Vážení študenti predmetu Algoritmická teória grafov.

Podľa príkazu rektora 5/2020 sa prerušuje prezenčná výučba do 22.3.2020. Pedagogický proces bude prebiehať prostredníctvom samoštúdia, dištančnou alebo inou vhodnou formou.

To znamená, že nemáte prázdniny, ale stále ste vo vyučovacom procese. V čase prerušenia výučby prídete o 4 hodiny prednášok a 4 hodiny cvičení – spolu o 8 hodín. Ďalších 8 hodín by ste na predmete mali pracovať samostatne. Týchto 16 hodín nariaďujem využiť takto.

Zopakujete si prednesenú časť látky „Cestu v grafoch " (strany 1-26 podľa číslovania v slajdoch, 1-84 podľa číslovania AcrobatReader) –zvlášť Tarryho a základný algoritmus a chýbajúce prednášky nahradíte samoštúdiom nasledujúcich tém a algoritmov (v slajdoch od strany 27, podľa AcrobatReader do str. 85 až do konca):

1. Dijkstrov algoritmus
2. Floydov algoritmus
3 Label set a label correct algortus
4. Hľadanie cyklu zápornej ceny v digrafe
5. Hľadanie cesty maximálnej priepustnosti

Všetko, čo potrebujete k naštudovaniu spomenutých tém nájdete v slajdoch k prednáške
https://frcatel.fri.uniza.sk/users/paluch/Prezentacie/GrafPrez_03.pdf

Môžete použiť aj podrobnejšie učebné texty, ktoré nájdete tu
https://frcatel.fri.uniza.sk/users/paluch/grafy.pdf ,
kapitola 3, časti 3.5 -- 3.9, kde navyše nájdete v časti 3.10 aj zaujímavé aplikácie.

V rámci nahradenia cvičení:

1. Stiahnuť súbor priklad1.zip zo stránky https://frcatel.fri.uniza.sk/users/paluch/zadanie1.html
a z neho uložiť na disk súbor pr1.hrn. Jeho štruktúra je popísaná na tejto stránke.

2. Napísať program na načítanie súboru pr1.hrn.
Program písať pre ľubovoľný súbor so štruktúrou rovnakou, ako má pr1.hrn.
2.1 Zistiť počet hrán m – t.j. počet riadkov v súbore pr1.hrn tak, že riadky čítame postupne ako reťazce (string), počítame počet načítaných stringov pokiaľ nenarazíme na koniec súboru. Načítané stringy neukladáme.

2.2 Načítať hrany zo súboru pr1.hrn do poľa H[ ][ ] o 3 stĺpcoch a m+1 riadkoch.
H[i][0] bude začiatočný vrchol orientovanej hrany i,
H[i][1] koncový vrchol hrany i
H[i][2] cena hrany i
Hrany číslujte od indexu 1, H[0][0] H[0][1] H[0][2] riadok ostane prázdny (index i=0 predstavuje neexistujúcu hranu).
Počet vrcholov n zistíte ako maximum čísel nultého stĺpca poľa H[ ][ ].
2.3 Skontrolujte, či je pole H[ ][ ] zotriedené neklesajúco podľa stĺpca 0, a ak nie, zotrieďte ho.
2.4 Podľa stránky 41 podľa číslovania slajdov (resp. 48 podľa číslovania Acrobat Reader) súboru
https://frcatel.fri.uniza.sk/users/paluch/Prezentacie/GrafPrez_01.pdf
zostrojte pole S[ ], kde S[i] je index prvého riadku poľa H[ ][ ],
v ktorom sa vyskytuje v stĺpci 0 vrchol i.

3. Naprogramujte Základný algoritmus na hľadanie najkratších orientovaných ciest z daného vrchola u do všetkých dosiahnuteľných vrcholov.
Aplikujte na digraf zadaný súborom pr1.hrn.

Pre zadané vrcholy u, v z terminálu vypíšte najkrašiu (u,v)-cestu.

