**Drumuri minime**

* **Ca și în laboratorul trecut,** **fișierul** grafpond.in **are următoarea structură: numărul de vârfuri n, numărul de muchii/arce m şi lista muchiilor/arcelor cu costul lor (o muchie fiind dată prin extremităţile sale și cost).**

| **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 |

**Justificaţi complexitatea+corectitudinea algoritmilor propuşi.**

1. **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)
* d1, d2, …., dn 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.

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).**

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

| **activitati.in** | **iesire** |
| --- | --- |
| 6  **7 4 30 12 2 5**  **6**  **1 2**  **2 3**  **3 6**  **4 3**  **2 6**  **3 5** | Timp minim 47  Activitati critice: 4 3 6  1: 0 7  2: 7 11  3: 12 42  4: 0 12  5: 42 44  6: 42 47 |

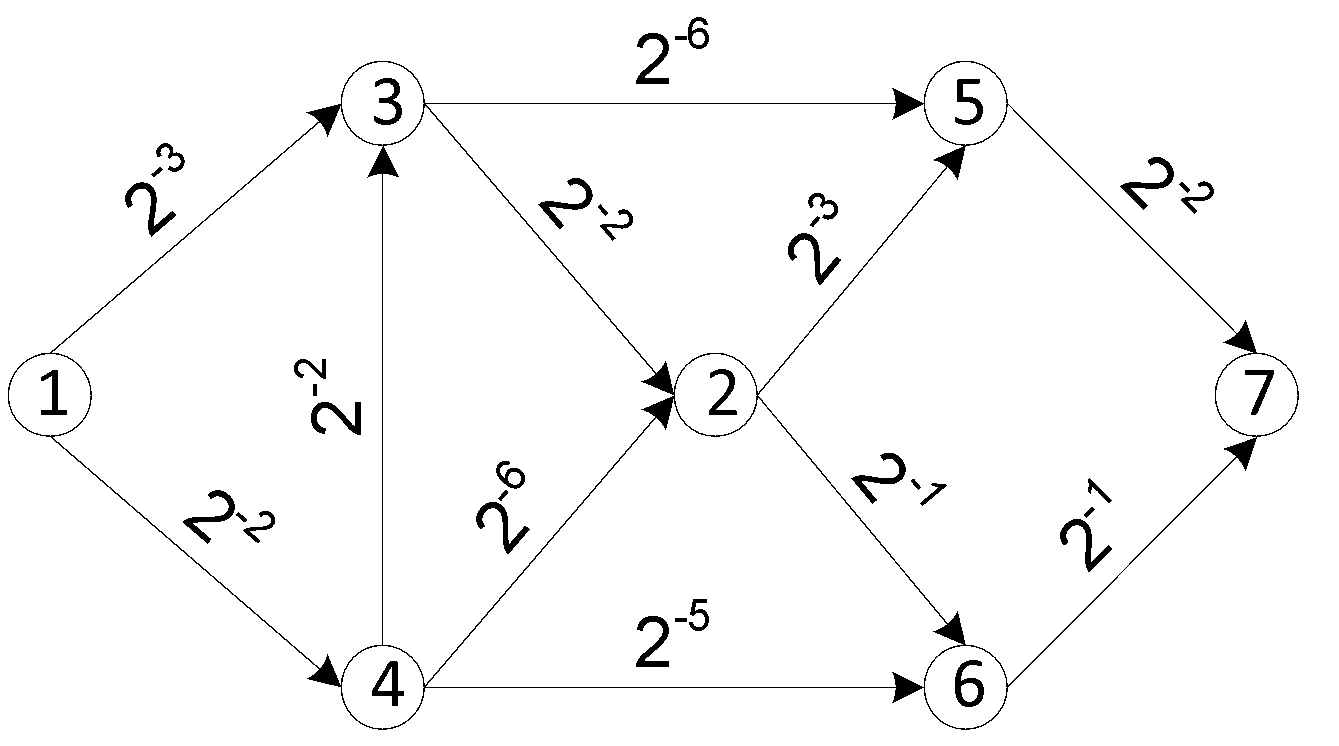
**Robert Sedgewick and Kevin Wayne, Algorithms, 4th Edition, Addison-Wesley, 2011.**

1. Se citesc din fișierul grafpond.in informații despre un graf **neorientat** ponderat, un număr k, o listă de k puncte de control ale grafului şi un vârf s. Determinați cel mai apropiat punct de control de vârful s şi afișați un lanț minim până la acesta, folosind algoritmul lui **Dijkstra** (problema B.2. din laboratorul 1 pentru cazul ponderat) - **O(m log(n)).**

