Algoritmi de găsire a drumului optim

Din cauza varietăţii nelimitate a grafurilor posibile, nu există un algoritm care să rezolve orice problemă în timp util, dar s-au elaborat o mulţime de algoritmi, fiecare fiind cel mai eficace în anumite cazuri. Aceşti algoritmi pot fi grupaţi în cinci categorii:

1. Algoritmi prin calcul matricial (Bellman-Kalaba, I. Tomescu, Bellman-Schimbell);
2. Algoritmi prin ajustări succesive: (Ford);
3. Algoritmi prin inducţie (Dantzig);
4. Algoritmi prin ordonare prealabilă a vârfurilor grafului;
5. Algoritmi prin extindere selectivă (Dijkstra).

În continuare vom prezenta patru dintre aceşti algoritmi.

A. Algoritmul lui Bellman - Kalaba

Algoritmul se aplică în grafuri finite care nu au circuite de valoare negativă (pentru o problemă de minim) sau care nu au circuite de valoare pozitivă (într-o problemă de maxim) şi găseşte drumurile de valoare minimă (maximă) de la toate nodurile grafului la un nod oarecare, fixat. Dacă dorim să cunoaştem drumurile de valoare minimă (maximă) între oricare două noduri vom aplica algoritmul, pe rând, pentru fiecare nod al grafului.

Fie $G = \{x_1, x_2, \ldots, x_n\}$ un graf orientat finit. Presupunem (fără a restrânge generalitatea, că am numerotat nodurile astfel încât nodul spre care căutăm drumurile de valoare minimă (maximă) de la celelalte noduri să fie x_n.

Pasul 1. Se construieşte matricea pătratică M cu dimensiunea egală cu numărul de noduri ale grafului ale cărei elemente sunt:

$$m_{ij} = \begin{cases} \text{valoarea arcului } (x_i, x_j) & \text{daca exista arcul } (x_i, x_j) \text{ si } i \neq j \\ 0 & \text{daca } i = j \\ +\infty \text{ (într - o problema de minim)} \\ -\infty \text{ (într - o problema de maxim)} & \text{daca nu exista arcul } (x_i, x_j) \end{cases}$$

Pasul 2. Se adaugă succesiv liniile L_i la matricea M, elementele acestora calculându-se prin relaţiile de recurenţă:

1. $L_{1j} = m_{jn}$ $j = 1,\ldots,n$ (prima linie este ultima coloană, transpusă, a matricii M)
2. $L_{ij} = \min (L_{i-1,j}, \min\limits_{k=1,n} (m_{jk} + L_{i-1,k}))$ într-o problemă de minim

   sau $L_{ij} = \max (L_{i-1,j}, \max\limits_{k=1,n} (m_{jk} + L_{i-1,k}))$ într-o problemă de maxim

Pasul 3. După calcularea fiecărei linii noi se compară elementele ei cu cele ale precedentei:

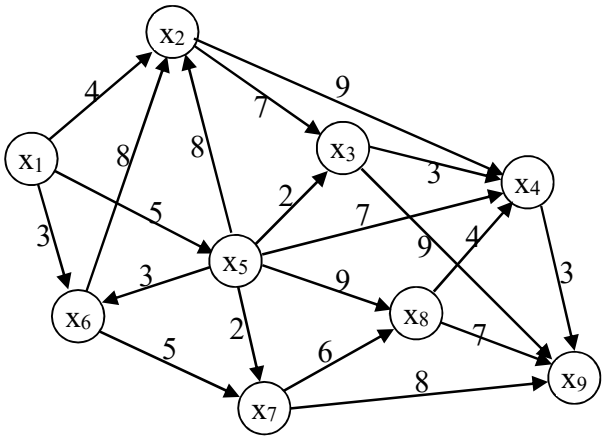
- Dacă $L_{ij} = L_{i-1,j}$ pentru orice $j = 1,\ldots,n$ atunci se opreşte recurenţa şi ultima linie calculată conţine valorile minime ale drumurilor de la celelalte noduri la nodul x_n.