Námet, ako to urobiť nájdete na slajde 26 (strana 85 podľa číslovania Acrobet Reader) v súbore
https://frcatel.fri.uniza.sk/users/paluch/Prezentacie/GrafPrez_03.pdf

4. Naprogramujte Floydov algoritmus

4.1 Zostrojte maticu C[ ][ ] rozmeru (n+1)x(n+1) a maticu X[ ][ ] rozmeru (n+1)x(n+1) podľa slajdu 33 str. 119 podľa číslovanie Acrobat Reader súboru
https://frcatel.fri.uniza.sk/users/paluch/Prezentacie/GrafPrez_03.pdf
Nultý stĺpec a nultý riadok poľa C[ ][ ] nechajte prázdne.
POZOR. Nekonečno voľte ako polovicu maximálneho zobraziteľného celého čisla, aby vám pri sčitovaní dvoch nekonečien nedošlo k pretečeniu.

4.2 Naprogramujte samotný Floydov algoritmus podľa tohoto návodu

```
for (k = 1; k<= n; k++) {
   for (i := 1; i<= n; i++) {
      for {j := 1; j<= n; j++ {
         if (C[i, j] > C[i, k] + C[k, j]) {  C[i, j] := C[i, k] + C[k, j];
                                                X[i, j] := X[k, j];
                                            }
      }// koniec cyklu for j
   }// koniec cyklu for i
}// koniec cyklu for k
```

4.3 Skúste preskočiť cyklus „**for** j“, ak C[i, k]= nekonečno. Podobne skúste preskočiť „**if**“ v cykle „**for** j“ , ak C[k,j]= nekonečno.

5. Naprogramujte Label Correct a potom aj Label Set algoritmus.

Množinu Ɛ implementujte ako pole rozmeru (n+1), spolu s poľom Z[ ] tiež rozmeru (n+1).
E[0] a Z[0] nevyužívajte.
Počet prvkov množiny Ɛ označte nE.
Prvky množiny Ɛ budú E[1], E[2], … E[ne]. Pole Z bude obsahovať informáciu, či vrchol v je v množine Ɛ.
Z[v] = 1 práve vtedy, keď v ϵ Ɛ, t.j. ak vrchol v je v množine Ɛ.

Pridanie vrchola v do množiny Ɛ sa urobí takto

```
if  (Z[v]=0) { Z[v] = 1;
               nE = nE+1;
               E[nE] = v; }
```

Vybratie ľubovoľného vrchola z množiny Ɛ sa urobí napríklad takto z konca

```
v = E[nE];
Z[v] = 0;
nE = nE - 1
```

alebo takto zo začiatku

```
v = E[1];
Z[v] = 0;
E[1] = E[nE]
nE = nE - 1
```

Vybratie vrchola s minimálnou hodnotou značky t[ ] sa urobí takto:

```
min = nekonečno;
for ( i = 1; I <= nE; i++){
   if (min > t[i]) {min = t[i];  imin = i;}
  }
 v = E[imin];
  Z[v] = 0;
  E[imin] = E[nE];
   nE = nE - 1 ;
```

Nasledujúcu časť LabelSet resp. LabelCorrect algoritmu:

”Pre všetky hrany (r, j) ϵ $H^+(r)$ urob:
Ak t(j) > t(r) + c(r, j), potom t(j) := t(r) + c(r, j), x(j) := r.”

možno implementovať nasledovne

```
for(i=S[r]; i<S[r+1]; i++) {
    j=H[i][1];
   if ( t[j] >  t[r] + H[i][2] ) {
                        t[j] = t[r] + H[i][2];
                        x[j] = r;}
 }
```

6. Skúste porovnať rýchlosti Základného, LabelCorrect a LabelSet algoritmov

7. Vypočítajte maticu vzdialenosti algoritmami z bodu 6 po riadkoch a porovnajte s rýchlosťou výpočtu Floydoým algoritmom.

Tarryho ani Dijkstrov algoritmus zatiaľ neprogramujte.

Výsledky vašej práce skontroluju cvičiaci na najbližších cvičeniach.

Konzultácie plánujem na pondelok a štvrtky od 08:00 do 10:00, alebo kedykoľvek po mailovej dohode.