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 7 9  1 2 8  1 3 1  1 4 2  3 2 4  2 6 3  3 5 6  4 5 5  6 5 2  5 7 6  3  5 6 7  1 | 5  1 3 5 (poate fi si 1 4 5) |

1. (**v. Seminar** – **Aplicație Dijkstra**) Pentru fiecare arc al unei reţele de comunicaţie acestui graf se cunoaşte o pondere pozitivă subunitară reprezentând probabilitatea ca legătura corespunzătoare să nu se defecteze (de forma 1/2p = 2-p). Aceste probabilităţi sunt independente, deci **siguranţa unui drum** este egală cu produsul probabilităţilor asociate arcelor care îl compun. Arătaţi că problema determinării unui drum de siguranţă maximă de la un vârf de start s la un vârf destinaţie t (accesibil din s) se poate reduce la o problemă de determinare a unui drum minim între s şi t (pentru un graf cu ponderile modificate). Pornind de la acest fapt, implementaţi un algoritm bazat pe algoritmul lui **Dijkstra** pentru determinarea unui drum de siguranţă maximă între două vârfuri s şi t citite de la tastatură pentru o reţea orientată dată în fişierul retea.in prin următoarele informații:

* n, m – numărul de vârfuri, respectiv arce
* m linii conţinând triplete de numere naturale i j p cu semnificaţia: (i,j) este arc în reţea cu probabilitatea să nu se defecteze egală cu 2-p **O(m log(n)).**



Drumul de de siguranță maximă de la 1 la 7 este 1 3 2 6 7. Siguranța acestui drum este 2-7

1. **Drumuri minime din surse multiple** <http://www.infoarena.ro/problema/catun> **O(m log(n)).**
2. **Bellman Ford** Se dă un graf orientat ponderat (în fișierul grafpond.in) și un vârf s. Dacă graful nu conține circuite negative accesibile din s afișați câte un drum minim de la s la fiecare dintre celelalte vârfuri accesibile din s, altfel afișați un astfel de circuit (folosind algoritmul Bellman Ford) **O(nm)**

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 4 4  1 2 1  4 2 -7  2 3 2  3 4 3  2 | Circuit de cost negativ:  2 3 4 |

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 4 5  1 2 2  4 2 7  2 3 2  3 4 3  1 3 1  1 | Drum: 1 2 Cost: 2  Drum: 1 3 Cost: 1  Drum: 1 3 4 Cost: 4 |

1. **Floyd-Warhsall**

**a)** Dat un graf orientat ponderat (în fisierul grafpond.in), afișați matricea distanțelor dacă graful nu conține circuite de cost negativ și un circuit cu cost negativ în caz contrar. **O(n**3)

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 4 4  1 2 1  4 2 -7  2 3 2  3 4 3 | Circuit de cost negativ:  2 3 4 |

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 4 4  1 2 1  4 2 7  2 3 2  3 4 3 | 0 1 3 6  0 0 2 5  0 0 0 3  0 7 9 0 |

**b)** Fie G un graf neorientat ponderat.Pentru două vârfuri u şi v ale lui G, notăm cu d(u, v) **distanța** de la vârful u la vârful v.

Pentru un vârf v, **excentricitatea** lui v este cea mai mare distanță de la acest vârf la celelalte vârfuri:

e(v) = max{d(v, u)| u ∈ V }

Excentricitatea minimă a vârfurilor se numește **raza** grafului:

r(G) = min{e(v)| v ∈ V }

Mulțimea vârfurilor cu excentricitatea minimă (egală cu r(G)) se numeşte **centrul** grafului:

c(G) = { v ∈ V | e(v) = r(G)}

Excentricitatea maximă a vârfurilor se numeşte **diametrul** grafului; altfel spus, diametrul este cea mai mare distanță dintre două vârfuri:

diam(G) = max{e(v)| v ∈ V } = max{d(u, v)| v, u ∈ V }

Se citesc din fișierul grafpond.in informații despre un graf **neorientat** ponderat G. Să se determine, folosind algoritmul **Floyd-Warhsall**, raza, diametrul, centrul grafului și un lanț diametral (un lanț minim P între două vârfuri u și v cu ponderea w(P)=d(u,v)=diam(G)). **O(n**3)

| **grafpond.in** | **grafpond.out** |
| --- | --- |
| 7 9  1 2 8  1 3 1  1 4 2  3 2 4  2 6 3  3 5 6  4 5 5  6 5 2  5 7 6 | Raza: 7  Centrul: 5  Diametrul: 13  Lant diametral: 1 4 5 7 |