- Dacă există cel puţin un indice j cu $L_{ij} \neq L_{i-1,j}$ se trece la calcularea noii linii L_{i+1}

Pasul 4. Pentru găsirea drumului care dă valoarea minimă de la un nod x_j la nodul x_n se găsesc, începând înapoi de la ultima linie, pe care s-au obţinut valorile finale, notată L_f, nodurile $x_{k_1}, x_{k_2}, ..., x_{k_r}$ care formează drumul căutat, unde $x_{k_1} = x_j$, $x_{k_r} = x_n$ şi fiecare alt indice k_{i+1} este cel pentru care s-a obţinut minimul(maximul) de pe poziţia k_i al liniei L_i.

Observaţie: Pentru grafuri foarte mari, algoritmul necesită un volum mare de memorie, prin necesitatea memorării matricei M, care este greu de manipulat. Chiar dacă din cele n^2 arce posibile graful ar avea doar un procent foarte mic matricea grafului va avea tot n^2 poziţii de memorat şi analizat.

Exemplu: Presupunem dat graful orientat de mai jos, în care se doreşte găsirea drumului de valoare minimă de la nodul x_1 la nodul x_9.



Matricea M va fi $\begin{pmatrix} 0 & 4 & \infty & \infty & 5 & \infty & \infty & \infty & \infty \\ \infty & 0 & 7 & 9 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 3 & \infty & \infty & \infty & \infty & 9 \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & 3 \\ \infty & 8 & 2 & 7 & 0 & 3 & 2 & 9 & \infty \\ \infty & 8 & \infty & \infty & \infty & 0 & 5 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & 6 & 8 \\ \infty & \infty & \infty & 4 & \infty & \infty & \infty & 0 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

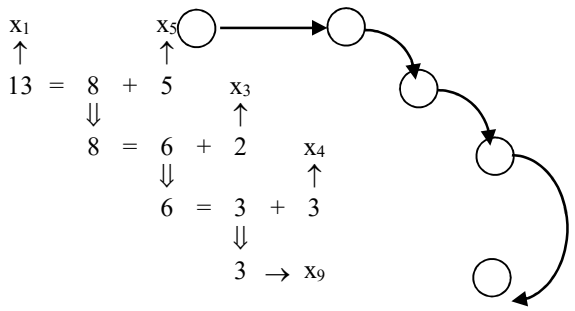
iar după calcularea liniilor L_i obţinem:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_1	0	4	∞	∞	5	∞	∞	∞	∞
x_2	∞	0	7	9	∞	∞	∞	∞	∞
x_3	∞	∞	0	3	∞	∞	∞	∞	9
x_4	∞	∞	∞	0	∞	∞	∞	∞	3
x_5	∞	8	2	7	0	3	2	9	∞
x_6	∞	8	∞	∞	∞	0	5	∞	∞
x_7	∞	∞	∞	∞	∞	∞	0	6	8
x_8	∞	∞	∞	4	∞	∞	∞	0	7

x_9	∞	∞	∞	∞	∞	∞	∞	∞	0
L_1	∞	∞	9	3	∞	∞	8	7	0
L_2	∞	12	6	3	10	13	8	7	0
L_3	15	12	6	3	8	13	8	7	0
L_4	13	12	6	3	8	13	8	7	0
L_5	13	12	6	3	8	13	8	7	0

Deoarece $L_4 = L_5$ oprim calcularea liniilor după calcularea liniei 5. În această linie se află valorile celor mai scurte de la toate nodurile la nodul x_9. Drumul dorit de noi ($x_1 \rightarrow x_9$) are valoarea dată de prima poziţie a liniei 5, fiind egal cu 13.

Pentru a găsi acest drum, plecăm înapoi de la linia 4 şi avem:



B. Algoritmul lui Ford simplificat

Algoritmul lui Ford simplificat ***se aplică doar în grafuri care nu admit circuite***. Cu ajutorul lui se găseşte drumul de valoare optimă între două noduri fixate x_i şi x_j. Printr-o eventuală renumerotare a nodurilor putem presupune că nodul de la care porneşte drumul este x_1, care va fi numit nod iniţial, iar nodul la care se termină este x_n, numit nod final.

Algoritmul este:

Pasul 1. I se dă vârfului iniţial valoarea 0 (zero): $w(x_0) = 0$

Pasul 2. Se construieşte mulţimea A formată din nodul iniţial: $A = \{x_1\}$

Pasul 3. Se analizează nodurile din afara mulţimii A.

