Univerzitet u Tuzli

Fakultet elektrotehnike

Uvod u Računarske Algoritme

Zadaća 1

Rekurzija i pretraživanje

25. april 2019.

Napomena

U svim problemima koji slijede nije dozvoljena upotreba komandi i funkcija koje dosad nisu korištene na predavanjima ili vježbama. Dozvoljena je upotreba kontejnera iz standardne biblioteke (*std::vector* i *std::list*), kao i C nizova. Nerekurzivna rješenja se ne mogu smatrati tačnim ukoliko je u zadatku naglašeno da je potrebno koristiti rekurziju.

Zadatak 1

Napisati funkciju ***divide*** koja uz pomoć rekurzije i oduzimanja određuje rezultat cjelobrojnog dijeljenja dva broja.
Funkcija treba da se izvršava u *O(n)* vremenu.

Zadatak 2

Napisati funkciju ***fast_divide*** koja uz pomoć rekurzije određuje količnik dva cijela broja metodom egipatskog dijeljenja. Ova metoda je slična metodi egipatskog množenja. Mala pomoć: tražiti najveći stepen broja 2 koji je manji od broja koji se dijeli.

Zadatak 3

Napisati rekurzivnu funkciju koja provjerava da li je proslijeđeni string palindrom.

Zadatak 4

Napisati rekurzivnu funkciju koja računa sumu svih elemenata u nizu. Funkcija treba da ima sljedeći prototip:

int sum(int* array, int n);

gdje je ***array*** adresa prvog elementa u nizu, a ***n*** ukupan broj elemenata.

Zadatak 5

Izračunati proizvod parnih prirodnih brojeva manjih i jednakih broju ***n*** rekurzivnom funkcijom. Od korisnika se traži unos broja ***n***.

Zadatak 6

Napisati funkciju *push_unique* sa potpisom:

```
bool push_unique(std::vector<int>&, int)
```

Gdje je prvi parametar niz brojeva, a drugi parametar je element kojeg treba ubaciti na kraj niza samo u slučaju da element već nije u nizu. Funkcija treba da se izvršava u O(n) vremenu. Funkcija vraća *true* ukoliko je element ubačen u niz.

Zadatak 7

Implementirati funkciju *sorted_insert* sa potpisom:

```
void sorted_insert(std::vector<int>&, int);
```

Gdje je prvi parametar niz brojeva koji je sortiran, a drugi parametar je element kojeg funkcija treba ubaciti u niz tako da niz ostane sortiran.
Funkcija treba da u O(log n) vremenu pronađe mjesto u nizu gdje treba ubaciti element tako da nakon ubacivanja elementa vektor ostao sortiran. Za ubacivanje elementa na pronađeno mjesto koristiti neki od metoda iz klase std::vector, neovisno od njihove kompleksnosti izvršenja.

Zadatak 8

Implementirati verziju algoritma std::partition koja ima sljedeći potpis:

```
template <typename Iter, typename P>
Iter partition(Iter begin, Iter end, const P& p);
```

Ovaj algoritam prima opseg niza opisan pomoću dva iteratora, uzetih putem template parametra, a koji trebaju zadovoljavati karakteristike *ForwardIteratora*.
Dodatni parametar *p* je predikat funkcija koja prima element niza, a vraća natrag bool argument.
Funkcija treba da izmjeni redoslijed elemenata u nizu na način da se svi elementi za koje je predikat funkcija *p* vratila *true* nalaze ispred elemenata za koje je funkcija vratila *false*.
Međusobni poredak elemenata u ove dvije grupe nije bitan.
Funkcija vraća iterator na prvi element iz druge grupe, za koje je predikat funkcija *p* vratila *false*.

Dva primjera upotrebe partition algoritma:

```
std::vector<int> v{0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
std::partition(v.begin(), v.end(), [](int elem) { return elem & 2 == 0; });
// sadržaj vektora v nakon poziva:
// v = {0, 8, 2, 6, 4, 5, 3, 7, 1, 9};
// funkcija bi vratila iterator koji pokazuje na element 5 (boldiran).


std::vector<int> v{2, 1, 3, 8, 4, 5, 6, 9, 7};
auto s = std::partition(v.begin(), v.end(),
            [](const int& e) -> bool { return e < 4; });
for (auto it = begin(v); it < s; ++it)
      std::cout << *it << ' ';
std::cout << '*';
for (; s < end(v); ++s)
      std::cout << ' ' << *s;
std::cout << std::endl;

// Ispis programa bi bio
// 2 1 3 * 8 4 5 6 9 7
```

Zadatak 9

U prilogu zadaće se nalazi file *shakespeare.txt*. Potrebno je implementirati program koji će učitati sve riječi koje se nalaze u tom fileu, a zatim od korisnika traziti unos jedne po jedne riječi sve do kraja standardnog ulaza. Nakon unosa riječi, program treba da provjeri da li se ta riječ nalazi u fileu i da ispiše jednostavnu poruku korisniku (yes/no) i vrijeme trajanja pretrage za tom riječi.
Implementirati dva rješenja ovog problema:

- prvo rješenje će učitati riječi u kontejner tipa std::set<std::string> (kontejner koji je jako sličan poznatom std::map kontejneru). std::set pruža svoju find metodu kojim se može u log(n) vremenu provjeriti da li je riječ sadržana u kontejneru. Također, std::set će riječi koje se pojave više od jednom spremiti samo jednom.
- drugo rješenje će učitati riječi u kontejner tipa std::vector<std::string>, a zatim koristeći binarno pretraživanje tražiti riječi unutar vektora. Za ovo rješenje je potrebno implementirati algoritam binarnog pretraživanja nad vektorom. Također, potrebno je eliminirati riječi koje se u vektoru jave više od jednom.

Ova dva rješenja je potrebno napraviti u jednom programu gdje će se prvo tražiti kroz std::set, pa ispisati vrijeme izvršenja, a zatim kroz vektor i ispisati vrijeme izvršenja. Za mjerenje vremena koristiti std::chrono zaglavlje.