Algoritmi in podatkovne strukture – 2
Drugi kolokvij (2013/14)

Kolokvij morate pisati posamič. Pri reševanju je literatura dovoljena. Pri odgovarjanju bodi natančni in: (i) odgovarjajte *na zastavljena* vprašanja; in (ii) odgovorite na *vsa* zastavljena vprašanja.

Čas pisanja izpita je 60 minut.

Veliko uspeha!

NALOGA	TOČK	OD TOČK	NALOGA	TOČK	OD TOČK
1			3		
2			4		

IME IN PRIIMEK: ______________________________

ŠTUDENTSKA ŠTEVILKA: ______________________________

DATUM: ______________________________

PODPIS: ______________________________

1. naloga: Na predavanjih smo z metodo *Monte Carlo* izračunali število π s pomočjo ploščine kroga. Poleg tega smo spoznali še metodo Las Vegas in kako povzporejamo takšne algoritme.

VPRAŠANJA:

1. Podrobno *opišite in utemeljite* postopek, kako bi z metodo *Monte Carlo* izračunali π s pomočjo volumna krogle.

2. Natančnost metode *Monte Carlo* definira formula $O(\sqrt{1/N})$, kjer je N število iteracij. Kako se ta formula spremeni, če uporabimo p procesorjev, od katerih vsak izvede N iteracij. Utemeljite odgovor.

3. Na predavanjih smo tudi izvedeli, da je hitro urejanje (*QuickSort*) primer Las Vegas algoritma. Na predavanjih je bila tudi opisana splošna shema za povzporejanje algoritmov Las Vegas. Predlagajte, kako bi povzporedili hitro urejanje s p procesorji. Rešitev:

   ```
   HitroUrejanje(polje A):
     (A1, A2) := Razdeli(A)
     vzporedno (HitroUrejanje(A1), HitroUrejanje(A2))
   ```

   ne šteje.

2. naloga: Tokrat so Petra poklicali organizatorji njujorškega maratona, ki imajo posebno željo. Namreč želijo postaviti spletno stran, preko katere bi lahko pregledovali trenutni vrstni red tekačev. Od Petra želijo, da njegova rešitev podpira čim učinkoviteje naslednje funkcije:

- `RaceStart()` – tekma se je pričela;
- `RaceEnded()` – tekma se je zaključila;
- `CurrentTime(who, time)` – ki osebi `who` popravi trenutni čas[1]; in
- `Place(who)` – ki vrne trenutno mesto tekmovalca `who`.

Seveda, spletna stran mora delovati tudi še po zaključku tekme.

VPRAŠANJA:

1. Za izračun ranga in za izbiro smo uporabili razširjene podatkovne strukture. Opišite, kako se spremenijo vrednosti v vozliščih pri enojnem in pri dvojnem vrtenju AVL drevesa. Narišite strukturo pred vstavljanjem, po vstavljanju pred vrtenjem in še po vrtenju.

[1] Lahko predpostavite, da sta `who` in `time` celi števili – štartna številka in čas v sekundah.

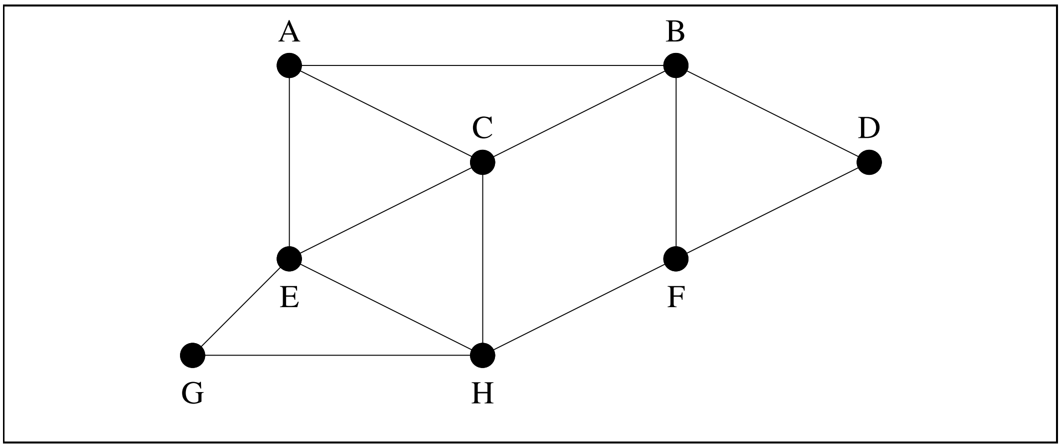
2. Opišite podatkovno strukturo, ki bo rešila zgoraj opisani problem in podajte opis ali psevdokodo posamezne operacije. Učinkovitejša bo vaša rešitev, več točk boste dobili.