- Dacă există noduri în care se poate ajunge prin arce directe doar de la nodurile mulţimii A, acestea se adaugă la mulţimea A, cu valoarea:

$$w(x_i) = \min_{\exists(x_j, x_i)}^{x_j} \left(w(x_j) + v(x_j, x_i)\right), \text{ în problemele de minim}$$

sau

$$w(x_i) = \max_{\exists(x_j, x_i)}^{x_j} \left(w(x_j) + v(x_j, x_i)\right), \text{ în problemele de maxim}$$

apoi se trece la pasul 4

- Dacă nu există nici un nod de acest tip atunci nu există nici un drum de la x_1 la x_n. STOP

Pasul 4. Se analizează mulţimea A:

– Dacă $x_n \in A$ atunci valoarea sa reprezintă valoarea drumului de valoare optimă de la x_1 la x_n. Pentru găsirea acestui drum se porneşte înapoi de la nodul final x_n şi se găsesc nodurile $x_{k_1}, x_{k_2}, ..., x_{k_r}$ care formează drumul căutat, unde $x_{k_1} = x_n$, $x_{k_r} = x_1$ şi fiecare alt indice k_{i+1} este cel pentru care:

$$w(x_{k_{i+1}}) + v(x_{k_{i+1}}, x_{k_i}) = w(x_{k_i}) \quad \text{STOP}$$

– Dacă $x_n \notin A$ se reia algoritmul de la pasul 3.

Exemplu: Pentru acelaşi graf şi aceeaşi pereche de noduri din exemplul rezolvat cu algoritmul lui Bellman-Kalaba vom avea succesiv:

pas1: $w(x_1) = 0$
pas2: $A = \{x_1\}$
pas3: Nodurile în care se poate ajunge doar din x_1: $\{x_5\} \neq \emptyset$
$w\{x_5) = \min(w(x_1) + v(x_1,x_5)) = 0 + 5 = 5$
pas4: $x_9 \notin A$
pas3: $A = \{x_1,x_5\}$ şi nodurile în care se poate ajunge prin arce directe doar din x_1 şi x_5 sunt: $\{x_6\} \neq \emptyset$
$w\{x_6) = \min(w(x_1) + v(x_1,x_6), w(x_5) + v(x_5,x_6)) = \min(0 + 3 , 5 + 3) = 3$
pas4: $x_9 \notin A$
pas3: $A = \{x_1,x_5,x_6\}$ şi nodurile în care se poate ajunge prin arce directe doar din x_1, x_5 şi x_6 sunt: $\{x_2,x_7\} \neq \emptyset$
$w\{x_2) = \min(w(x_1) + v(x_1,x_2), w(x_5) + v(x_5,x_2), w(x_6) + v(x_6,x_2)) = \min(0 + 4,5 + 8,3 + 8) = 4$
$w\{x_7) = \min(w(x_5) + v(x_5,x_7), w(x_6) + v(x_6,x_7)) = \min(5 + 2,3 + 5) = 7$
pas4: $x_9 \notin A$
pas3: $A = \{x_1,x_2,x_5,x_6,x_7\}$ şi nodurile în care se poate ajunge prin arce directe doar din x_1, x_2, x_5, x_6 şi x_7 sunt: $\{x_3,x_8\} \neq \emptyset$
$w\{x_3) = \min(w(x_2) + v(x_2,x_3), w(x_5) + v(x_5,x_3)) = \min(4 + 7,5 + 2) = 7$
$w\{x_8) = \min(w(x_5) + v(x_5,x_8), w(x_7) + v(x_7,x_8)) = \min(5 + 9,7 + 6) = 13$
pas4: $x_9 \notin A$
pas3: $A = \{x_1,x_2,x_3,x_5,x_6,x_7,x_8\}$ şi nodurile în care se poate ajunge prin arce directe doar din x_1, x_2,x_3,x_5, x_6, x_7 şi x_8 sunt: $\{x_4\} \neq \emptyset$
$w\{x_4) = \min(w(x_2) + v(x_2,x_4), w(x_3) + v(x_3,x_4),w(x_5) + v(x_5,x_4), w(x_8) + v(x_8,x_4)) = \min(4 + 9,7 + 3,5 + 7,13 + 4) = 10$
pas4: $x_9 \notin A$
pas3: $A = \{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8\}$ şi nodurile în care se poate ajunge prin arce directe doar din x_1, x_2, x_3, x_4, x_5, x_6, x_7 şi x_8 sunt: $\{x_9\} \neq \emptyset$
$w\{x_9) = \min(w(x_3) + v(x_3,x_9), w(x_4) + v(x_4,x_9), w(x_7) + v(x_7,x_9), w(x_8) + v(x_8,x_9)) = \min(7 + 9, 10 + 3, 7 + 8, 13 + 7) = 13$
pas4: $x_9 \in A$ şi urmează să găsim drumul care are lungimea 13.
Avem succesiv:

$$w(x_9) = w(x_4) + v(x_4,x_9)$$
$$w(x_4) = w(x_3) + v(x_3,x_4)$$
$$w(x_3) = w(x_5) + v(x_5,x_3)$$

$$w(x_5) = w(x_1) + v(x_1,x_5)$$

deci drumul căutat este: $x_1 \rightarrow x_5 \rightarrow x_3 \rightarrow x_4 \rightarrow x_9$

Observaţia 1. Dacă graful are un circuit atunci se poate demonstra uşor că nu vom putea da valoare nici unui nod al acestuia şi dacă există vreun drum de la x_1 la x_n care trece prin unul din nodurile circuitului nu vom putea da valoare nici lui x_n, cu toate că există drum de la x_1 la x_n.

Observaţia 2: Algoritmul necesită pentru memorare şi manipulare doar cunoaşterea, pentru fiecare nod, a nodurilor spre care "pleacă" arce din acesta şi valorile acestor arce, fiind mult mai uşor de aplicat sau implementat pe calculator. El are însă dezavantajul că se poate aplica doar în grafuri fără circuite.

C. Algoritmul Ford generalizat

Algoritmul lui Ford generalizat a fost creat cu scopul de a putea găsi drumul optim şi în grafurile care au circuite. Cu ajutorul lui se găseşte drumul de valoare optimă între două noduri fixate x_i şi x_j. Printr-o eventuală renumerotare a nodurilor putem presupune că nodul de la care porneşte drumul este x_1, care va fi numit nod iniţial, iar nodul la care se termină este x_n, numit nod final.

Algoritmul este:

Pasul 1. I se dă vârfului iniţial valoarea 0 (zero): $w(x_0) = 0$ şi tuturor celelalte valoarea $+\infty$ (într-o problemă de minim) sau $-\infty$ (într-o problemă de maxim).

Pasul 2. În ordinea crescătoare a indicilor nodurilor se calculează pentru fiecare nod, pe bază fostelor valori, noile valori cu formula:

$$w^*(x_i) = \min\left(w(x_i),\ \min_{\exists(x_j,x_i)}^{x_j} \left(w(x_j) + v(x_j,x_i) \right) \right) \text{ în problemele de minim}$$

$$\textbf{sau}\ \ w^*(x_i) = \max\left(w(x_i),\ \max_{\exists(x_j,x_i)}^{x_j} \left(w(x_j) + v(x_j,x_i) \right) \right) \text{ în problemele de maxim}$$

Pasul 3. Se compară noile valori $w^*(x_i)$ cu fostele valori $w(x_i)$:

- Dacă $w^*(x_i) = w(x_i)$ pentru orice nod x_i atunci:
  - dacă $w(x_n) < \infty$ (la problema de minim) sau $w(x_n) > -\infty$ (la problema de maxim), valoarea nodului x_n reprezintă valoarea drumului de valoare minimă(maximă) de la x_1 la x_n. Pentru găsirea acestui drum se porneşte înapoi de la nodul final x_n şi se găsesc nodurile $x_{k_1}, x_{k_2}, ..., x_{k_r}$ care formează drumul căutat, unde $x_{k_1} = x_n$, $x_{k_r} = x_1$ şi fiecare alt indice k_{i+1} este cel pentru care:
    $$w(x_{k_{i+1}}) + v(x_{k_{i+1}}, x_{k_i}) = w(x_{k_i})\ \ \text{STOP}$$
  - dacă $w(x_n) = +\infty$ ($-\infty$) atunci nu există nici un drum de la x_1 la x_n. STOP
- Dacă există cel puţin un nod pentru care $w^*(x_i) < w(x_i)$ se reia algoritmul de la pasul 2 pentru noile valori ale vârfurilor.

