Univerzitet u Sarajevu
Prirodno-matematički fakultet
ODSJEK ZA MATEMATIKU

A Zmaja od Bosne 33-35, 71 000 Sarajevo, BiH
T +387 33 279 874 F +387 33 649 342
W www.pmf.unsa.ba/matematika
E matematika@pmf.unsa.ba

Strukture podataka i algoritmi - projekat

Rok za slanje projekta je 17.1.2024. do 23:59. Odbrana projekta će biti 18. i 19.1. Tačan termin odbrane (po temama) će biti objavljen 17.1.2024.

Pored traženih stvari u svakoj temi, podrazumijeva se da student (ukoliko je to neophodno za klasu) implementira osnovne stvari da bi klasa bila funkcionalna i upotrebljiva (konstruktor kopije, operator dodjele, destruktor, pomjerajući konstruktor i pomjerajući operator dodjele). Također, potrebno je da student napiše *main* funkciju u kojoj će pokazati kako rade sve tražene funkcionalnosti.

Tema 1 (16 bodova):

Implementirati klasu *Matrica* koja čuva matricu proizvoljnih dimenzija. Potrebno je preklopiti operatore $<<$ i $>>$, pri čemu prvi operator ispisuje matricu na ekran, a drugi omogućava unos objekata tipa *Matrica* sa tastature. Format za unošenje matrice formata 2×3 sa tastature je $[1\ 2\ 3; 4\ 5\ 6]$. Međutim, potrebno je podržati unos i složenijih izraza (ovo je ključna stvar u projektu pored algoritma za množenje), kao npr. $[1\ 2; 3\ 4]\text{^}3\ * (4 * [1\ 2\ 3; 4\ 5\ 6] + [7\ 8; 9\ 1; 2\ 4]\text{^}T) - [1\ 3; 7\ 2]\text{^} - 1 * [7\ 8\ 1; 1\ 2\ 3] * I_3$. Ukoliko izraz nije validan iz nekog razloga (npr. ne slažu se dimenzije matrica), treba baciti izuzetak. Također, imlementirati funkciju koja računa determinantu matrice (ukoliko se radi o kvadratnoj matrici). Za množenje je potrebno implementirati brzi algoritam koji koristi strategiju podijeli pa vladaj.

Tema 2 (16 bodova)

Napraviti generičku strukturu podataka koja čuva elemente kao binarno stablo pretrage. Stablo održava balansiranost na sljedeći način: kada se na visini barem 3 desi da je broj elemenata u jednom podstablu duplo veći od broja elemenata u drugom podstablu, podstablo tog čvora se mijenja. To se radi tako što se od elemenata podstabla napravi sortirani niz i onda se napravi podstablo koje je idealno balansirano. Samo prilikom umetanja i brisanja koji zahtijevaju balansiranje je dozvoljeno vrijeme sporije od $O(\log n)$.

Tema 3 (15 bodova)

U stablo rađeno na vježbama dodati sljedeće funkcionalnosti:

- Iterator i Reverse iterator za kretanje kroz stablo (petlja za prolaženje kroz stablo treba da radi kao što radi i u strukturi set, prolazak kroz čitavo stablo je $O(n)$). Obavezno dodati funkcije Begin() i End() koristeći dva fiktivna čvora (kao što smo radili za listu). Neophodno je omogućiti da radi petlja *for(auto it = s.Begin(); it != s.End(); it++).*
- Prijateljsku funkciju koja provjerava da li je jedno stablo podskup drugog. Funkcija treba da radi u vremenu $O(m + n)$, gdje su m i n brojevi elemenata ta dva stabla;
- Indeksiranje, tako da *Stablo[k]* vraća k −ti po veličini element stabla (indeksiranje kreće od nule). Funkcija treba da radi u vremenu $O(h)$, gdje je h visina stabla.

