UNIVERZITET U SARAJEVU
PRIRODNO-MATEMATIČKI FAKULTET

**Dokumentacija za projekat iz predmeta "Analiza i sinteza algoritama"**

**TEMA 4**

Student:

Nejra Šabanović

5780/M

# Sadržaj

Uvod / Objašnjenje problema .....3
Osnovna ideja .....4
CurrencyGraph.....5
addCurrency.....6
getConversion.....7

# Uvod / Objašnjenje problema

Tema **4**, projekta iz analize i sinteze algoritama zahtjeva pravljenje strukture koja će čuvati odnose valute, te izgleda kao uređena trojka {“EUR“,“BAM“,1.95}. Potrebno je omogučiti ubacivanje novih odnosa, te vraćanje traženog odnosa.

Pod pretpostavkom da postoji neki fajl u kojem se nalazi veliki broj ovakvih odnosa, potrebno je misliti na brzinu algortima.

Podrazumijeva se da su odnosi u strukturi konzistentni, tj. nemoguće je da se na različite načine dobiju različiti odnosi (na primjer, ako su dati odnosi USD-EUR i EUR-BAM s jedne strane, i USD-GBP i GBP-BAM s druge strane, kako god da se računa odnos USD-BAM mora se dobiti ista vrijednost).

Također, odnosi su „dvosmjerni“, tj. ako su dati odnosi EUR-BAM i USD-EUR, moguće je računanje odnosa BAM-EUR i BAM-USD. U slučaju da ipak nema dovoljno podataka da se izračuna traženi odnos, program treba vratiti 0.

# Osnovna ideja

Dati problem se može zamisliti kao jedan od problema težinskih grafova, gdje valute predstavljaju čvorove, a njihove vrijednosti razmjene predstavljaju težine po granama. Također, radi lakše i brže pretrage grafa, valute se čuvaju u hash mapi (unordered_map). Na taj način će pretraga odnosa biti pogodnija. Da bi se dobila vrijednost razmjene između dvije valute potrebno je pronaći najbolji "put" između te dvije valute, ukoliko nisu date kao direktno povezane. Algoritam koji računa najpogodije udaljenosti težinskih grafova je Djikstra algoritam. Djikstra daje udaljenost svih čvorova od traženog, stoga u ovom problemu bit će potrebno malo modifikovati algoritam tako da vraća najpogodniji put između dva zadata čvora tj. dvije zadate valute.

# CurrencyGraph

Interfejs klase CurrencyGraph izgleda ovako:

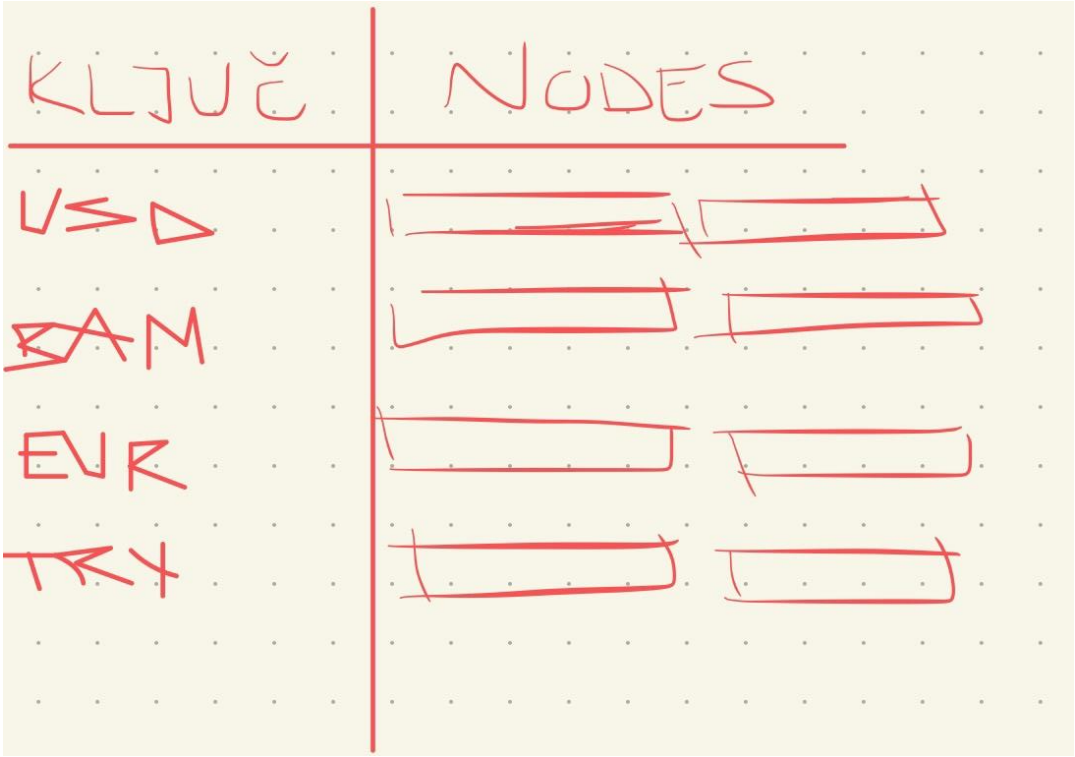
```
class CurrencyGraph{

    struct Node
      string currency;                              //naziv valute
      list<pair<string,double>> connections;        //lista susjedstva
      Node(string name):currency(name){}            //konstruktor
   };

unordered_map<string,Node*> mainCurrencyMap;
void Upper(string& s);
public:
void addCurrency(string firstNode,string secondNode,double rate);
//dodavanje valute
double getConversion(string fromNode,string toNode);     //vraćanje odnosa
};
```

Dvije osnovne funkcije su addCurrency i getConversion, stoga u nastavku će najveći fokus biti na njima.

Ovdje je bitno zamisliti kako je jedna valuta predstavljena. Dakle, kao čvor koji ima ime i svoju listu susjedstva, a lista se sastoji od uređenog para {string,double}. Zatim su svi ti čvorovi sačuvani u mainCurrencyMap.




## addCurrency

Funkcija addCurrency prima dva stringa i vrijednost tipa double, zatim se vrši provjera da li se neka od valuta već nalazi u mapi, narednim uslovom .

```
mainCurrencyMap.find(firstNode) == mainCurrencyMap.end())
```

Ukoliko valuta ne postoji kreira se, a ukoliko postoji samo se vrši dodjela čvoru. I na kraju funkcije izvrši se push_back u listu susjedstva svakog od čvorova.

# getConversion

Osnovna funkcija ovog problema je getConversion koja prima dva stringa i vraća njihovu vrijednost konverzije. Prvo se vrši provjera da li je korisnik unio zahtjev za valute koje postoje u mapi, ukoliko je unešena neka od nepoznatih valuta, program vraća -1.

Zatim, kreira se još jedna mapa `unordered_map<string,double> distances`

koja čuva udaljenosti između čvorova. Razlog čuvanja u mapi je samo brzina.

Pored mape, napravi se i set `set<pair<double,string>> tree`

koji služi prilikom Djikstra algoritma za lakše manipulisanje posjećenih čvorova i korigovanje njihovih težina. Nakon toga se vrše uobičajene provjere za svaki čvor i njegove liste susjedstva, gdje se računaju težine. Dodatno, zbog uslova provjerava se da li je neki od čvorova upravo krajnji čvor, pa se prolazi opet kroz listu susjedstva i traži odgovarajući ključ.

Napomena:

Za izradu projekta poslužile su vježbe u kojima je implementiran DjikstraAlgoritam