Observaţie: Algoritmul poate găsi drumul şi în grafuri cu circuite dar este evident mult mai lent decât cel simplificat. Pentru scurtarea duratei de execuţie se poate modifica algoritmul în sensul că o valoare nou calculată a unui vârf va fi folosită imediat ca atare la calculul noilor valori ale celorlalte, nu doar după ce se calculează noile valori ale tuturor vârfurilor.

D. Algoritmul lui Dijkstra

În algoritmul Ford simplificat, pentru a găsi valoarea nodului final, deci a drumului minim, plecăm de la nodul iniţial în toate direcţiile posibile, păstrând de fiecare dată toate nodurile analizate. Acest fapt duce la un consum inutil de timp, deoarece foarte multe din aceste noduri nu vor face parte din drumul optim. Pentru a elimina acest neajuns, algoritmul lui Dijkstra încearcă să păstreze, la fiecare iteraţie, mulţimea minimă de noduri care să le conţină pe toate cele care vor forma efectiv drumul optim. În plus, algoritmul se poate aplica şi în drumuri cu circuite. Ca un minus este faptul că se aplică doar la probleme de minim. Algoritmul are următorii paşi:

Pasul 1. I se dă vârfului iniţial valoarea 0 (zero): $w(x_0) = 0$
Pasul 2. Se construieşte mulţimea A formată din nodul iniţial: $A = \{x_1\}$
Pasul 3. Se analizează nodurile din afara mulţimii A.

- Dacă există noduri ***în care se poate ajunge prin arce directe de la noduri din A*** (nu ***doar*** de la nodurile mulţimii A, ca la algoritmul lui Ford simplificat) se calculează pentru toate acestea:

  $$w(x_i) = \min_{\substack{x_j \in A \\ \exists (x_j, x_i)}} \left(w(x_j) + v(x_j, x_i) \right)$$ în problemele de minim

  dar, spre deosebire de algoritmul lui Ford simplificat, *se adaugă la mulţimea A **doar** cel pentru care se obţine valoarea minimă*, apoi se trece la pasul 4.
- Dacă nu există nici un nod de acest tip atunci nu există nici un drum de la x_1 la x_n. STOP

Pasul 4. Se analizează mulţimea A:

- Dacă $x_n \in A$ atunci valoarea sa reprezintă valoarea drumului de valoare optimă de la x_1 la x_n. Pentru găsirea acestui drum se porneşte înapoi de la nodul final x_n şi se găsesc nodurile $x_{k_1}, x_{k_2}, ..., x_{k_r}$ care formează drumul căutat, unde $x_{k_1} = x_n$, $x_{k_r} = x_1$ şi fiecare alt indice k_{i+1} este cel pentru care:

  $$w(x_{k_{i+1}}) + v(x_{k_{i+1}}, x_{k_i}) = w(x_{k_i}) \quad \text{STOP}$$
- Dacă $x_n \notin A$ se reia algoritmul de la pasul 3.

Exemplu Vom aplica algoritmul la acelaşi graf folosit la ceilalţi algoritmi, pentru a putea face comparaţii:

pas1: $w(x_1) = 0$
pas2: $A = \{x_1\}$
pas3: Nodurile în care se poate ajunge şi din x_1: $\{x_2, x_5, x_6\} \neq \varnothing$

$w\{x_2) = \min(w(x_1) + v(x_1,x_2)) = 0 + 4 = 4$
$w\{x_5) = \min(w(x_1) + v(x_1,x_5)) = 0 + 5 = 5$
$w\{x_6) = \min(w(x_1) + v(x_1,x_6)) = 0 + 3 = 3$
$\min(w\{x_2),w\{x_5),w\{x_6)) = w\{x_6) = 3$

pas4: $x_9 \notin A$

pas3: $A = \{x_1,x_6\}$ şi nodurile în care se poate ajunge prin arce directe din x_1 sau x_6 sunt: $\{x_2,x_5,x_7\} \neq \emptyset$
$w\{x_2) = \min(w(x_1) + v(x_1,x_2), w(x_6) + v(x_6,x_2)) = \min(0 + 4 , 3 + 8) = 4$
$w\{x_5) = \min(w(x_1) + v(x_1,x_5)) = \min(0 + 5) = 5$
$w\{x_7) = \min(w(x_6) + v(x_6,x_7)) = \min(3 + 5) = 8$
$\min(w\{x_2),w\{x_5),w\{x_7)) = w\{x_2) = 4$

