = \{x\}$ u dva odvojenata skupova:

$$S_1 = \{x \in S \setminus \{x\} \mid x \leq x\} \quad \text{sve što je manje od stožera, preseli lijevo}$$

$$S_2 = \{x \in S \setminus \{x\} \mid x \geq x\} \quad \text{sve što je veće od stožera, preseli desno}$$

④ vrati mrežu sastavljen od $\{quicksort(S_1), x, quicksort(S_2)\}$

Indirektno sortiranje

- za sortiranje velikih struktura, nema smisla obavljati mnogo zamjena velikog broja podataka
- podaci se izdvaje u posebno polje s pripadnim pokazivačima na ostale podatke
- sortira se (isto kojim od postupaka) samo takođe izdvajeno polje

Usporedba metoda sortiranja

| maziv | najbolje | prosječno | najgore | stabilan | metoda |
|----------------|---------------|---------------|---------------|----------|----------|
| selection sort | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ | ne | bifanje |
| insertion sort | $O(n)$ | $O(n^2)$ | $O(n^2)$ | da | umetanje |
| bubble sort | $O(n)$ | - | $O(n^2)$ | da | zamjena |
| heap sort | - | - | $O(n^{3/2})$ | ne | umetanje |
| cocktail sort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ | da | spajanje |
| shell sort | $O(n \log n)$ | $O(n \log n)$ | $O(n^2)$ | ne | podjela |
| merge sort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ | ne | bifanje |
| quick sort | $O(n)$ | - | $O(n^2)$ | da | zamjena |
| comb sort | $O(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ | da | zamjena |

stog - LIFO

struktura podataka kod koje se posljednji pohranjeni podatak prvi uzima u obrađu :

dodavanje (push) elemenata na vrh stoga (top)

brisanje (pop) elemenata s vrha stoga

liste - elementi liste su atomi

- linearna lista $A = (a_1, a_2, \dots, a_m)$ je struktura podataka koja se sastoji od uredenog niza elemenata odabiranih iz nekog skupa podataka
- može se realizirati statičkom strukturalnom podatku - poljem
- dinamička podatkovna struktura za realizaciju liste sastoji se od pokazivača na prvi element liste, i od proizvoljnog broja atomi
- svaki se atom sastoji od podatkovnog dijela, i pokazivača na sljedeći element liste
- memorija za svaki atom liste zauzme se u trenutku kad je potrebna za pohranu podatka, a oslobađa se kad se podatak briše
- stog se može realizirati i listom

(umetanje i brisanje iz liste radi da jednom krajem liste koji se maziva vrh stoga)

gomila (heap)

- majprirodni i majčinkoviti macim prikaz prioritetskog reda
- podaci se mogu dodavati (ubacivati) u prioritetski red, te skidati (izbacivati) iz njega
- ne skida se podatak koji je prvi bio dodan, već onaj koji ima/ma veću vrijednost (prioritet)
- gomila je potpuno himalno stablo gdje se čvorovi mogu uspoređivati mekom urednjom/relacijom, i gdje je čvor koji u smislu te relacije veći ili jednak od svoje djece / ako postoji - gomili s relacijom veći od zovemo max heap, a s relacijom manji od min heap