Tema 4 (15 bodova)

Postoje mnoge metode koje služe za održavanje balansiranosti binarnog stable pretrage (kao što su AVL stabla ili crveno-crna stabla). Jedan od načina je da se simulira slučajni redoslijed umetanja u stablo, odakle onda na

A Zmaja od Bosne 33-35, 71 000 Sarajevo, BiH
T +387 33 279 874 F +387 33 649 342
W www.pmf.unsa.ba/matematika
E matematika@pmf.unsa.ba

osnovu teorije vjerovatnoće slijedi da će stablo biti balansirano sa velikom vjerovatnoćom (tj. visina stabla će biti $O(\log n)$).

Ideja je da elementi stabla pored vrijednosti (koja će u ovom slučaju biti i ključ za pretragu), sadrže i prioritete, koji se slučajno generišu pri umetanju elemenata u stablo. Svaka dva čvora treba da zadovoljavaju osobinu heapa za prioritete, tj. svaki čvor ima veći prioritet od svog djeteta, a manji od svog roditelja.

Operacije koje je potrebno podržati (sve treba da rade u vremenu $O(\log n)$):

- Pretraga – vrši se kao u običnom binarnom stablu, po vrijednosti (koja je ujedno i ključ);
- Umetanje – najprije se element umeće u stablo na klasičan način, kao list, te mu se slučajno dodjeljuje prioritet. Nakon toga, ako nije zadovoljena osobina heapa za prioritete, vrše se neophodne modifikacije (u ovom slučaju rotacije) kako bi se očuvala ta osobina prilikom svakog umetanja.
- Brisanje – ako je dati element list, jednostavno ga obrišemo. Ako ima samo jedno dijete, na njegovo mjesto postavimo to dijete, a njega obrišemo. U slučaju da ima dvoje djece, onda ga zamijenimo sa njegovim sljedbenikom (ili prethodnikom) u sortiranom redoslijedu (po vrijednosti). U posljednjem slučaju, može se desiti da se naruši osobina heapa za prioritete, te će onda trebati izvršiti dodatne rotacije.
- Razdvajanje – funkcija koja prima stablo i neku vrijednost ključa k (za koji se može pretpostaviti da ne postoji u stablu), te vraća par stabala, jedno stablo gdje su svi elementi manji od k, te drugo stablo gdje su svi elementi veći od k. Ideja je da se u postojeće stablo umetne element koji ima vrijednost k i dodijeli mu se prioritet veći od prioriteta svih elemenata u stablu (tj. od prioriteta korijena), tako da će on prilikom umetanja postati korijen. Tada njegovo lijevo i desno podstablo predstavljaju stabla koja treba vratiti. Nije bitno u kakvom stanju nakon funkcije ostaje originalno stablo (tj. nije potrebno da se poziva konstruktor kopije koji bi usporio funkciju).

Ako implementirate do sada pomenute funkcionalnosti možete osvojiti maksimalno 13 bodova. Ukoliko implementirate i naredne funkcionalnosti možete osvojiti maksimalno 15 bodova:

- Pomoćna funkcija - spajanje dva stabla koja su prethodno nastala razdvajanjem – funkcija koja prima dva stabla S_1 i S_2, takva da su svi elementi u S_1 manji od svih elemenata u S_2. Napravi se čvor čija je vrijednost veća od svih elemenata u S_1, a manja od svih elemenata u S_2, te mu se dodijeli prioritet manji od svih prioriteta u stablima S_1 i S_2. Nakon toga, napravi se stablo čiji je korijen novokreirani čvor, te mu je lijevo podstablo S_1, a desno S_2. Nakon toga se izvrše potrebne rotacije da bi se zadržala osobina heapa za prioritete, nakon čega će ovaj element postati list te se može bezbjedno izbrisati.
- Implementirati funkciju koja prima dva stabla i vraća njihovu uniju kao stablo, pri čemu se može pretpostaviti da stabla nemaju zajedničkih elemenata. Funkcija radi na način da se provjeri koji od korijena ova dva stabla ima veći prioritet. Ako je to korijen prvog stabla, onda se funkcija (unija) rekurzivno poziva najprije za lijevo podstablo prvog stabla i lijevo stablo koje nastane razdvajanjem drugog stabla na vrijednosti manje i veće od vrijednosti korijena prvog stabla, a zatim na desno podstablo prvog stabla i desno stablo drugog stabla koje nastane razdvajanjem. Nakon toga se izvrši spajanje dvije novodobijene unije pomoću prethodne funkcije. Prodiskutovati složenost ove funkcije. (Napomena: nije bitno u kakvom su stanju početna stabla nakon završetka funkcije, tj. ne treba da se poziva konstruktor kopije)

