Shortest paths

Shortest path problems

- Let G=(X,U,v) with:
  - $X=\{x_0, x_1, x_2, \ldots, x_{n-1}\}$ et $v : U \rightarrow \Re$

- Length of a path: number of arcs composing the path
- Weight(value) of a path : sum of weights of its arcs

- Some path from x_i to x_k is of minimal weight if its weight is the smallest one (<= to all others paths from x_i to x_k.)
  - We call this the shortest path

Ps. all paths and cycles are assumed directed.

The shortest path problems

Three types of problems:

- Given two vertices x_i and x_k, find the shortest path (when such a path exists);
- Given a vertex x_s, find all shortest paths (if they exist) from x_s to any other vertex x_i;
- Find the shortest paths for all couples of vertices in the graph.

Applications

- Subproblem for numerous optimisation problems.
- Applications to transport:
  - Vehicle routing problem;
- Applications in telecommunications, ATM…
- *etc.*

Some properties of shortest paths (I)

- Lemma 1. Any subpath of a shortest path is as well a shortest path.

  We assume below that there exists at least a path from x_0 to x_i for any i.

- Lemma 2. A necessary and sufficient condition such that, for any i, there exists a shortest path from x_0 to x_i is that graph G doesn't contain a negative cycle.

Some properties of shortest paths (II)

- Theorem 1. Let G be a graph without negative cycles and λ_i values of paths from x_0 to x_i. A necessary and sufficient condition such that $\{\lambda_i \,/\, 0 \leq i \leq n-1\}$ be the set of shortest paths values from x_0 is that:

1- $\lambda_0 = 0$;

2- $\lambda_j - \lambda_i \leq v_{ij}$, for all arc $(x_i, x_j) \in U$.

$\lambda_j - \lambda_i \leq v_{ij}$

Proof (hint):

NC. If for some arc $(x_i, x_j) \in U$. $\lambda_j - \lambda_i > v_{ij}$, **we have a shorter path than λ_j to x_j.**

SC: Let μ be a shortest path to x_j, then we write down the eq. for all arcs composing it and sum on them, we obtain $\lambda_j \leq \text{value}(\mu)$.

Corollary. The set of arcs (x_i, x_j) such that $\lambda_j - \lambda_i = v_{ij}$ is the set of arcs involved in shortest paths.

Shortest path algorithms

- FORD (or Bellman-Ford) algorithm:
  - Works for all weights given to arcs
  - O(n m)
  - Label correcting algorithm

$O(n) \leq O(m) \leq O(n^2)$

- DIJKSTRA algorithm:
  - Works for all non negative weights given to arcs
  - $O(n^2)$
  - Label setting algorithm

- BELLMAN algorithm:
  - Works in acyclic graphs
  - O(m)
  - Label setting algorithm

FORD Algorithm

(x_0, x_1) $\lambda_0 + v_{(x_0,x_1)} = 0+2=2$

$\lambda_1 = \infty$

Algorithm:

(i) *Initialization* $O(n)$

Poser $\lambda_0 = 0$ et $\lambda_i = +\infty$ pour i > 0.

(ii) *Edges examination*

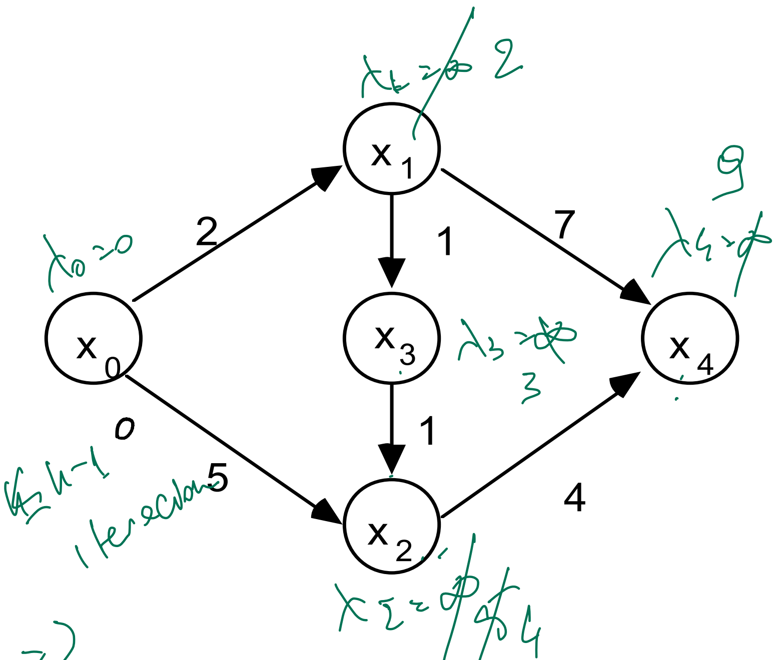
for each vertex x_i, check all (x_i,x_j) from x_i and substitute λ_j with $\lambda_i + v_{ij}$ when $\lambda_i + v_{ij} < \lambda_j$.

