DRUMURI IN GRAFURI

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

8. DRUMURI IN GRAFURI

Definitii:

- Valoarea unui drum
- Probleme de optim
- Principiul optimalitatii (Bellman-Kalaba, BK) in programarea dinamica

Proprietati. Teoreme

- Probleme care se pun referitor la drumuri
- Conditie: sa existe drum de valoare minima: graful sa nu contina circuite de valoare negativa
- daca exista un asemenea circuit => valoarea min.scade la fiecare parcurgere –infinit
- Drumuri de valoare maxima

- alg. Bellman-Kalaba se transforma in alg. sa depisteze circuite (cicluri)

- val.max. exista daca graful nu contine circuite de valoare pozitiva

- se pot aplica alg. anteriori + modificari (initializare valori: +1 => -1; conditiile de test: "min""<" cu "max", " > ")

- Drumuri in grafuri ; Optimizari de drumuri/lanturi

Algoritmi (pseudocod)

3 alg. Djikstra (a) Initializare, b) Test de oprire:c) Iteratia de baza)

- det. Val. **mini drumuri de la nod 1 la toate celelalte noduri** (val. pozitive arce)
- det. lungimi **min. drum. de la vf.1 la celelalte** (R = {1}; S = {1}, l(u)=1)
- det. Val. **Min. drumuri de la nodul 1 la toate celelalte; graf cu valori oarecare fara circuite de valoare negativa**

- alg. Bellman-Kalaba--- det. **val. minime ale drumurilor de la vf. 1 la nod oarecare sau detecteaza existenta unui circuit de valoare negativa**
- alg. **Ford** (a) Initializari; b) Iteratia de baza c) Criteriul de stop) --- det. Val. **Min. drumuri de la nodul 1 la orice nod fara circuite de valoare negativa**
- **alg.** (a) Initializari; b) Iteratia de baza c) Criteriul de stop)---det. drumurilor de val. Min. daca val. Min. drumurilor de la nodul 1 la celelalte sunt determinate
- alg. matriceali pt. Val.min. (programarea dinamica)---det. val. min. drumuri intre oricare doua vf.

3 Alg. **Floyd-Warshall-Hu**

- det. **matricea valorilor minime** ale drumurilor intre oricare 2 noduri ale unui graf care **nu contine circuite de valoare negativa**
- det. **matricea valorilor minime** ale drumurilor intre oricare 2 vf. sau **depisteaza existenta unui circuit de valoare negativa**
- det**. matricea valorilor minime** ale drumurilor intre oricare 2 noduri**; graf fara circuite val. negativa si drumurile respective**

2 alg. Bellman-Kalaba (BK) (a) Init.; b) Iteratia de baza c) Criteriul de stop)

--- det.val. **MAXIME** ale drumurilor de la vf. 1 la oarecare sau detecteaza existenta unui circuit de valoare POZITIVA

--- Drumuri de valoare minima cazul **lungime=Produs--**det.val. min. drumuri de la vf. 1 la orice nod sau detecteaza existenta **circuit val. subunitara**

Drumuri în grafuri

Probleme care se pot modela ca probleme de drumuri în grafuri:

- trasee turistice, rutiere,...
- programarea unor turnee, sportive sau nu
- proiectarea sau realizarea unor investiții
- gestiunea stocurilor
- optimizări în rețele
- probleme de inteligență artificială și recunoașterea formelor
- probleme de generare/acceptare a unor limbaje
- probleme de tratare numerică a semnalelor, a codificării/decodificării informației
- probleme militare.

Drumuri în grafuri

Probleme care se pun referitor la drumuri:

- verificarea existenței unui drum între două noduri ale unui graf
- verificarea existenței unui drum de lungime dată
- determinarea drumurilor de lungime dată
- optimizări pe mulțimea drumurilor.

Drumuri în grafuri

$G = (X, U)$ graf orientat/neorientat

- se asociază fiecărui arc sau muchie $u \in U$ un număr real $l(u)$

- semnificația lui $l(u)$ poate fi:

  - distanța dintre extremitățile arcului u
  - durata trecerii de la extremitatea inițială la extremitatea terminală a arcului
  - costul acestei treceri
  - probabilitatea de trecere pe acel arc
  - capacitatea arcului respectiv, etc

  $l(u)$ - valoarea sau ponderea arcului/muchiei $u \in U$

Drumuri în grafuri

Valoarea unui drum μ, notată $l(\mu)$, în graful $G = (X, U)$ se poate defini în mai multe moduri

$$l(\mu) = \sum_{u \in \mu} l(u)$$

$$l(\mu) = \prod_{u \in \mu} l(u)$$

$$l(\mu) = min(max)\{l(u) \mid u \in \mu\}$$

Probleme de optim - determinarea valorilor minime sau maxime a drumurilor dintr-o mulțime de drumuri ale unui graf G, eventual precizarea unui drum din mulțime care să aibă valoarea optimă.

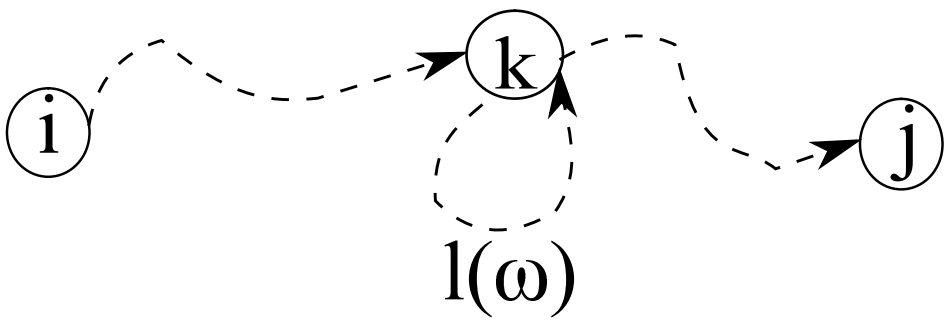
- se va studia la început problema determinării valorilor minime și apoi a drumurilor/lanțurilor cu această valoare în graf.

- dacă se alege ca valoare pentru fiecare arc/muchie valoarea 1, atunci se determină lungimile minime/maxime ale drumurilor/lanțurilor în graful dat.

Optimizări de drumuri/lanțuri în ipoteza $l(\mu) = \sum_{u \in \mu} l(u)$

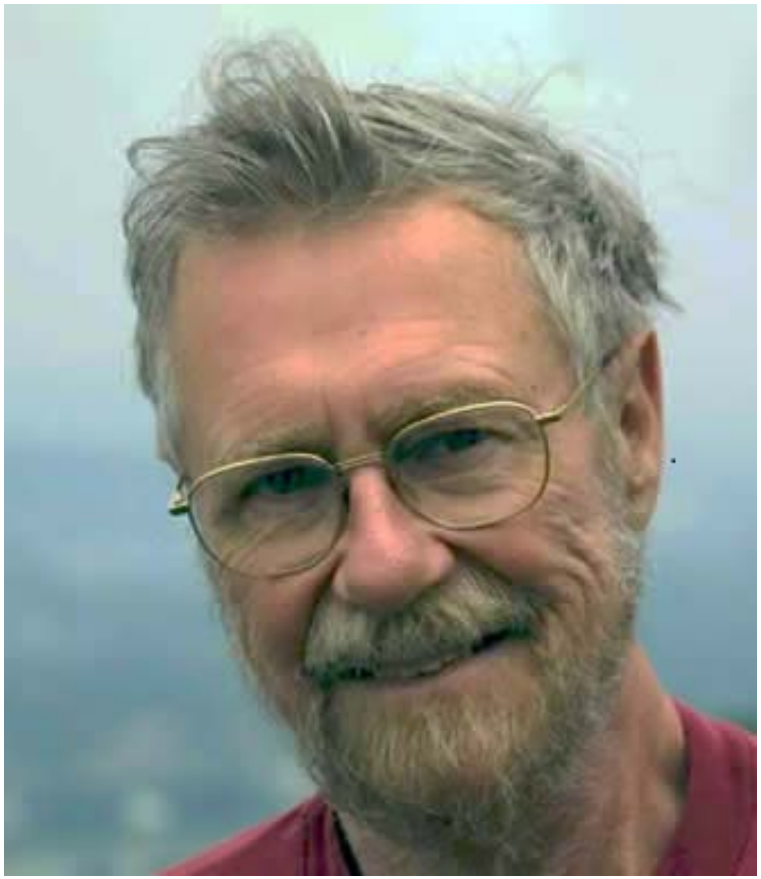
- o condiție ca într-un graf $G = (X, U)$ să existe drum de valoare minimă este ca graful să nu conțină circuite de valoare negativă