Univerzitet u Sarajevu
Prirodno-matematički fakultet
ODSJEK ZA MATEMATIKU

A Zmaja od Bosne 33-35, 71 000 Sarajevo, BiH
T +387 33 279 874 F +387 33 649 342
W www.pmf.unsa.ba/matematika
E matematika@pmf.unsa.ba

Tema 5 (14 bodova)

Polinom $x^{999}+1$ nema smisla čuvati kao vektor koeficijenata, jer bi za čuvanje koeficijenata trebao vektor sa 1000 elemenata. Rad sa ovakvim polinomima bi se sveo na sabiranje i množenje nula. Rješenje je u implementaciji polinoma kao povezane liste (ne koristite tip *list* iz STL). Potrebno je podržati operatore $<<$ i $>>$, pri čemu operator $<<$ ispisuje polinom na ekran (od članova sa manjim stepenima do onih sa većim, npr. $1-x+2x\hat{}4$, u ispisu ne treba da bude znak „ $*$ “), dok operator >> (koji je glavna stvar u ovom projektu) omogućava unos objekata tipa *Polinom* sa tastature, tako da unos $(3x-(-x-1)\hat{}2)(x\hat{}3-2x\hat{}2+1)''$ treba podržati i smjestiti polinom $-6x^3+10x^2-10x+4$ u odgovarajuću varijablu. U slučaju nepravilnog unosa treba baciti izuzetak. Znakovi koji se mogu naći u unosu (pored brojeva i zagrada) su $x, +, -, *, \hat{}, ', \$, dx$ pri čemu stepenovanje ima veći prioritet od množenja. Znak $'$ se može naći samo iza zatvorene zagrade ili iza drugog znaka $'$. Znak $ predstavlja neodređeni integral, te iza njega mora ići otvorena zagrada, dok prije znaka dx mora ići zatvorena zagrada. Kod integrala uzeti da je konstanta uvijek jednaka 0 (npr. $\$(2x+1)dx$ uzmite da je x^2+x, a ne npr. x^2+x+1). U stepenu se može naći samo prirodan broj (tako da npr. $x\hat{}x$ nije validno). Također, izraz tipa $3x$ je validan, ali izraz tipa $x3$ nije (mada $x*3$ jeste, kao i izraz $5*x*3$). Implementirati sve funkcije koje su potrebne da bi se omogućile pomenute funkcionalnosti. U stepenu se može naći samo prirodan broj (tako da npr. $x\hat{}x$ nije validno). Još jedan primjer validnog izraza je $(2x+5)\hat{}3+2x*\$(3x+(5x\hat{}2-3)'+11)dx*x\hat{}2$.

Pored toga, potrebno je implementirati funkciju *double NulaPlinoima(double a, double b, double epsilon)* koja vraća nulu polinoma između a i b, a *epsilon* je dozvoljena greška (možete staviti neku defaultnu vrijednost za *epsilon*, npr. 10^{-5}). Podrazumijevati da su brojevi $P(a)$ i $P(b)$ suprotnog znaka, te koristiti metod bisekcije.