1 iteration $= O(m)$

$k \le n-1$ iterations $\Rightarrow O(mn)$

(iii) *Stop Test*

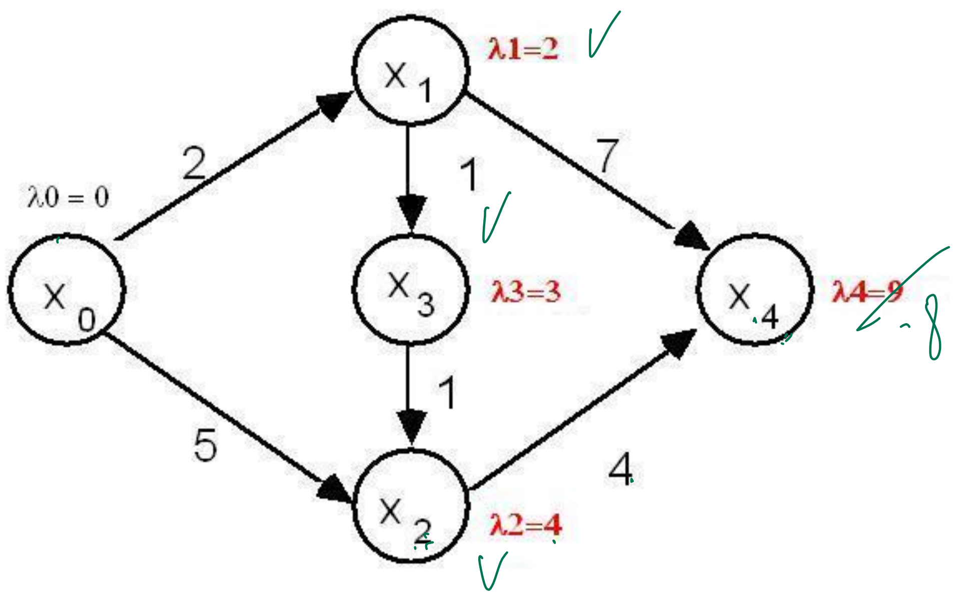
Iterate (ii) until some λ_j is updated in (ii).