pas4: $x_9 \notin A$

pas3: $A = \{x_1,x_2,x_6\}$ şi nodurile în care se poate ajunge prin arce directe din x_1, x_2 sau x_6 sunt: $\{x_3,x_4,x_5,x_7\} \neq \emptyset$
$w\{x_3) = \min(w(x_2) + v(x_2,x_3)) = \min(4 + 7) = 11$
$w\{x_4) = \min(w(x_2) + v(x_2,x_4)) = \min(2 + 9) = 11$
$w\{x_5) = \min(w(x_1) + v(x_1,x_5)) = \min(0 + 5) = 5$
$w\{x_7) = \min(w(x_6) + v(x_6,x_7)) = \min(3 + 5) = 0$
$\min(w\{x_3),w\{x_4),w\{x_5),w\{x_7)) = w\{x_5) = 5$

pas4: $x_9 \notin A$

pas3: $A = \{x_1,x_2,x_5,x_6\}$ şi nodurile în care se poate ajunge prin arce directe din x_1, x_2, x_5, x_6 şi x_7 sunt: $\{x_3,x_4,x_7,x_8\} \neq \emptyset$
$w\{x_3) = \min(w(x_2) + v(x_2,x_3), w(x_5) + v(x_5,x_3)) = \min(4 + 7,5 + 2) = 7$
$w\{x_4) = \min(w(x_2) + v(x_2,x_4), w(x_5) + v(x_5,x_4)) = \min(4 + 9,5 + 7) = 12$
$w\{x_7) = \min(w(x_5) + v(x_5,x_7), w(x_6) + v(x_6,x_7)) = \min(5 + 2,3 + 5) = 7$
$w\{x_8) = \min(w(x_5) + v(x_5,x_8)) = \min(5 + 9) = 14$
$\min(w\{x_3),w\{x_4),w\{x_7),w\{x_8)) = w\{x_3) = w\{x_7) = 7$

pas4: $x_9 \notin A$

pas3: $A = \{x_1,x_2,x_3,x_5,x_6,x_7\}$ şi nodurile în care se poate ajunge prin arce directe din x_1, x_2, x_3, x_5, x_6, şi x_7 sunt: $\{x_4,x_8,x_9\} \neq \emptyset$
$w\{x_4) = \min(w(x_2) + v(x_2,x_4), w(x_3) + v(x_3,x_4),w(x_5) + v(x_5,x_4)) = \min(4 + 9,7 + 3,5 + 7) = 10$
$w\{x_8) = \min(w(x_5) + v(x_5,x_8), w(x_7) + v(x_7,x_8)) = \min(5 + 9,7 + 6) = 13$
$w\{x_9) = \min(w(x_3) + v(x_3,x_9), w(x_7) + v(x_7,x_9)) = \min(7 + 9,7 + 8) = 15$
$\min(w\{x_4),w\{x_8),w\{x_9)) = w\{x_4) = 10$

pas4: $x_9 \notin A$

pas3: $A = \{x_1,x_2,x_3,x_4,x_5,x_6,x_7\}$ şi nodurile în care se poate ajunge prin arce directe din x_1, x_2, x_3, x_4, x_5, x_6, şi x_7 sunt: $\{x_8,x_9\} \neq \emptyset$
$w\{x_9) = \min(w(x_3) + v(x_3,x_9), w(x_4) + v(x_4,x_9), w(x_7) + v(x_7,x_9)) = \min(7 + 9,10 + 3,7+8)=13$
$w\{x_8) = \min(w(x_5) + v(x_5,x_8), w(x_7) + v(x_7,x_8)) = \min(5 + 9,7 + 6) = 13$
$\min(w\{x_8),w\{x_9)) = w\{x_8) = w\{x_9) = 13$

pas4: $x_9 \in A$ şi urmează să găsim drumul care are lungimea 13.

Avem succesiv:

$w(x_9) = w(x_4) + v(x_4,x_9)$
$w(x_4) = w(x_3) + v(x_3,x_4)$

$$w(x_3) = w(x_5) + v(x_5,x_3)$$
$$w(x_5) = w(x_1) + v(x_1,x_5)$$

deci drumul căutat este: $x_1 \rightarrow x_5 \rightarrow x_3 \rightarrow x_4 \rightarrow x_9$