Tema 6 (14 bodova)

Potrebno je razviti klasu *CijeliBroj* koja podržava operacije sa cijelim brojevima proizvoljne veličine (obratite pažnji da brojevi koji mogu stati u tipove *int* ili *long int* nisu proizvoljne veličine, poenta je da ova klasa može služiti za rad sa brojevima koji imaju npr. *10000* cifara). Klasu implementirati kao dvostruko povezanu listu, pri čemu se u pojedinačnim čvorovima nalaze cifre. Pri kreiranju klase ne koristiti tip list iz STL. Klasa treba da ima dva konstruktora, od kojih jedan prima string, a drugi *int*, pri čemu konstruktor koji prima string treba da baci izuzetak ukoliko se ne radi o cijelom broju (npr. ukoliko string sadrži nedozvoljene znakove). Potrebno je preklopiti unarni operator $-$. kao i binarne operatore $+, -, *, +=, -=, *=$. Potrebno je preklopiti operatore $<<$ i $>>$, pri čemu prvi operator ispisuje *CijeliBroj* na ekran, a drugi omogućava unos objekata tipa *CijeliBroj* sa tastature. Pri tome, operator $>>$ omogućava unos proizvoljnog (validnog) izraza u *CijeliBroj*. Na primjer, ako je *broj* varijabla tipa *CijeliBroj*, te ukoliko je unos $(3+2*(8-2)\hat{}2)*4$, naredba $cin>>broj$ treba da u varijablu *broj* smjesti 300 (stepenovanje ima najveći prioritet). Međutim, ono što je ključno, ovaj unos treba omogućiti unos brojeva proizvoljne veličine (a ne samo onih koji mogu stati u *int*). Potrebno je implementirati algoritam za množenje koji koristi strategiju podijeli pa vladaj, a koji dva broja sa n cifara množi u vremenu bržem od kvadratnog (npr. Karatsuba algoritam).

Univerzitet u Sarajevu
Prirodno-matematički fakultet
ODSJEK ZA MATEMATIKU

A Zmaja od Bosne 33-35, 71 000 Sarajevo, BiH
T +387 33 279 874 F +387 33 649 342
W www.pmf.unsa.ba/matematika
E matematika@pmf.unsa.ba

Tema 7 (13 bodova)

Potrebno je implementirati klasu *Imenik* koja predstavlja telefonski imenik. Svaki imenik ima atribut naziv (npr. „PMF osoblje“ ili „Vladislav Skarić roditelji“). Klasu implementirati kao dvostruko povezanu listu, pri čemu čvorovi liste pored pokazivača sadrže i atribut tipa *Kontakt* (struct), a to je struktura koja se sastoji od naziva osobe/firme (tipa *string*), mobilni telefon (tipa string, u obliku „062123456“) i fiksni telefon (također tipa string, u obliku „033222555“). Moguće je da je u kontaktu upisan mobilni telefon, a fiksni nije (tj. fiksni je prazan string), ili obrnuto, ali je neophodno da je upisan bar jedan od njih, kao i naziv. Mobilni i fiksni telefon mogu sadržavati samo cifre (u suprotnom se baca izuzetak). Elemente imenika čuvati u sortiranom poretku po nazivu osobe/firme. Pored stvari koje su neophodne za ispravno funkcionisanje klase (nije neophodno implementirati iteratore, ali nije ni zabranjeno), implementirati sljedeće funkcionalnosti:

- Funkcije za dodavanje novog kontakta, brisanje postojećeg, promjenu naziva osobe/firme i promjenu broja (mobilnog ili fiksnog). Vodite računa da u imeniku kontakti uvijek moraju biti u sortiranom redoslijedu, te da ne smije biti duplikata (tj. dva kontakta s istim imenom). Ako korisnik pokuša dodati kontakt čiji naziv već postoji, baca se izuzetak.
- Pretragu po nazivu osobe/firme, pri čemu funkcija ne mora primiti puni naziv, već je moguće samo dio. Npr. ako funkcija primi string „adm“, treba izlistati sve podatke o kontaktima Administracija, Admir i Admira (naravno, pod uslovom da takvi postoje u Imeniku). Redoslijed kontakata u izlistavanju pretrage je sortiran po nazivu.
- Pretragu po broju telefona (bilo fiksnog, bilo mobilnog). Npr. ako se vrši pretraga po stringu „0626“, funkcija treba da izlista sve kontakte čiji broj (bilo fiksni, bilo mobilni) počinje sa tim stringom. Nemojte podrazumijevati kojim ciframa može počinjati fiksni telefon, a sa kojim mobilni (jer moguće je npr. da korisnik pogrešno unese). Redoslijed kontakata u izlistavanju pretrage je sortiran po nazivu.
- Konstruktor koji prima niz (vektor) kontakata, ali gdje kontakti ne moraju biti u sortiranom redoslijedu. Konstruktor treba da se pobrine za sortiranje kontakata, pri čemu ne smije biti duplikata (a u nizu ih može biti). Također, najprije dodati kontakte u imenik (bez sortiranja), a samo sortiranje je potrebno uraditi direktno na našoj listi (dakle, nije dozvoljeno npr. kontakte staviti u niz, pa ih sortirati, te ih nakon toga vratiti u našu listu). Obavezno je da vrijeme izvršavanja ovog konstruktora bude $O(n \log n)$, gdje je n broj kontakata.
- Funkcija za spajanje (sortiranih) imenika u novi (sortirani) imenik. Ako su imenici veličina p i q, funkcija treba da radi u vremenu $O(p+q)$. U slučaju da u imenicima postoje dva konktakta sa istim nazivima, potrebno je:
  i. ukoliko su kontakti isti (u smislu da imaju isti naziv, mobilni telefon i fiksni telefon), samo jedan od njih treba da bude u novom imeniku (tj. ne smije biti duplikata);
  ii. ukoliko kontakti imaju isti naziv, te oba imaju upisan mobilni broj (dakle, nije prazan string), ali su ti brojevi različiti, tada će u novom imeniku postojati dva kontakta, pri čemu se na kraj naziva dodaje naziv imenika (npr. „Admir – PMF osoblje“ i „Admir – Vladislav Skarić roditelji“). Isto radimo ukoliko oba imaju upisan fiksni broj, ali su ti brojevi različiti.
  iii. ukoliko kontakti imaju isti naziv, te isti mobilni (fiksni) telefon, ali je kod jednog upisan fiksni (mobilni) telefon, a kod drugog nije, tada u novom imeniku treba ostati samo onaj čvor kod kojeg je upisan i mobilni i fiksni telefon.

Univerzitet u Sarajevu
Prirodno-matematički fakultet
ODSJEK ZA MATEMATIKU

A Zmaja od Bosne 33-35, 71 000 Sarajevo, BiH
T +387 33 279 874 F +387 33 649 342
W www.pmf.unsa.ba/matematika
E matematika@pmf.unsa.ba

Tema 8 (13 bodova)

Potrebno je napraviti sljedeće dvije strukture podataka:

- strukturu podataka koja omogućava pristup srednjem elementu po veličini (ako imamo neparan broj elemenata, jasno je šta predstavlja srednji element, a ako imamo $2n$ elemenata, onda uzeti da je to element koji je n. po veličini). Potrebno je implementirati funkcije *push(), top(), pop(),* pri čemu se *top()* i *pop()* odnose na srednji element. Uputa: koristiti dva heapa, od kojih je jedan max heap, a drugi min heap, pri čemu ne treba koristiti priority queue, već heap napravljen "from scratch". Oba heapa implementirati koristeći niz, kao na vježbama.
- strukturu podataka koja na neki način predstavlja dvostruki heap, u smislu da je moguće brisanje i najmanjeg i najvećeg elementa u vremenu O(log n). Taj heap implementirati preko niza kao na vježbama, a osobina ovog heapa je da za svaki element vrijedi da ako je na parnoj dubini onda ja manji od svog roditelja a veći od roditelja svog roditelja, dok za elemente na neparnoj dubini vrijedi da su veći od svojih roditelja, a manji od roditelja svojih roditelja.

Obje strukture podataka treba da budu generičke, te je kod obje potrebno u konstrukturu omogućiti da se pošalje komparator (tj. kriterij poređenja). Po defaultu to treba da bude operator < za taj tip podataka.