An example

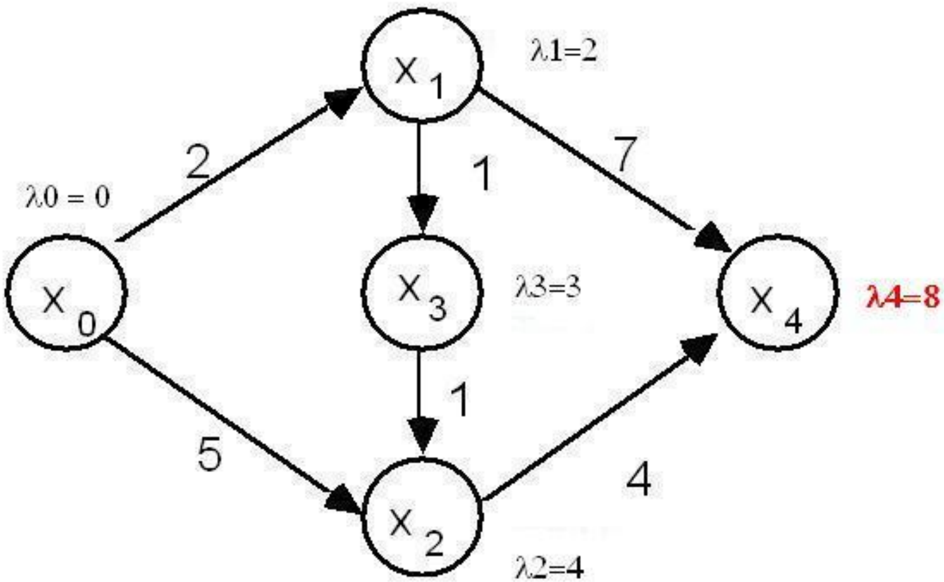
FORD Algorithm
an example

End of first iteration



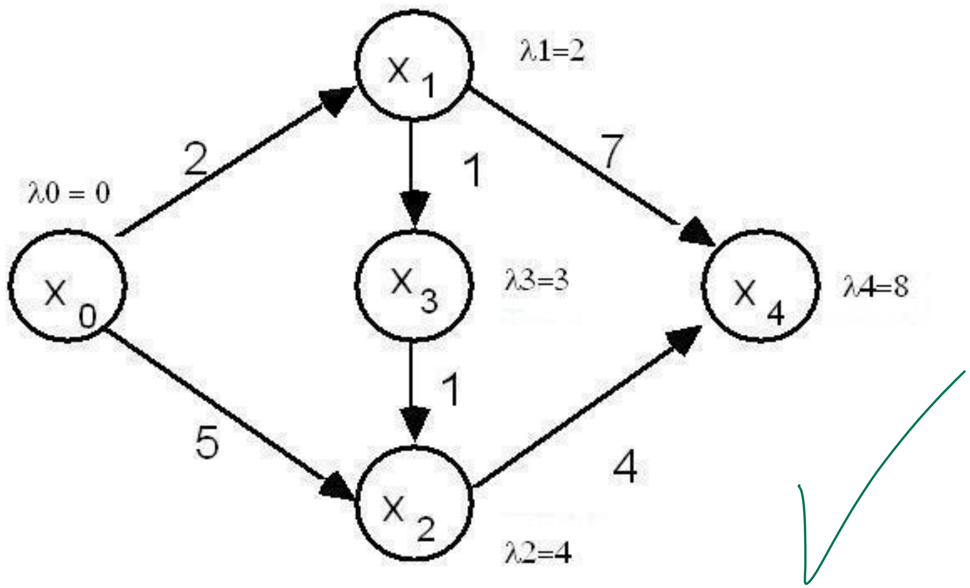
FORD Algorithm
an example

End of second iteration



FORD Algorithm
an example

Last iteration



Validity et complexity of Ford algorithm

<u>Theorem 2:</u> Ford algorithm computes values of the shortest path from x_0 when the graph is without negative circuits.

Proof by recurrence (hint):

- Set λ_i^{k*}, the min value of a path from $\mathbf{x_0}$ to $\mathbf{x_i}$ containing at most k arcs.
- Set λ_i^k, the value λ_i after k steps in the loop while.

Invariant:

At the end of k^{th} step, λ_i^k gives the value of a path from x_0 to x_i s.t. $\lambda_i^k \leq \lambda_i^{k*}$.

<u>Theorem 3:</u> The complexity of Ford algorithm is in $\boxed{O(nm)}$ where $n = |X|$ and $m = |U|$.

(A) $O(n)$

DIJKSTRA Algorithm

$O(n)$

Algorithm

(i) Set $S = \{x_0\}$, $\lambda_0 = 0$, $\lambda_i = v_{0i}$, if $(x_0, x_i) \in U$, and $\lambda_i = +\infty$, otherwise.

