Objektno programiranje (C++)

Treća domaća zadaća (ak. god. 2022./2023.)

Datum objave na Merlinu: četvrtak, 13. travnja 2023. od 20:00h	Rok za predaju preko Merlina: utorak, 25. travnja 2023. do 20:00h
Ukupan broj zadataka: **1 zadatak**	Ukupno moguće ostvariti bodova: **10 bodova**

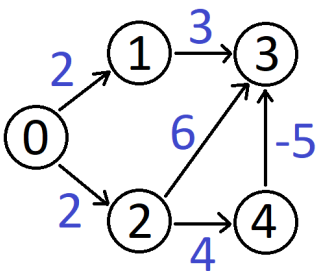
Zadatak 1. *(10 bodova)* U ovome zadatku potrebno je implementirati klasu `Graf` koja ima implementiranu cjelokupnu kontrolu kopiranja (za to je predviđeno ukupno 5 od 10 bodova na ovome zadatku), te pomoću koje je moguće odrediti optimalan put između zadanih vrhova grafa pomoću Bellman-Fordovog algoritma (za to je predviđeno preostalih 5 od 10 bodova na ovome zadatku).

Klasa `Graf` sprema podatke o usmjerenom težinskom grafu (svakom bridu pridružen je cijeli broj (tipa `int`) koji predstavlja njegovu težinu (ili cijenu)). Nema višestrukih bridova (ali su moguće petlje). Klasa `Graf` ima sljedeća tri konstruktora (pri čemu to nisu nužno jedini koje će naposlijetku imati): konstruktor bez argumenata stvara prazan graf (bez vrhova i bez bridova), jedan koji prima nenegativan broj `n` te stvara graf s `n` vrhova (i bez bridova), te konstruktor koji iz datoteke učitava podatke o grafu na sljedeći način: u prvom retku datoteke naveden je (prirodan) broj vrhova, a u svakom od preostalih redaka datoteke nalaze se podaci u formatu:

```
početni_vrh završni_vrh cijena_brida
```

Primjer jedne takve datoteke naziva `"graf.txt"` (grafički prikaz tog usmjerenog težinskog grafa nalazi se na slici desno):

```
5
0 1 2
0 2 2
1 3 3
2 3 6
2 4 4
4 3 -5
```



Ako je broj vrhova u grafu `n`, tada su oni poistovjećeni s brojevima `0`, `1`, ..., `n-1` (npr. za `n = 5`, imamo vrhove `0`, `1`, `2`, `3` i `4`). Za zadane vrhove `a` i `b`, korištenjem **Bellman-Fordovog algoritma** (informacije dostupne, primjerice, na Wikipedii, iako nije zabranjeno koristiti i drugu literaturu - u tom slučaju potrebno je bilo navesti dodatnu literaturu

u polju za tekst uz ovaj zadatak na Merlinu), potrebno je (za `Graf G` to dobivamo pozivom `G.optimalni(a,b)`):

- ukoliko ne postoji put između vrhova `a` i `b`, ispisati prikladnu poruku,
- inače, ukoliko postoji negativan ciklus u grafu (u kojeg je moguće doći iz vrha `a` - takav ciklus detektirat će upravo Bellman-Fordov algoritam), ispisati prikladnu poruku,
- inače, ispisati cijenu optimalnog puta između vrhova `a` i `b` te vrhove koji se nalaze na tom putu (u obratnom redoslijedu, tj. od vrha `b` do vrha `a`). Optimalan put između vrhova `a` i `b` je onaj usmjereni put od vrha `a` do vrha `b` takav da je zbroj težina bridova od kojih se sastoji minimalan. U slučaju da je više optimalnih puteva između vrhova `a` i `b`, nije nam važno koji se od njih ispiše.

Implementirajte klasu `Graf`. Uz ovaj zadatak bilo je potrebno na Merlinu priložiti dvije datoteke: `Graf.h` i `Graf.cpp`. U tim datotekama mora biti u potpunosti razdvojeno sučelje od implementacije. Datoteka `Graf.h` mora izgledati ovako (naravno, potrebno je dopuniti dijelove koji nedostaju):

```
#ifndef GRAF_H
#define GRAF_H

...potrebni includeovi...

class Graf {
    public:
        typedef int vrh;
        Graf();
        Graf(int);
        Graf(std::ifstream&);
        ...
    private:
        ...
};
#endif
```

`main` funkcija za testiranje nalazi se na posljednjoj stranici ovog dokumenta. U toj funkciji uočit ćete zakomentirane dijelove. Ti dijelovi se koriste pri testiranju implementacije kontrole kopiranja:

- drugi dio zadatka je implementacija **kontrole kopiranja**. Klasa `Graf` mora sama sve svoje resurse dinamički alocirati, osim ako se radi o jednom `int` podatku. Primjerice, u redu je da klasa ima `int br_vrhova;` no ne može imati, primjerice,

`map<int,int> bridovi;` (ali može imati `int** bridovi`). Potrebno je implementirati: *copy* konstruktor, operator pridruživanja kopiranjem, *move* konstruktor, konstruktor pridruživanja premještanjem i destruktor. Sve navedene kontrole moraju ispravno obavljati svoje predviđene funkcije (npr. destruktor oslobađa zauzete resurse).