Analiza majstrog slučaja

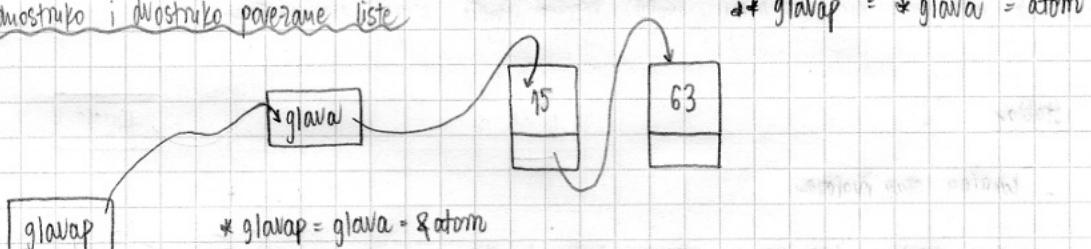
- imamo m elemenata
- na i-toj razini potpunoog binarnog stabla ima majvišće 2^{i-1} čvorova
- na svim nizim razinama do tada ima ukupno $2^{i-1} - 1$ čvorova, za $i > 1$
- stablo s k razinama ima majvišće $2^k - 1$ čvorova
- stablo s $k-1$ razinama ima majvišće $2^{k-1} - 1$ čvorova
- ako je stablo potpuno, započeta je posljednja razina, pa vrijedi: $2^{k-1} - 1 \leq m \leq 2^k - 1$
- za m podataka broj razina je $k = \lceil \log_2(m+1) \rceil$

Sortiranje gomilom (heap sort) $O(m \log_2 m)$

element sa vrha gomile zamjenjuje se s posljednjim elementom polja

gomila se skraćuje za jedan element i podešava

Jednostruko i dvostruko povezane liste



$$glavap = \&glava = \&(2\&atom)$$

glavap sadrži adresu pokazivača na prvi član liste $2(\&atom*)$ ili $\&(2\&atom)$

$*glavap$ sadrži pokazivač na prvi član liste $(atom*)$ ili $(2\&atom)$

$**glavap$ je prvi član liste atom

Brisanje elemenata iz liste

```

int brisi(atom **&glavap, int elem) {
    atom *p;
    for ( ; *glavap && (*glavap->elem) != elem; glavap = &(*glavap->sljed));
    if (*glavap) {
        p = *glavap;
        *glavap = (*glavap)->sljed;
        free(p);
        return 1;
    } else return 0;
}

```

Red (queue) - načelo FIFO (First In First Out)

- linearna lista kod koje se umetanje u listu izvodi na jednomu, a brisanje iz liste na njenom drugom kraju (funkcije: dodaj, skini)
 - koriste se dva indeksa (ulaz i izlaz)
 - kraj reda na kojem se vrši umetanje matrica se ulaz (stariji kraj - rear)
 - drugi kraj je izlaz (prednji kraj - front)
 - učinkovit način realizacije reda statičkom strukturalno je jednodimenzionalno polje zadane podatkovne strukture koje se koristi čekajuće - upotabom operatora modulo (%)
 - jedan element uiza je prazan
 - time se omogućuje razlikovanje punog i praznog reda
- prazan red : $\text{ulaz} = \text{izlaz}$
- pun red : $(\text{ulaz} + 1) \% \text{maxelementa} = \text{izlaz}$
- kad li ujeli mit kao ispunjen, morali moramo imati brojac elemenata

Stablo

- konstrukcija skup čvorova
- postoji poseban čvor koji se naziva korijen (root)
- stupanj čvora je broj podstabala nekog čvora
- korijeni podstabala nekog čvora su dječa tog čvora, a taj čvor nazivamo soditeljem
- skup srodnih čvorova nekog stabla su listovi

Osnovni pojmovi

stupanj stabla - maksimalni stupanj od svih čvorova tog stabla

rasine (level) nekog čvora - korijen je rasine 1, a dječa nekog čvora rasine M su rasine M+1

dužina (depth) stabla - maksimalna rasina nekog čvora u stablu

Binarno stablo - koso stablo, potpuno stablo

stabla koje se sastoji od vijednog, jednog ili više čvorova dugog stupnja

- kod binarnog stabla razlikujemo lijevo i desno podstabla svakog čvora

Obilazak stabla - organizira se da je svaki čvor stabla imao "projeciju"

inorder : lijevi čvor \rightarrow korijen \rightarrow desni čvor

preorder : korijen \rightarrow lijevi čvor \rightarrow desni čvor

postorder : lijevi čvor \rightarrow desni čvor \rightarrow korijen