(ii) <u>While $S \neq X$ do:</u>

choose $x_i \in X - S$ of λ_i minimum. $O(n)$

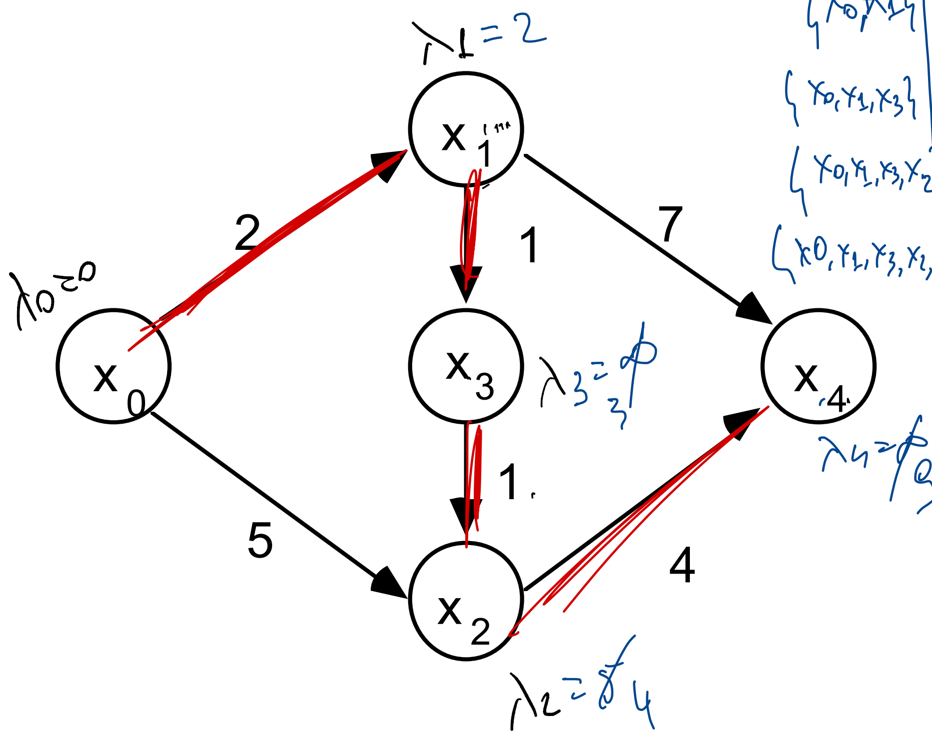
set $S = S + \{ x_i \}$.