Za ranije spomenutu datoteku `"graf.txt"`, te uz zakomentiran dio funkcije `main()`, dobivamo sljedeći ispis, pri čemu `G.ispis();` ispisuje graf tako da za svaki brid imamo po jednu liniju oblika:

`(početni_vrh,završni_vrh) = težina_brida`

dok `G.ima_vrh(a)` vraća `bool` podatak - `true` ako vrh `a` pripada grafu `G`, inače `false`:

- za unos početnog vrha `2` i završnog vrha `3`:

```
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ g++ -Wall Graf.cpp main.cpp -std=c++11 -o prog
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ ./prog
Unesite naziv datoteke (bez .txt): graf
Unesite pocetni vrh:
2
Unesite zavrsni vrh:
3
Ispis grafa:
(0,1) = 2
(0,2) = 2
(1,3) = 3
(2,3) = 6
(2,4) = 4
(4,3) = -5
Cijena optimalnog puta od vrha 2 do vrha 2 je 0.
Ispis optimalnog puta (unatrag):
2
Cijena optimalnog puta od vrha 2 do vrha 3 je -1.
Ispis optimalnog puta (unatrag):
3 4 2
```

- za unos početnog vrha `4` i završnog vrha `0`:

```
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ g++ -Wall Graf.cpp main.cpp -std=c++11 -o prog
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ ./prog
Unesite naziv datoteke (bez .txt): graf
Unesite pocetni vrh:
4
Unesite zavrsni vrh:
0
Ispis grafa:
(0,1) = 2
(0,2) = 2
(1,3) = 3
(2,3) = 6
(2,4) = 4
(4,3) = -5
Cijena optimalnog puta od vrha 4 do vrha 4 je 0.
Ispis optimalnog puta (unatrag):
4
Nema puta od vrha 4 do vrha 0.
```

- ukoliko u "graf.txt" redak 2 3 6 zamijenimo s 3 2 -6 te unesemo kao početni vrh 2 i završni vrh 3:

```
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ g++ -Wall Graf.cpp main.cpp -std=c++11 -o prog
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ ./prog
Unesite naziv datoteke (bez .txt): graf
Unesite pocetni vrh:
2
Unesite zavrsni vrh:
3
Ispis grafa:
(0,1) = 2
(0,2) = 2
(1,3) = 3
(2,4) = 4
(3,2) = -6
(4,3) = -5
Postoji negativan ciklus!
Postoji negativan ciklus!
```

- s promjenom kao u prethodnoj točki, ali za početni vrh 1 i završni vrh 0:

```
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ g++ -Wall Graf.cpp main.cpp -std=c++11 -o prog
sehorva@DESKTOP-S92RB8H:~/OPCPP/dz3$ ./prog
Unesite naziv datoteke (bez .txt): graf
Unesite pocetni vrh:
1
Unesite zavrsni vrh:
0
Ispis grafa:
(0,1) = 2
(0,2) = 2
(1,3) = 3
(2,4) = 4
(3,2) = -6
(4,3) = -5
Postoji negativan ciklus!
Nema puta od vrha 1 do vrha 0.
```

Na gornjim slikama uočite upotrebu opcije -Wall jer želimo paziti na upozorenja poput neiskorištene varijable i slično - takva upozorenja se ne smiju javljati.

```
#include <iostream>
#include <fstream>
#include <string>
#include "Graf.h"
using namespace std;

/* Graf f(const Graf& G) {
      Graf H = G;
      return H;
} */

int main() {
      string naziv;
      cout << "Unesite naziv datoteke (bez .txt): ";
      cin >> naziv;
      ifstream dat(naziv + ".txt");
      if(!dat) {
            cout << "Ne mogu otvoriti datoteku "
                 << naziv << ".txt!" << endl;
            return -1;
      }
      Graf G(dat);
      /* Graf J;
      const Graf H(G);
      J = J = H;
      G = f(J); */
      Graf::vrh pocetni, zavrsni;
      cout << "Unesite pocetni vrh: " << endl;
      cin >> pocetni;
      cout << "Unesite zavrsni vrh: " << endl;
      cin >> zavrsni;
      if(!G.ima_vrh(pocetni) || !G.ima_vrh(zavrsni))
            cout << "Neki od unesenih vrhova"
                    " nije u grafu!" << endl;
      else {
            cout << "Ispis grafa:" << endl;
            G.ispis().optimalni(pocetni,pocetni)
             .optimalni(pocetni,zavrsni);
            /* H.ispis().optimalni(pocetni,pocetni)
             .optimalni(pocetni,zavrsni); */
      }

      return 0;
}
```

□