- dacă există un asemenea circuit ω, $l(\omega) < 0$ atunci valoarea minimă a drumului scade la fiecare parcurgere a circuitului și devine $-\infty$



E.W.Dijkstra

EDSGER WYBE DIJKSTRA (1930-2002)



E.W.Dijkstra

- a studiat fizică teoretică la Universitatea din Leiden

1959 - doctorat la Universitatea din Amsterdam cu teza 'Communication with an Automatic Computer'- descrierea limbajului de asamblare

1952 - 1962 Mathematisch Centrum (Amsterdam) - "primul programator" în Olanda

1962- 1984 profesor de matematică la Universitatea Tehnică din Eindhoven- sistemul de operare THE

1965 " Notes on Structured Programming" declară programarea ca fiind disciplină nu meșteșug

1984-1999 Universitatea din Austin

E.W.Dijkstra

Domenii de cercetare:

- compilatoare și limbaje de programare

- sisteme de operare

- programare concurentă

- programare distribuită

- programare- design,dezvoltare, verificare

-principiile ingineriei software

- algoritmi

1972 - Association for Computing Machinery (ACM) îi oferă Turing Award

E.W.Dijkstra

Program testing can be used to show the presence of bugs, but never to show their absence!

We must be very careful when we give advice to younger people: sometimes they follow it!

We must not forget that it is not our [computing scientists'] business to make programs, it is our business to design classes of computations that will display a desired behaviour.

The question of whether Machines Can Think... is about as relevant as the question of whether Submarines Can Swim.

Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

Optimizări de drumuri/lanțuri în ipoteza $l(\mu) = \sum_{u \in \mu} l(u)$

Fie graful $G = (X, U)$ cu mulțimea nodurilor $X = \{1, 2, ..., n\}$

- $u = (i, j) \in U$ notăm $l_{ij} = l(u)$ valoarea sa

- λ_i valoarea minimă a drumurilor de la vârful 1 la vârful i

Algoritmul lui Dijkstra

- determinarea valorilor minime ale drumurilor de la nodul 1 la toate celelalte noduri, într-un graf cu valori pozitive ale arcelor,
$l_{ij} = l(u) > 0,\ \forall\ u \in U$

a) Inițializare:

$$\lambda_1 = 0,\ \lambda_i = \begin{cases} l_{1i},\ (1,i) \in U \\ \infty,\ altfel \end{cases},\ i = \overline{2,n},\ S = \{2,3,...,n\}$$

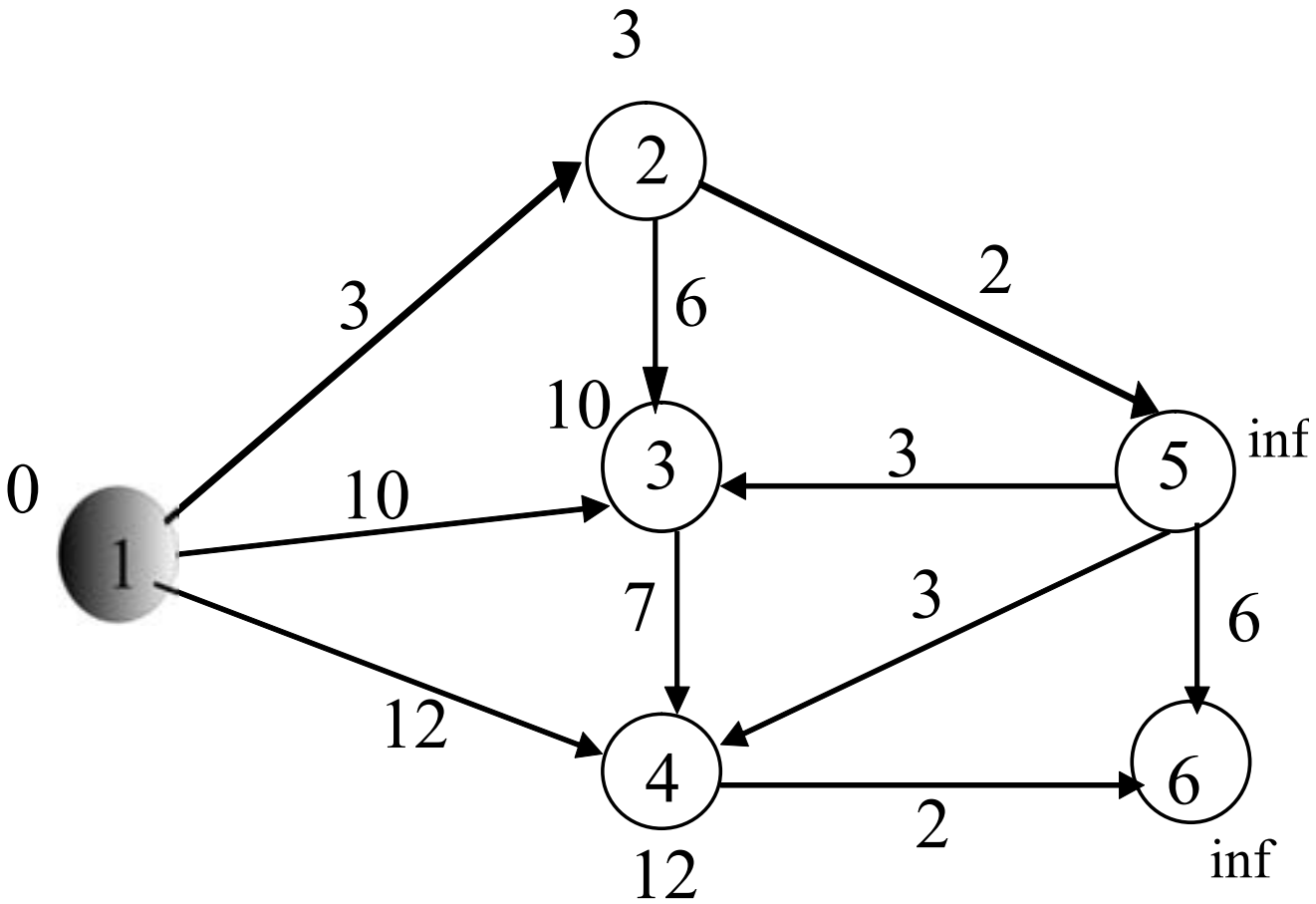
b) Test de oprire:

Fie $j \in S$ a.î. $\lambda_j = min\{\lambda_i \mid i \in S\}$; $S = S \setminus \{j\}$;
Dacă $S = \emptyset$ atunci STOP;
altfel goto c)

c) Iterația de bază:

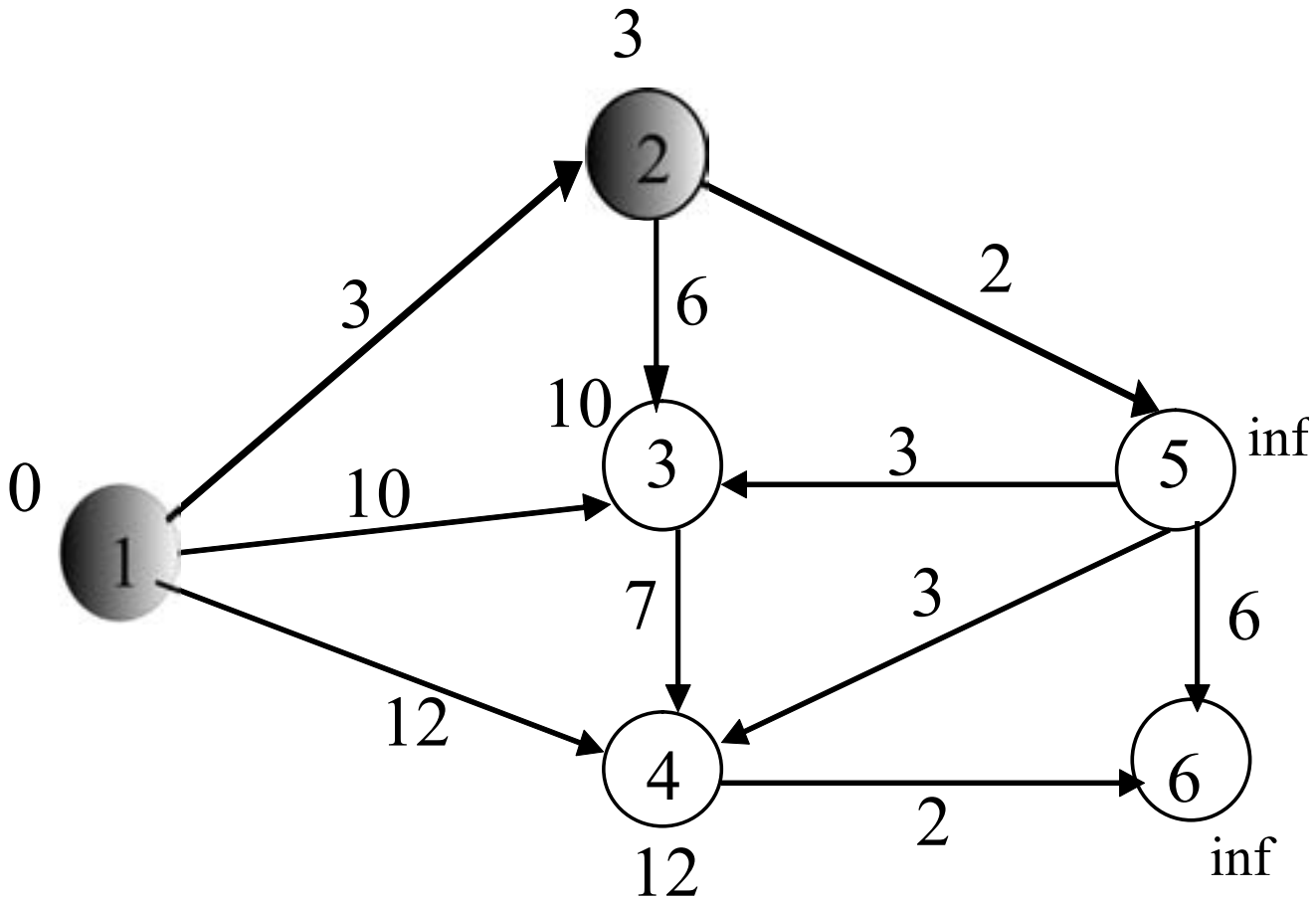
Pentru $i \in \Gamma j \cap S$ execută $\lambda_i = min\{\lambda_i,\ \lambda_j + l_{ji}\}$;
Goto b)

Algoritmul lui Dijkstra



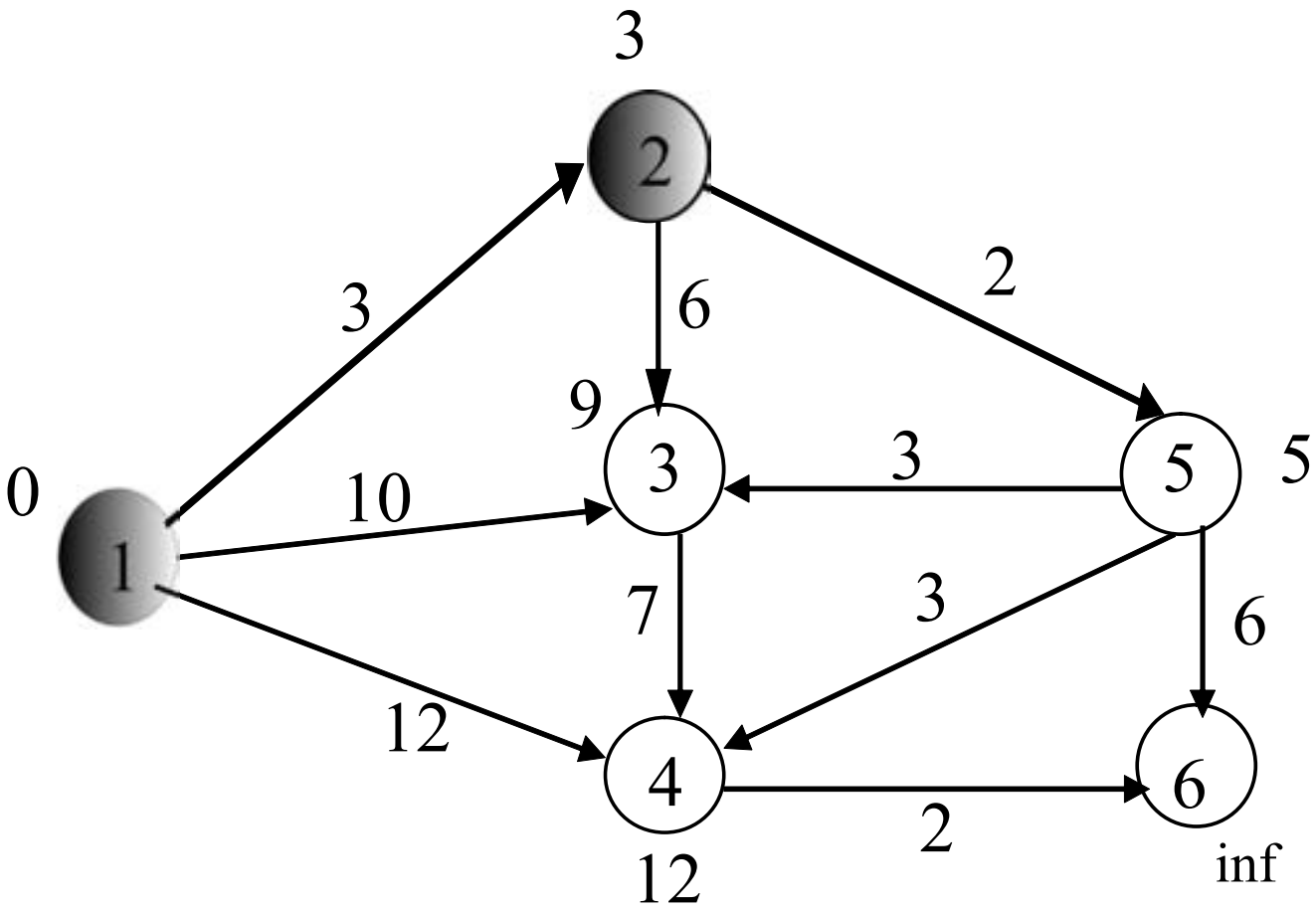
$S = \{2, 3, 4, 5, 6\}$,
$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 10,\ \lambda_4 = 12,\ \lambda_5 = inf,\ \lambda_6 = inf$