For any $x_j \in (X - S)$, successor of x_i,

set: $\lambda_j = \min(\lambda_i + v_{ij}, \lambda_j)$. $O(n)$

$O(n) * n = O(n^2)$

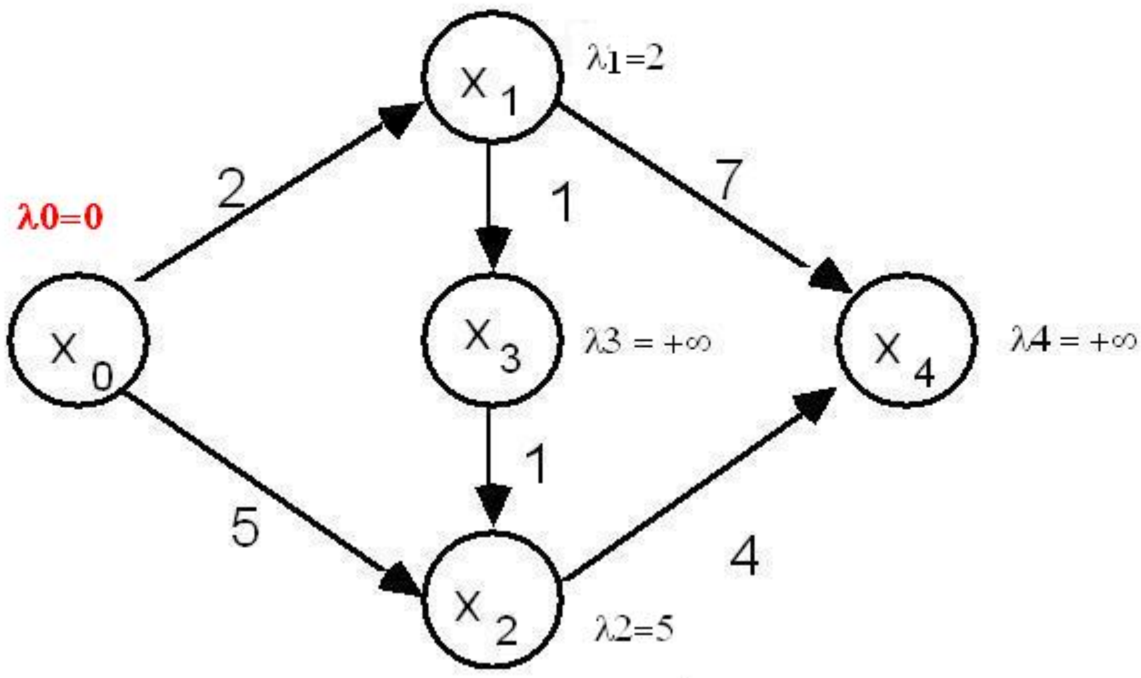
DIJKSTRA Algorithm
an example



S	λ_0	λ_1	λ_2	λ_3	λ_4
$\{x_0\}$	0	2	5	∞	∞
$\{x_0, x_1\}$	0	2	5	3	9
$\{x_0, x_1, x_3\}$	0	2	4	3	9
$\{x_0, x_1, x_3, x_2\}$	0	2	4	3	8
$\{x_0, x_1, x_3, x_2, x_4\}$	0	2	4	3	8