3. Kakšna je časovna zahtevnost posamezne operacije? Utemeljite odgoovor.

3. naloga: Imamo graf $G(V, E)$ in v njem vozlišče s. Poleg tema imamo algoritem, ki ga poženemo na grafu G, kjer je `Queue` navadna vrsta:

```
Poisci(G, s)
  Queue q;
  V[*]= not visited;   // označimo vsa vozlišča za neobiskana
  q.Enqueue( (s, 0) );
  while NOT q.Empty()
    (u, d)= q.Dequeue();
    V[u]= visited;
    for v= all neighbours of u do  // vse sosede vozlišča u obdelamo
      if V[v] == not visited then
        q.Enqueue( (v, d+1) );
  println( u, d )
```

Poleg tega imamo graf G kot je definiran na sl. 1.



Slika 1: Neutežen graf $G(V, E)$.

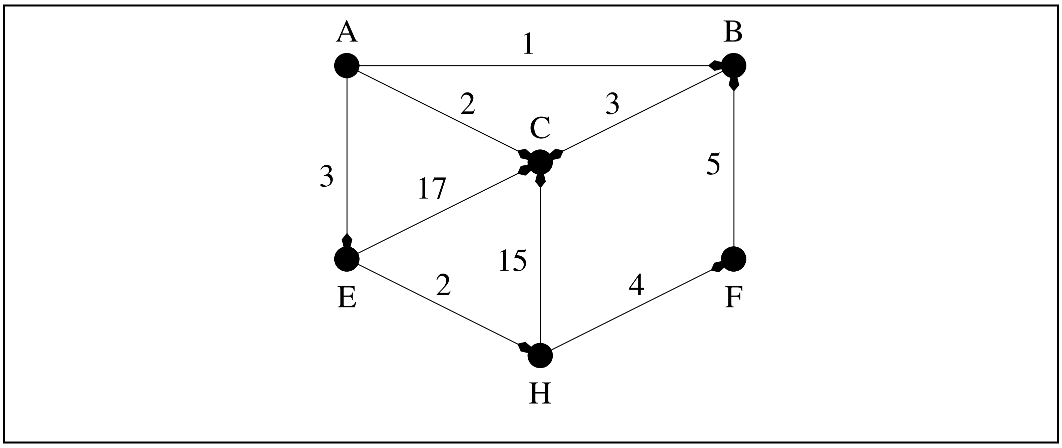
VPRAŠANJA:

1. Prikažite izvajanje algoritma `Poisci(G, C)`, pri čemer izpisujte vsebino uporabljenih podatkovnih struktur. Kaj izpiše algoritem na koncu?

2. Naj bo graf definiran kot $G(V, E)$, kjer je $|V| = n$ in $|E| = m$. Kakšna je časovna zahtevnost zgornjega algoritma? Utemeljite odgovor.

3. Kaj v resnici poišče zgornji algoritem – kaj izpiše? Utemeljite odgovor.

4. naloga: V tej nalogi se bomo poukvarjali z iskanjem najkrajše poti med enim vozliščem ter vsemi ostalimi vozlišči. Kot primer si oglejmo graf na sl. 2.



Slika 2: Primer grafa.

VPRAŠANJA:

1. S pomočjo Dijkstrovega algoritma v grafu na sl. 2 poiščite najkrajše poti od vozlišča A do vseh ostalih vozlišč. Rešitev naj nazorno prikazuje, kako se spreminjajo vrednosti v podatkovnih strukturah.

2. Naš nesrečni Peter Zmeda si je zapomnil, da Dijkstrov algoritem ne deluje pravilno, če so v grafu povezave z negativno vrednostjo. Dokažite to trditev.

   NAMIG: Naredite graf, za katerega Dijkstrov algoritem ne deluje pravilno.

3. Peter je prišel na preprosto idejo, da bi v grafu $G(V, E)$ ($|V| = n, |E| = m$):

   (a) poiskal povezavo, ki ima najbolj negativno utež w_m;
   (b) utežem vseh povezav prištel $-w_m$, da dobi nov graf $G'(V, E')$, kjer imajo vse povezave nenegativno utež;
   (c) na grafu G' pognal Dijkstrov algoritem in

   dobil najkrajše povezave od enega vozlišča do vseh ostalih vozlišč v izvornem grafu G. (i) Analizirajte kakšna je časovna zahtevnost Petrovega algoritma; in (ii) s protiprimerom pokažite, da ne deluje pravilno.