Algoritmul lui Dijkstra



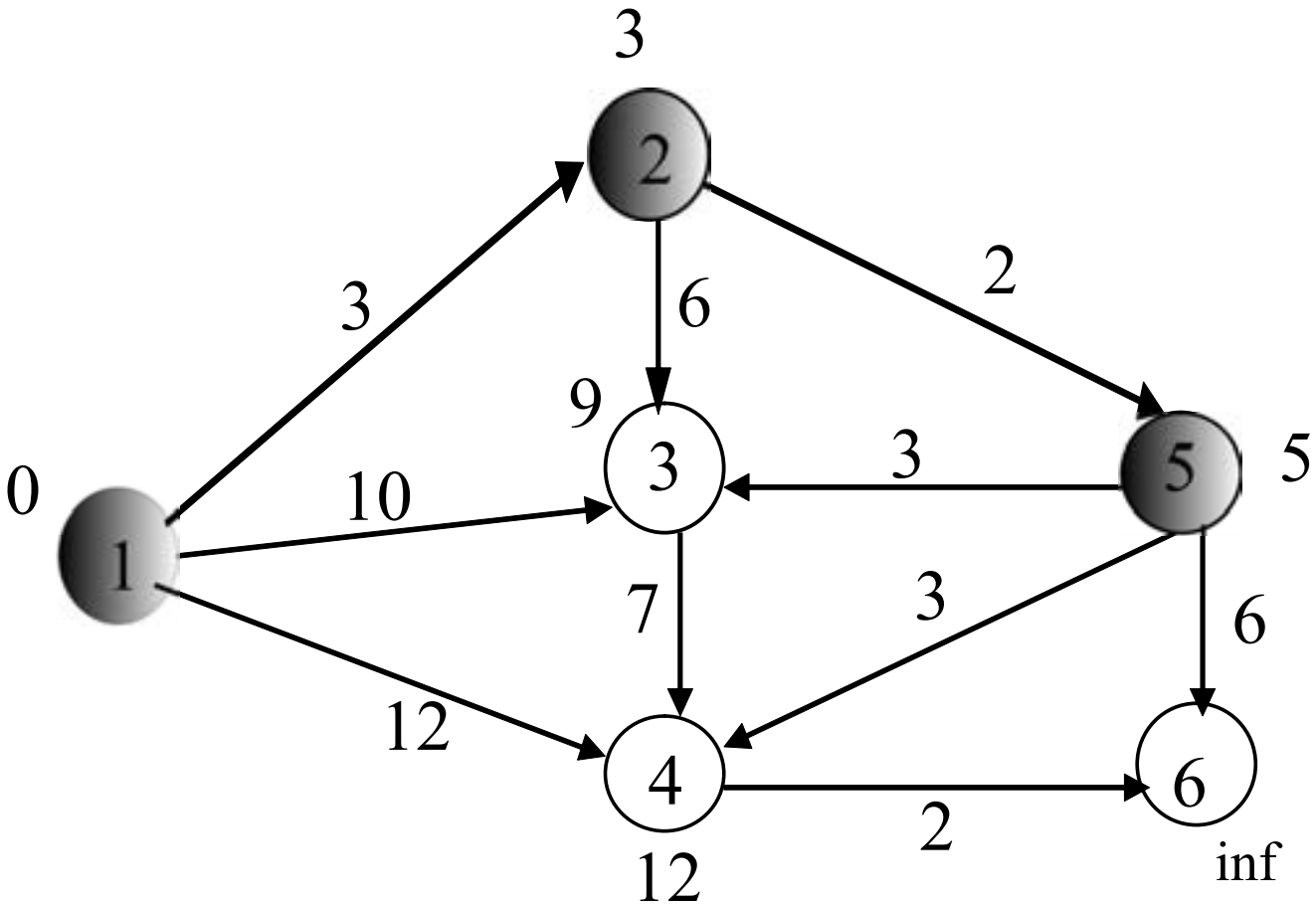
$j = 2;\ S = \{3, 4, 5, 6\},$

Algoritmul lui Dijkstra



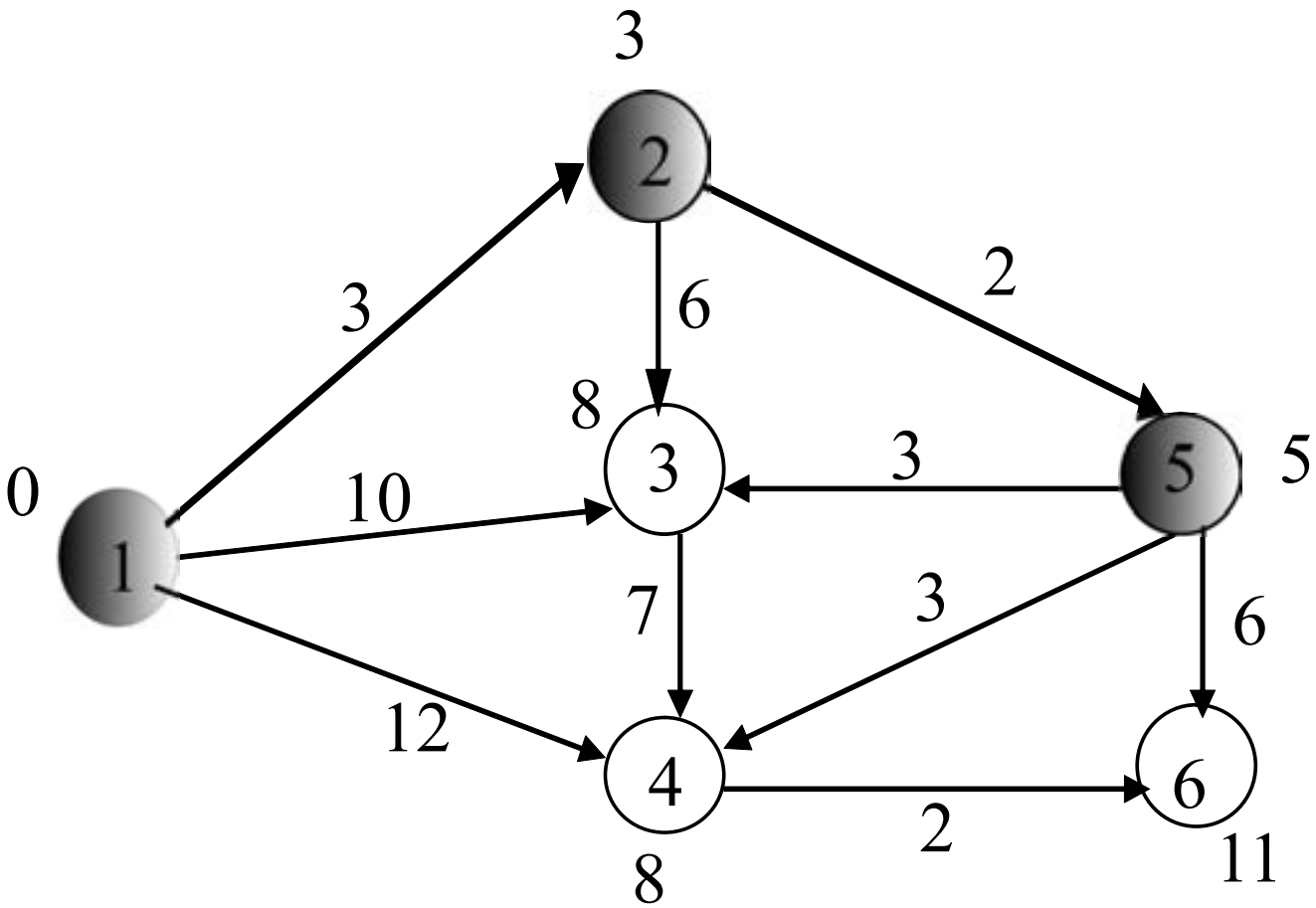
$j = 2;\ S = \{3, 4, 5, 6\},$
$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 9,\ \lambda_4 = 12,\ \lambda_5 = 5,\ \lambda_6 = inf$

