


Drumuri minime

+ Grafuri bipartite. Grafuri planare

 Ca și în laboratorul trecut, fișierul `grafpond.in` are următoarea structură: numărul de vârfuri n , numărul de muchii/arce m și lista muchiilor/arcilor cu costul lor (o muchie fiind dată prin extremitățile sale și cost). Costul unei muchii este număr natural.

grafpond.in		
5	7	
1	4	1
1	3	5
1	2	10
2	3	2
4	2	6
4	5	12
5	2	11

Justificați complexitatea+corectitudinea algoritmilor propuși.

1. **Graf bipartit.** Se citesc din fișierul `graf.in` următoarele informații despre un graf neorientat (neponderat): numărul de vârfuri n , numărul de muchii m și lista muchiilor (o muchie fiind dată prin extremitățile sale). Să se verifice dacă graful este sau nu bipartit. În caz afirmativ să se afișeze o bipartiție și să se studieze dacă graful este bipartit complet. În cazul în care graful nu este bipartit, să se afișeze **un ciclu elementar impar** al acestuia. **$O(n+m)$ (1p)**



Exemplu de aplicație: *O fabrică are la dispoziție două mașini pentru a transporta n substanțe numerotate 1, 2, ..., n . Pentru siguranța transportului în caz de eventuale accidente, este întocmită o listă de perechi (i, j) de substanțe care interacționează. Să se determine dacă există o modalitate de a încărca aceste substanțe în cele două mașini pentru a fi transportate, astfel încât în fiecare dintre cele două mașini să nu se găsească substanțe care interacționează (altfel spus, două substanțe care interacționează nu sunt transportate cu aceeași mașină). Dacă există o astfel de modalitate, să se afișeze substanțele care vor fi încărcate în prima mașină și substanțele care vor fi încărcate în cel de a doua mașină.*

2. **Drum critic (Critical Path Method).** Se citesc din fișierul `activitati.in` următoarele informații despre activitățile care trebuie să se desfășoare în cadrul unui proiect:

- n – numărul de activități (activitățile sunt numerotate 1, ..., n)
- d_1, d_2, \dots, d_n durata fiecărei activități
- m – număr natural
- m perechi (i, j) cu semnificația: activitatea i trebuie să se încheie înainte să înceapă j

Activitățile se pot desfășura și în paralel.

- a) Să se determine timpul minim de finalizare a proiectului, știind că acesta începe la ora 0 (echivalent – să se determine durata proiectului) și o succesiune (critică) de activități care determină durata proiectului (un drum critic – v. curs) **$O(m + n)$. (2p)**

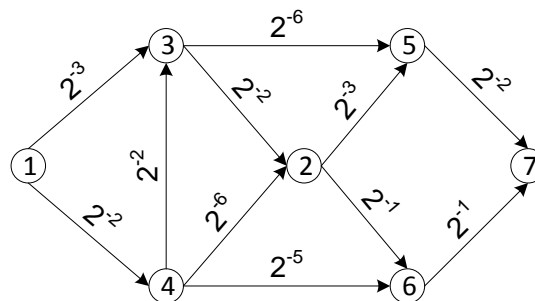
- b) Să se afișeze pentru fiecare activitate un interval posibil de desfășurare (!știind că activitățile se pot desfășura în paralel) $O(m + n)$. (1p)

Robert Sedgewick and Kevin Wayne, *Algorithms, 4th Edition, Addison-Wesley, 2011.*

activitati.in	iesire
6	Timp minim 47
7 4 30 12 2 5	Activitati critice: 4 3 6
6	1: 0 7
1 2	2: 7 8
2 3	3: 12 42
3 6	4: 0 12
4 3	5: 12 42
2 6	6: 42 47
3 5	

Algoritmul lui Dijkstra – două probleme la alegere între 3,4,5.

- Se citesc din fișierul `grafpond.in` informații despre un graf **neorientat** ponderat și de la tastatură un număr k , o listă de k puncte de control ale grafului și un vârf s . Determinați cel mai apropiat punct de control de vârful s și afișați un lanț minim până la acesta, folosind algoritmul lui **Dijkstra** (problema B.1. din laboratorul 1 pentru cazul ponderat) - $O(m \log(n))$. (3p)
- Pentru fiecare arc al unei rețele de comunicație acestui graf se cunoaște o pondere pozitivă subunitară reprezentând probabilitatea ca legătura corespunzătoare să nu se defecteze (de forma $1/2^p = 2^{-p}$). Aceste probabilități sunt independente, deci **siguranța unui drum** este egală cu produsul probabilităților asociate arcelor care îl compun. Arătați că problema determinării unui drum de siguranță maximă de la un vârf de start s la un vârf destinație t (accesibil din s) se poate reduce la o problemă de determinare a unui drum minim între s și t (pentru un graf cu ponderile modificate). Pornind de la acest fapt, implementați un algoritm bazat pe algoritmul lui Dijkstra pentru determinarea unui drum de siguranță maximă între două vârfuri s și t citite de la tastatură pentru o rețea orientată dată în fișierul `rete.in` prin următoarele informații:
 - n, m – numărul de vârfuri, respectiv arce
 - m linii conținând triplete de numere naturale i, j, p cu semnificația: (i, j) este arc în rețea cu probabilitatea să nu se defecteze egală cu 2^{-p} $O(m \log(n))$. (2p)



- <http://www.infoarena.ro/problema/catun> $O(m \log(n))$. (3p)

SUPLIMENTAR (*Se punctează doar dacă s-a obținut un minim de 8,5 puncte din cele 4 probleme precedente - 2 obligatorii si 2 la alegere*)

6. **Graf planar - 6-colorare** Se citesc din fișierul **graf.in** următoarele informații despre un graf neorientat planar: numărul de vârfuri n , numărul de muchii m și lista muchiilor (o muchie fiind dată prin extremitățile sale). Să se afișeze o 6-colorare proprie a acestuia. **$O(n+m)$** **(1.5p)**