Algoritmul lui Dijkstra



$j = 5;\ S = \{3, 4, 6\},$

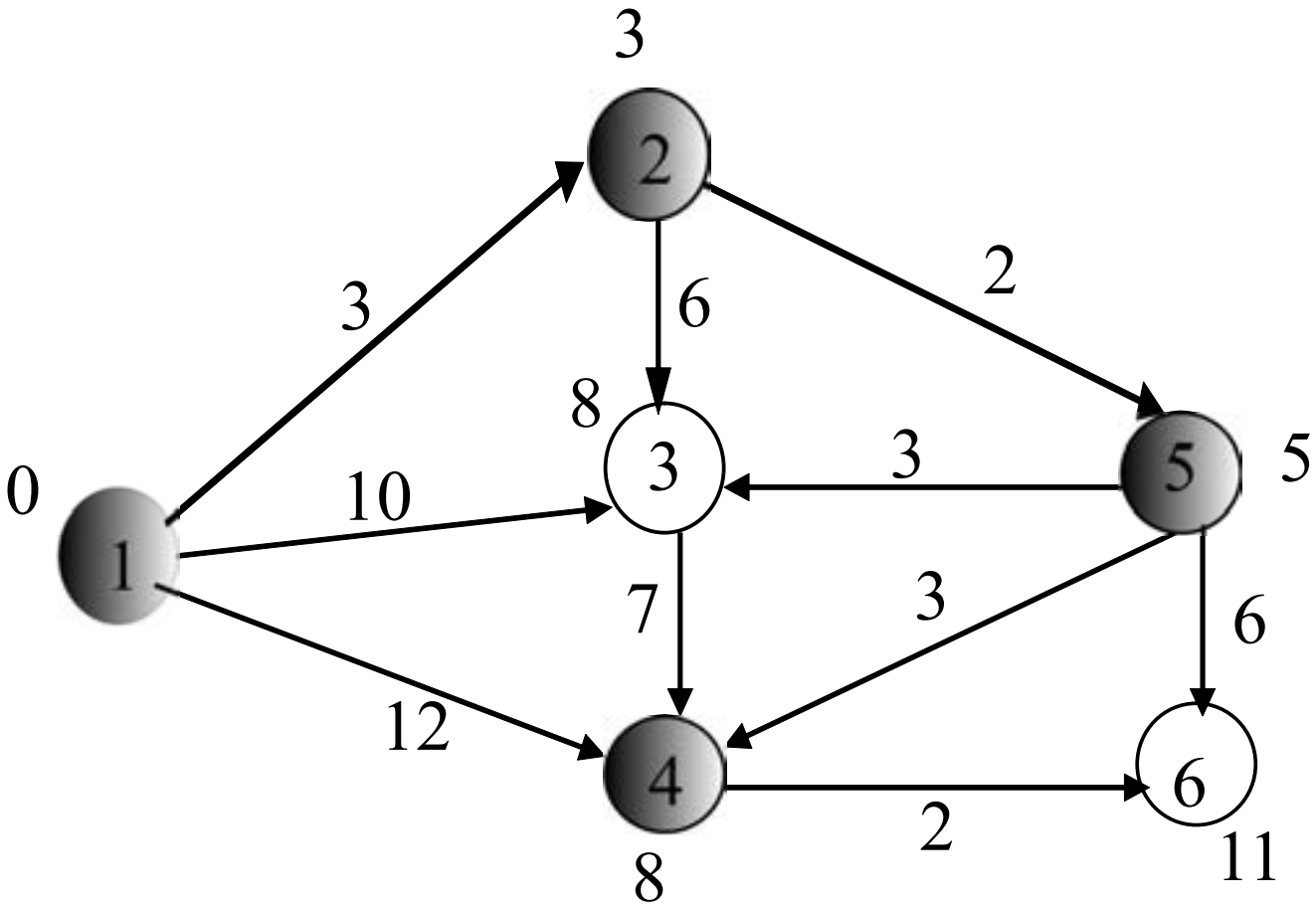
Algoritmul lui Dijkstra



$j = 5;\ S = \{3, 4, 6\},$

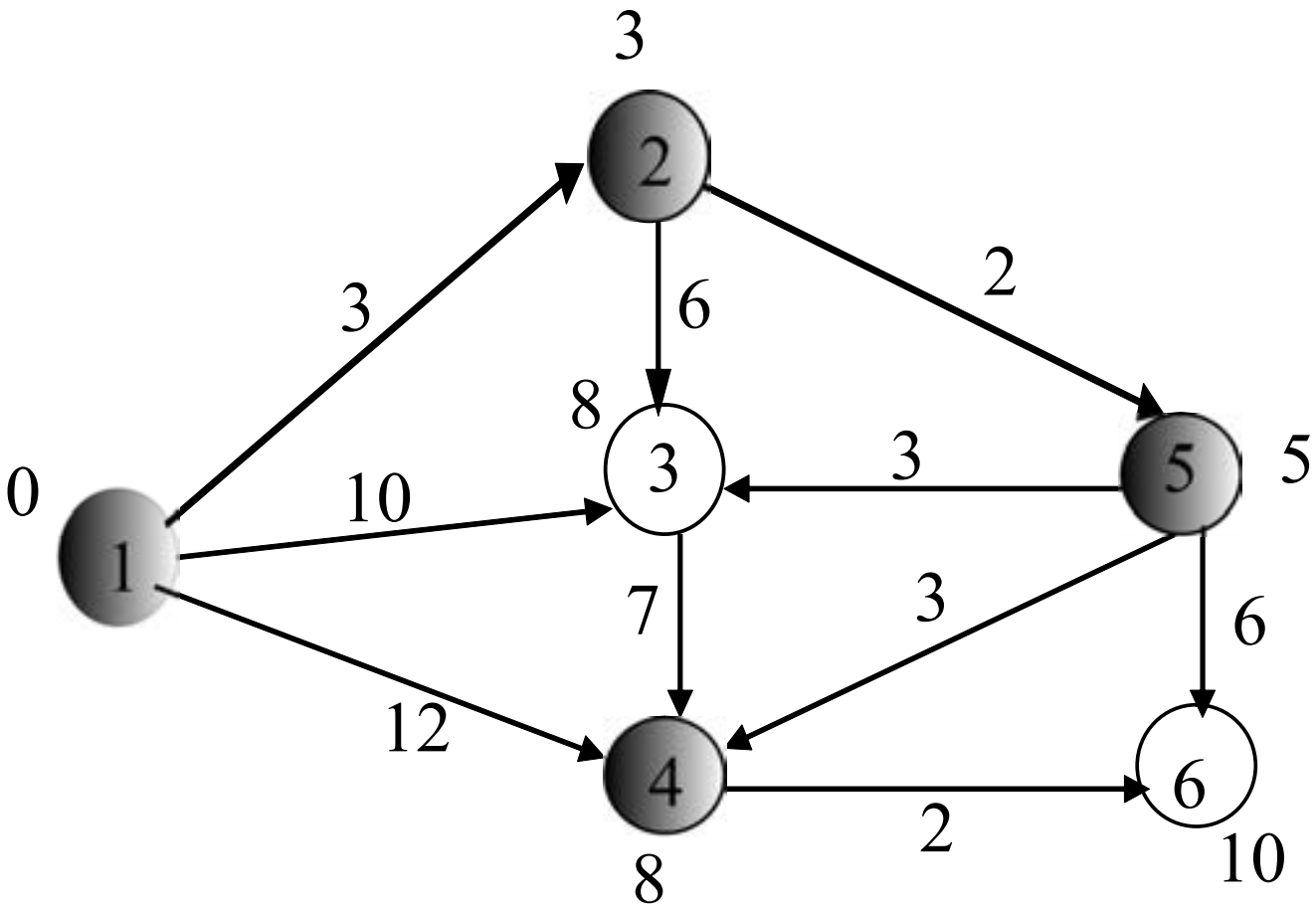
$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 8,\ \lambda_4 = 12,\ \lambda_5 = 5,\ \lambda_6 = 11$

Algoritmul lui Dijkstra



$j = 4;\ S = \{3, 6\}$

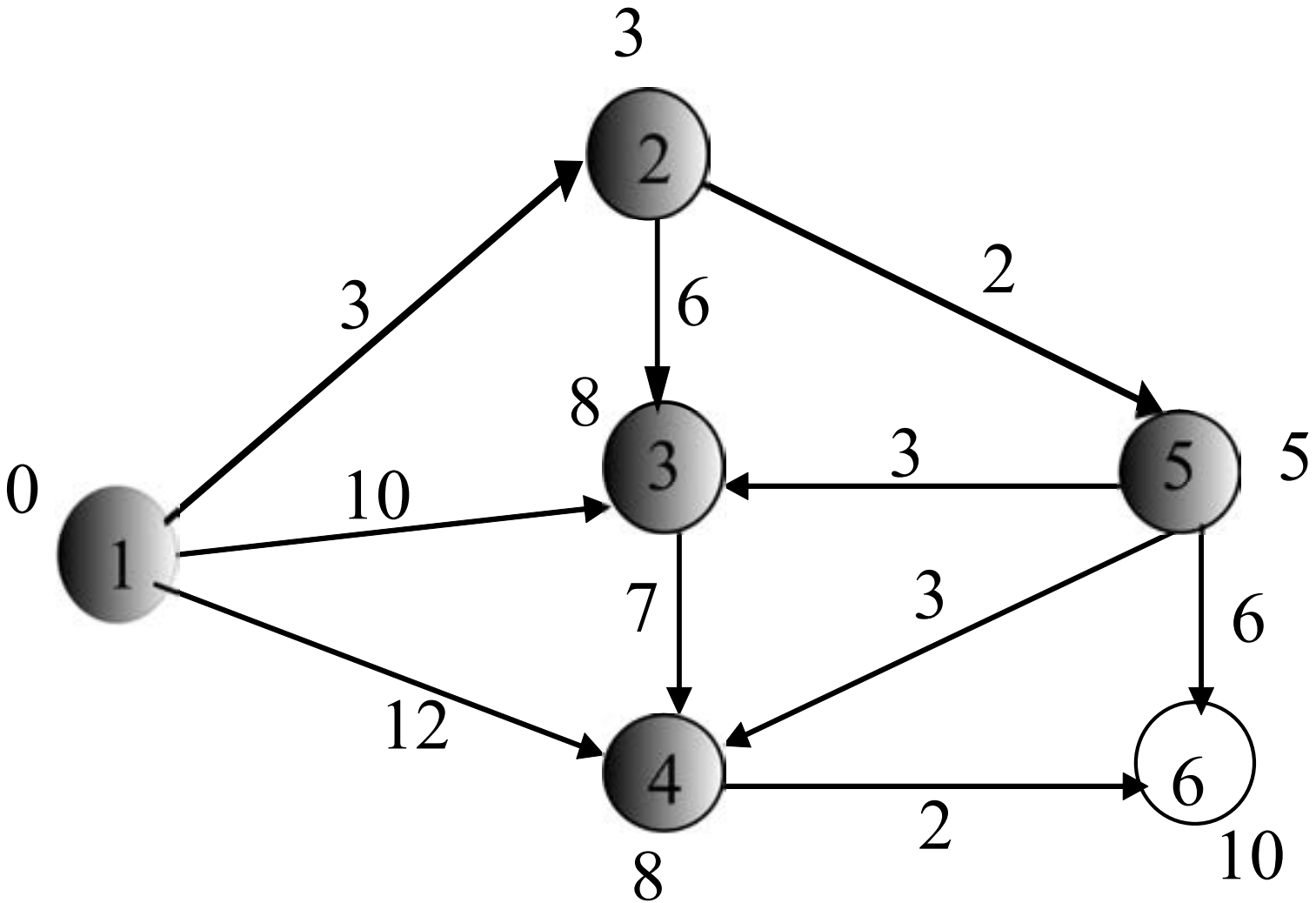
Algoritmul lui Dijkstra



$j = 4;\ S = \{3, 6\},$

$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 8,\ \lambda_4 = 12,\ \lambda_5 = 5,\ \lambda_6 = 10$

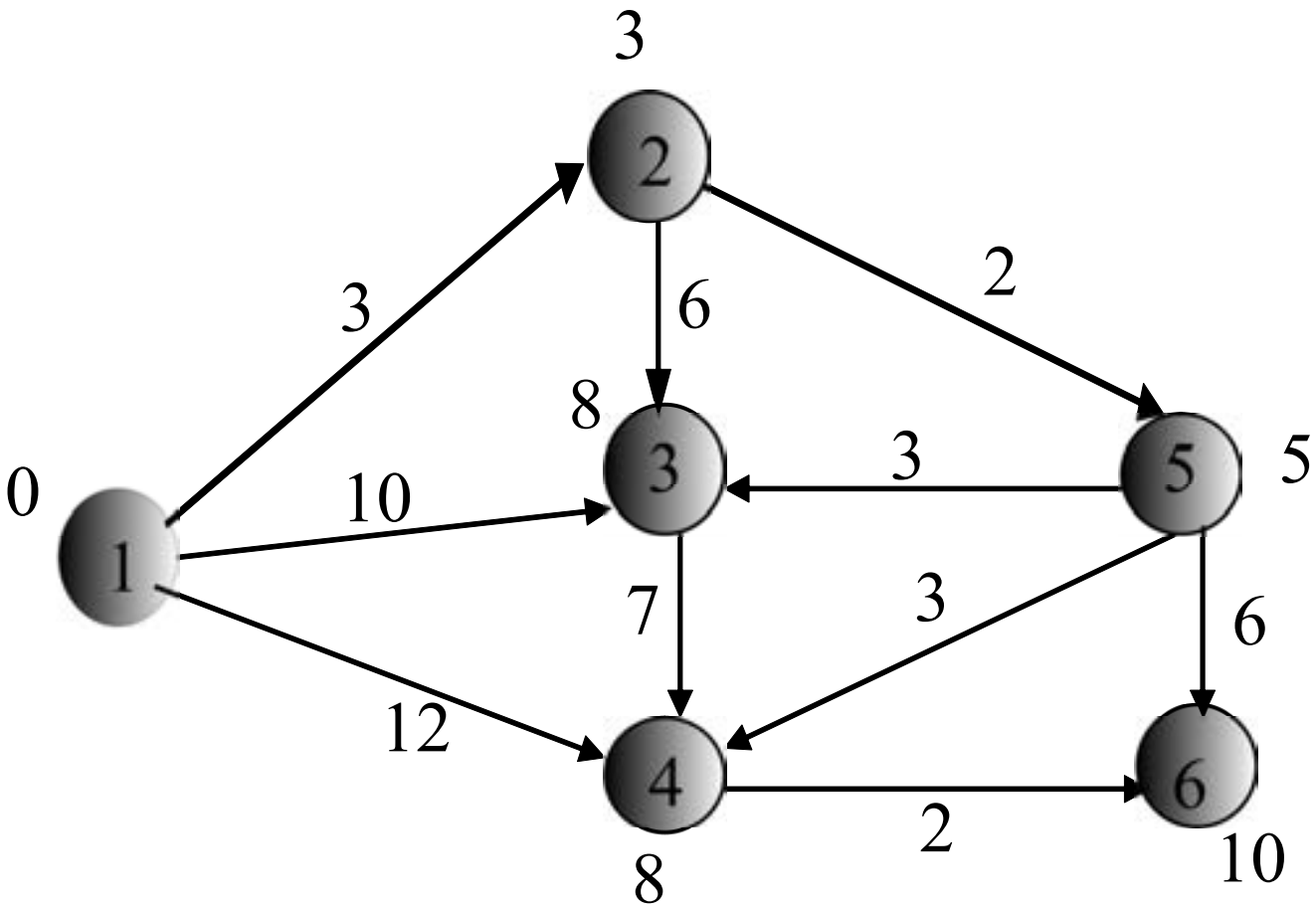
Algoritmul lui Dijkstra



$j = 3;\ S = \{6\}$

$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 8,\ \lambda_4 = 12,\ \lambda_5 = 5,\ \lambda_6 = 10$

Algoritmul lui Dijkstra



$j = 6;\ S = \emptyset \Rightarrow$ STOP

$\lambda_1 = 0,\ \lambda_2 = 3,\ \lambda_3 = 8,\ \lambda_4 = 12,\ \lambda_5 = 5,\ \lambda_6 = 10$

Algoritmul lui Dijkstra

- pentru determinarea lungimilor minime ale drumurilor de la vârful 1 la celelalte, avem $l(u) = 1 \ \forall \ u \in U$:

(a) Inițializări

$$\lambda_i = \begin{cases} 0, & i = 1 \\ \infty, & i \geq 2 \end{cases} \quad ; \ k = 0; \ R = \{1\}; \ S = \{1\}$$

(b) Iterația de bază

Fie $j \in S$ a.î. $\lambda_j = min\{\lambda_i \mid i \in S\}$;
Pentru $i \in \Gamma j \cap (X \setminus R)$ execută $\lambda_i = k + 1$;
$S = \Gamma S; \ R = R \cup S;$

(c) Testul de oprire

Dacă $(R = S)$ atunci STOP
altfel $k = k + 1$;
goto (b);

Algoritmul lui Dijkstra - variantă

- determinarea valorilor minime ale drumurilor de la nodul 1 la toate celelalte într-un graf cu valori oarecare ale arcelor, $l_{ij} = l(u) \in \mathbb{R},\ \forall\ u \in U$, fără circuite de valoare negativă

a) Inițializare:

$$\lambda_1 = 0,\ \lambda_i = \begin{cases} l_{1i},\ (1,i) \in U \\ \infty,\ altfel \end{cases},\ i = \overline{2,n};\ S = \{2,3,...,n\}$$

b) Test de oprire:

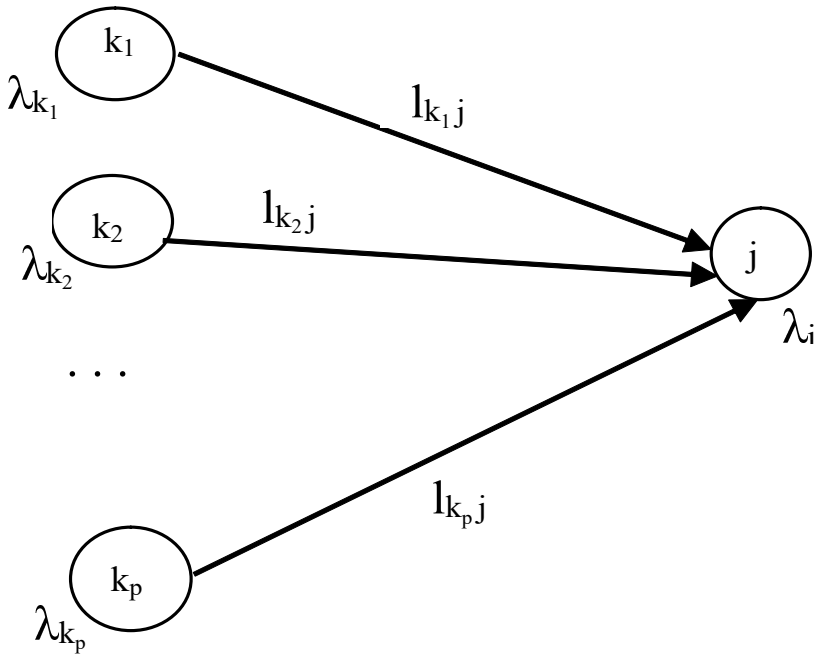
Fie $j \in S$ a.î. $\lambda_j = min\{\lambda_i \mid i \in S\}$;
$S = S \setminus \{j\}$;
Dacă $S = \emptyset$ atunci STOP;
altfel goto b)

c) Iterația de bază:

Pentru $i \in \Gamma j$ execută
dacă $(\lambda_j + l_{ji} < \lambda_i)$ atunci
$\lambda_i = \lambda_j + l_{ji}$; $S = S \cup \{i\}$;
goto b);

Algoritmul Bellman-Kalaba

Principiul optimalității în programarea dinamică: *o politică este optimală dacă la fiecare moment, oricare ar fi deciziile precedente, deciziile care urmează a fi luate constituie o politică optimală în raport cu rezultatul deciziilor anterioare.*



$$\lambda_j = \min\{\lambda_{k_1} + l_{k_1 j},\ \lambda_{k_2} + l_{k_2 j}, ...,\ \lambda_{k_p} + l_{k_p j} |\ \Gamma^{-1}(j) = \{k_1, k_2, ..., k_p\}\}$$

Algoritmul Bellman-Kalaba

Fie graful $G = (X, U),\ X = \{1, 2, ..., n\},\ l(u) \in \mathbb{R}$

$\lambda_j =$ valoarea minimă a drumului de la nodul 1 la nodul $j,\ j \in \{2, ..., n\}$

- conform principiului optimalității (BK)

$\lambda_j = \min\{\lambda_{k_1} + l_{k_1 j},\ \lambda_{k_2} + l_{k_2 j}, ...,\ \lambda_{k_p} + l_{k_p j} |\ \Gamma^{-1}(j) = \{k_1, k_2, ..., k_p\}\} =$

$= \min\{\lambda_i + l_{ij} |\ i \in \Gamma^{-1} j\}$

- se definește matricea valorilor $V \in \mathcal{M}_{n \times n}(\mathbb{R}),\ v_{ij} = \begin{cases} l_{ij},\ (i,j) \in U \\ 0,\ i = j \\ +\infty,\ altfel \end{cases}$

- atunci $\lambda_j = \min\{\lambda_i + v_{ij} |\ i = \overline{1, n}\}$

Algoritmul Bellman-Kalaba

- determină valorile minime ale drumurilor de la vârful 1 la $x \in X$ în G oarecare sau detectează existența unui circuit de valoare negativă

a) Inițializări: $k = 1;\ \lambda_1^k = 0,\ \lambda_j^k = v_{1j},\ j = \overline{2, n}$;

b) Iterația de bază:

$$\lambda_j^{k+1} = \begin{cases} 0,\ j = 1 \\ \min\{\lambda_i + v_{ij} |\ i = \overline{1, n}\},\ j = \overline{2, n} \end{cases} \quad ; \quad k = k + 1;$$

c) Criteriul de stop

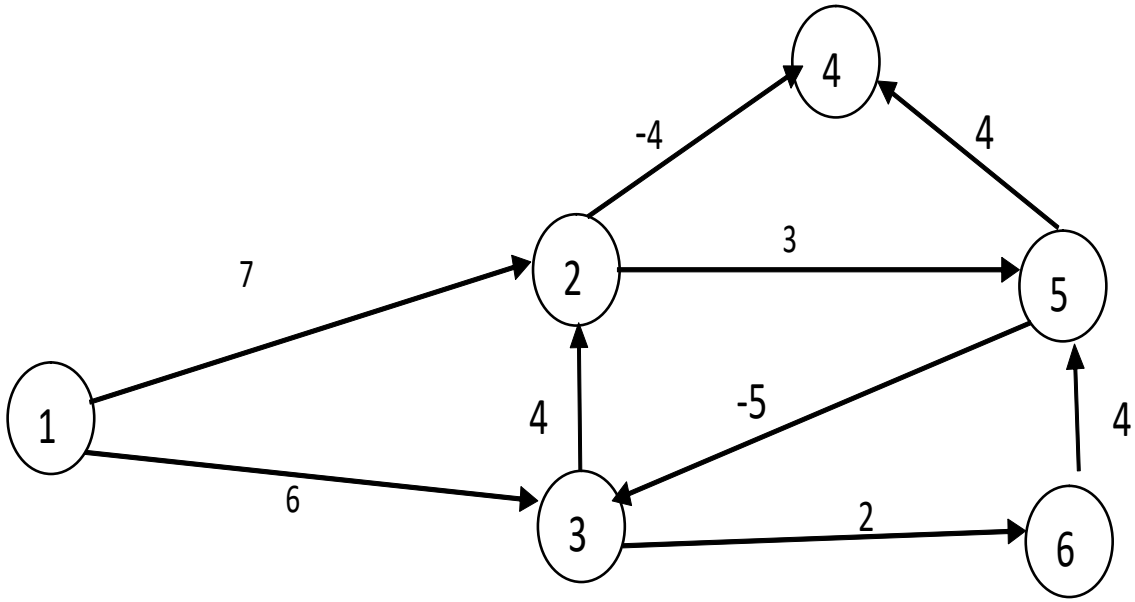
dacă $(\lambda_j^k = \lambda_j^{k-1},\ j = \overline{1, n})$ atunci STOP;

dacă $(k \leq m)$ atunci goto b)

altfel tipărește "există circuit de valoare negativă";

STOP;

Algoritmul Bellman-Kalaba



$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

Algoritmul Bellman-Kalaba

$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
$k=1$	0	7	6	∞	∞	∞

Algoritmul Bellman-Kalaba

$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
$k=1$	0	7	6	∞	∞	∞
$k=2$	0	7	6	3	10	8

Algoritmul Bellman-Kalaba

$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
$k=1$	0	7	6	∞	∞	∞
$k=2$	0	7	6	3	10	8
$k=3$	0	7	5	3	10	8

Algoritmul Bellman-Kalaba

$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
$k=1$	0	7	6	∞	∞	∞
$k=2$	0	7	6	3	10	8
$k=3$	0	7	5	3	10	8
$k=4$	0	7	5	3	10	7

Algoritmul Bellman-Kalaba

$$V = \begin{pmatrix} 0 & 7 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & -4 & 3 & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -5 & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 4 & 0 \end{pmatrix}$$

k	λ_1^k	λ_2^k	λ_3^k	λ_4^k	λ_5^k	λ_6^k
$k=1$	0	7	6	∞	∞	∞
$k=2$	0	7	6	3	10	8
$k=3$	0	7	5	3	10	8
$k=4$	0	7	5	3	10	7
$k=5$	0	7	5	3	10	7

$$\lambda_j^{(5)} = \lambda_j^{(4)},\ j = \overline{1,6} \Rightarrow \lambda_j = \lambda_j^{(5)},\ j = \overline{1,6}$$

Algoritmul Bellman-Kalaba

- pentru a determina elementul $\lambda_j^{(k+1)}$ de pe linia $k+1$ din tablou se procedează astfel:

  - se adună linia k din tablou cu coloana j din matricea valorilor arcelor apoi se determină valoarea minimă a rezultatelor;
  - valoarea minimă se trece în tablou pe poziția j din linia $k+1$

- numărul $\lambda_j^{(k)}$ reprezintă valoarea minimă a drumurilor de la vârful 1 la vârful j care au lungimea cel mult k, adică valoarea minimă a drumurilor care trec prin cel mult $k+1$ vârfuri

- dacă nu există circuite de valoare negativă atunci, cum drumul de valoare minimă este elementar, înseamnă că el trece prin cel mult n vârfuri și deci valoarea minimă se află în cel mult $n-1$ iterații

Algoritmul lui Ford

- se determină valorile minime ale drumurilor de la nodul 1 la orice $x \in X$ în G fără circuite de valoare negativă

a) Inițializări:

$$\lambda_1 = 0,\ \lambda_j = v_{1j},\ j = \overline{2, n}\ ;$$

b) Iterația de bază:

Pentru $j \in \{2, ..., n\}$ execută $\lambda_j = \min\{\lambda_i + l_{ij} |\ i \in \Gamma^{-1} j\}$;

c) Criteriul de stop:

Dacă (în vectorul λ nu se schimbă nici o valoarea) atunci STOP

altfel goto b);

- modul de parcurgere al vârfurilor în etapa b) a algoritmului lui Ford influențează numărul de iterații necesare: dacă vârfurile sunt parcurse în ordine crescătoare a valorilor λ_j atunci ar fi suficientă doar o iterație pentru a obține valorile minime

Determinarea drumurilor de valoare minimă

- dacă valorile minime λ_i ale drumurilor de la nodul 1 la celelalte sunt determinate, se pot detremina drumurile da valoare minimă astfel:

a) Inițializări:

$k = 1;\ d(k) = i;$

b) Iterația de bază:

$k = k + 1;$

Fie $j \in \Gamma^{-} i$ a.i. $\lambda_j + l_{ji} = \lambda_i;$

$d(k) = j;$

$i = j;$

c) Criteriul de stop:

Dacă $i = 1$ atunci STOP

altfel goto b)

Algoritmii matriceali pentru valori minime

- se calculează valorile minime ale drumurilor între oricare două vârfuri $i,\ j \in X$ ale grafului G

- se folosește programarea dinamică

- fie $L^* = (l_{ij}^*)$ matricea cu elementele definite astfel

$$l_{ij} = \begin{cases} \min\{l(\mu) |\ \mu \textit{ drum de la } i \textit{ la } j\},\ i,\ j \in X,\ j \textit{ descendent al lui } i \\ \infty,\ \textit{altfel} \end{cases}$$

- $V = (v_{ij})_{i,\ j \in X}$ matricea valorilor

- se construiesc matricile $V^{(k)}$ astfel:

$$V^{(1)} = V\ ,\ V^{(k+1)} = (v_{ij}^{(k+1)})_{i,j \in X},\ v_{ij}^{(k+1)} = \min\{v_{ij}^{(k)},\ v_{ik}^{(k)} + v_{kj}^{(k)}\}$$

- elementul $v_{ij}^{(k+1)}$ = valoarea minimă a drumurilor de la i la j care folosesc nodurile intermediare din mulțimea $\{1, 2, ..., k\}$

- deci $V^{(n+1)} = L^*$

Algoritmul Floyd-Warshall-Hu

- determină matricea valorilor minime ale drumurilor între oricare două noduri i și j ale unui graf care nu conține circuite de valoare negativă

a) Inițializări: $v_{ij}^1 = \begin{cases} 0, & i = j \\ l_{ij}, & j \in \Gamma i \\ \infty, & \textit{altfel} \end{cases}$

b) Iterația de bază

pentru $k = 1, n$ execută

pentru $i = 1, n$ execută

pentru $j = 1, n$ execută

$$v_{ij}^{(k+1)} = \min\{v_{ij}^{(k)},\ v_{ik}^{(k)} + v_{kj}^{(k)}\}$$

- se poate renunța la a construi o succesiune de matrici $V^{(k)}$ și modificările să se facă în matricea valorilor

Algoritmul Floyd-Warshall-Hu

- algoritm care determină matricea valorilor minime ale drumurilor între oricare două vârfuri ale unui graf sau depistează existența unui circuit de valoare negativă

$k = 1$;

a) $i = 1$;

b) dacă $(v_{ik} = \infty)$ atunci goto c);
dacă $(v_{ik} + v_{ki} < 0)$ atunci
tipărește "există circuit de valoare negativă";
STOP;
pentru $j = 1, n$ execută
$v_{ij} = \min\{v_{ij},\ v_{ik} + v_{kj}\}$

c) $i = i + 1$;
dacă $(i \leq n)$ atunci goto b);
$k = k + 1$;
dacă $(k \leq n)$ atunci goto a);

Algoritmul Floyd-Warshall-Hu

- determină matricea valorilor minime ale drumurilor între oricare două noduri i și j ale unui graf care nu conține circuite de valoare negativă și drumurile respective

a) Inițializări: $v_{ij}^1 = \begin{cases} 0, & i = j \\ l_{ij}, & j \in \Gamma i \\ \infty, & \textit{altfel} \end{cases}$, $t_{ij} = \begin{cases} i, & j \in \Gamma i \\ 0, & \textit{altfel} \end{cases}$

b) Iterația de bază

pentru $k = 1, n$ execută

pentru $i = 1, n$ execută

pentru $j = 1, n$ execută

dacă $v_{ij} > v_{ik} + v_{kj}$ atunci

$v_{ij} = v_{ik} + v_{kj}$;

$t_{ij} = k$;

Drumuri de valoare maximă

- algoritmul Bellman- Kalaba se poate transforma în algoritm care să depisteze circuite (cicluri) într-un graf oarecare

- valori maxime ale drumurilor într-un graf există dacă graful nu conține circuite de valoare pozitivă

- pentru determinarea valorilor maxime ale drumurilor se pot aplica algoritmii anteriori cu următoarele modificări:

  - la inițializarea valorilor în locul lui $+\infty$ se utilizează valoarea $-\infty$;
  - în condițiile de test se înlocuiește "*min*" și " $<$ " cu "*max*" respectiv " $>$ "

Algoritmul Bellman-Kalaba pentru valori maxime

- determină valorile maxime ale drumurilor de la vârful 1 la $x \in X$ în G oarecare sau detectează existența unui circuit de valoare pozitivă

a) Inițializări: $k = 1;\ \lambda_1^k = 0,\ \lambda_j^k = v_{1j} = \begin{cases} l_{1j},\ (1,j) \in U \\ 0,\ i = j \\ -\infty,\ altfel \end{cases},\ j = \overline{2,n}\ ;$

b) Iterația de bază:

$$\lambda_j^{k+1} = \begin{cases} 0,\ j = 1 \\ \max\{\lambda_i + v_{ij} |\ i = \overline{1,n}\},\ j = \overline{2,n} \end{cases} \quad ; \quad k = k + 1;$$

c) Criteriul de stop

dacă $(\lambda_j^k = \lambda_j^{k-1},\ j = \overline{1,n})$ atunci STOP;

dacă $(k \leq m)$ atunci goto b)

altfel tipărește "există circuit de valoare negativă";

STOP;

Drumuri de valoare minimă în cazul $l(\mu) = \prod_{u \in \mu} l(u)$

- determină valorile minime ale drumurilor de la vârful 1 la $x \in X$ în G oarecare sau detectează existența unui circuit de valoare subunitară

a) Inițializări: $k = 1;\ \lambda_1^k = 1,\ \lambda_j^k = v_{1j} = \begin{cases} l_{1j},\ (1,j) \in U \\ 1,\ i = j \\ -\infty,\ altfel \end{cases},\ j = \overline{2,n}\ ;$

b) Iterația de bază:

$$\lambda_j^{k+1} = \begin{cases} 1,\ j = 1 \\ \min\{\lambda_i \cdot v_{ij} |\ i = \overline{1,n}\},\ j = \overline{2,n} \end{cases} \quad ; \quad k = k + 1;$$

c) Criteriul de stop

dacă $(\lambda_j^k = \lambda_j^{k-1},\ j = \overline{1,n})$ atunci STOP;

dacă $(k \leq m)$ atunci goto b)

altfel tipărește "există circuit de valoare subunitară";