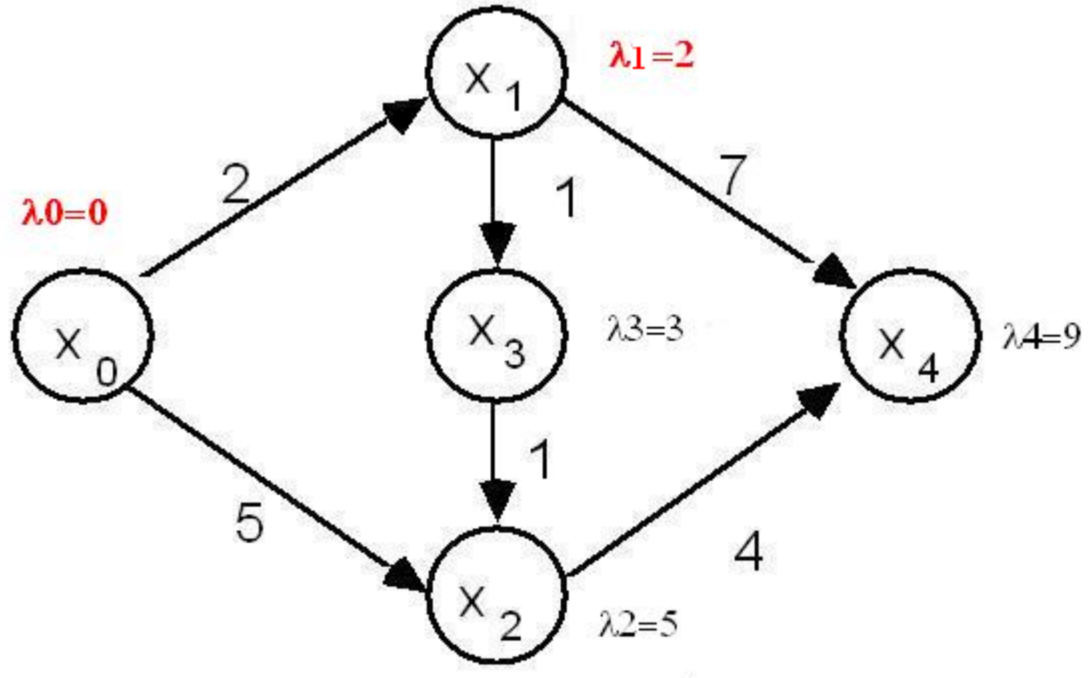
DIJKSTRA Algorithm
an example

End of the first iteration



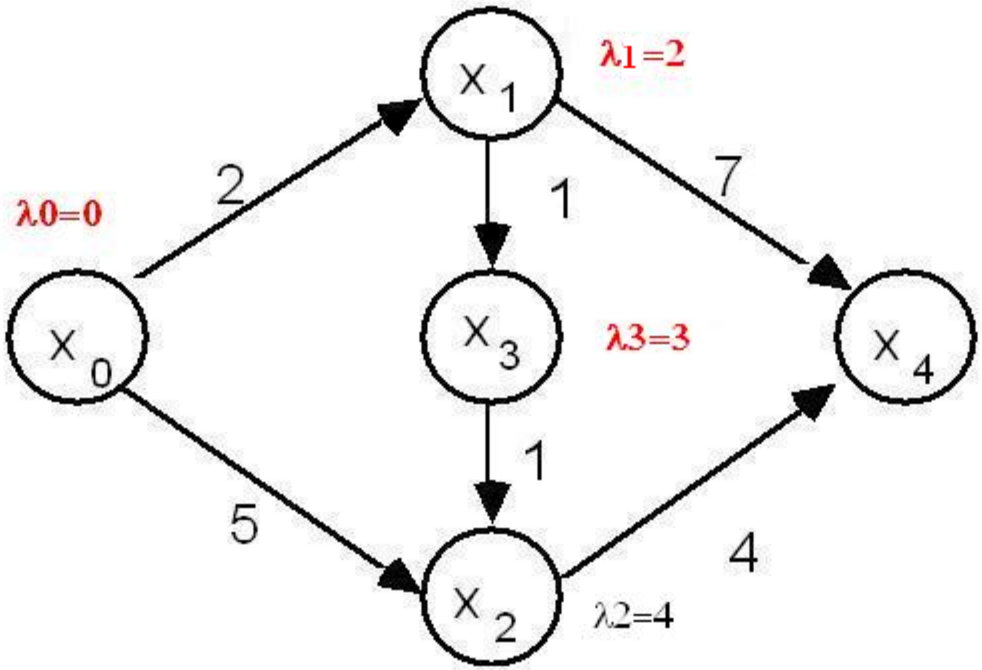
DIJKSTRA Algorithm
an example

End of the second iteration



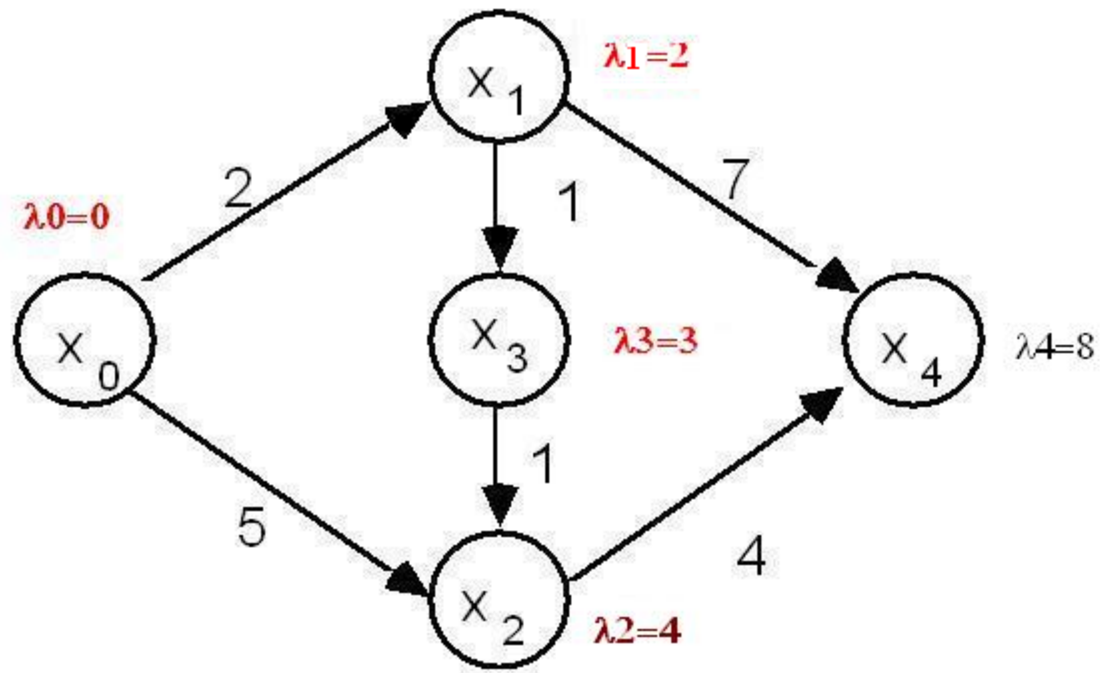
DIJKSTRA Algorithm
an example

End of the third iteration



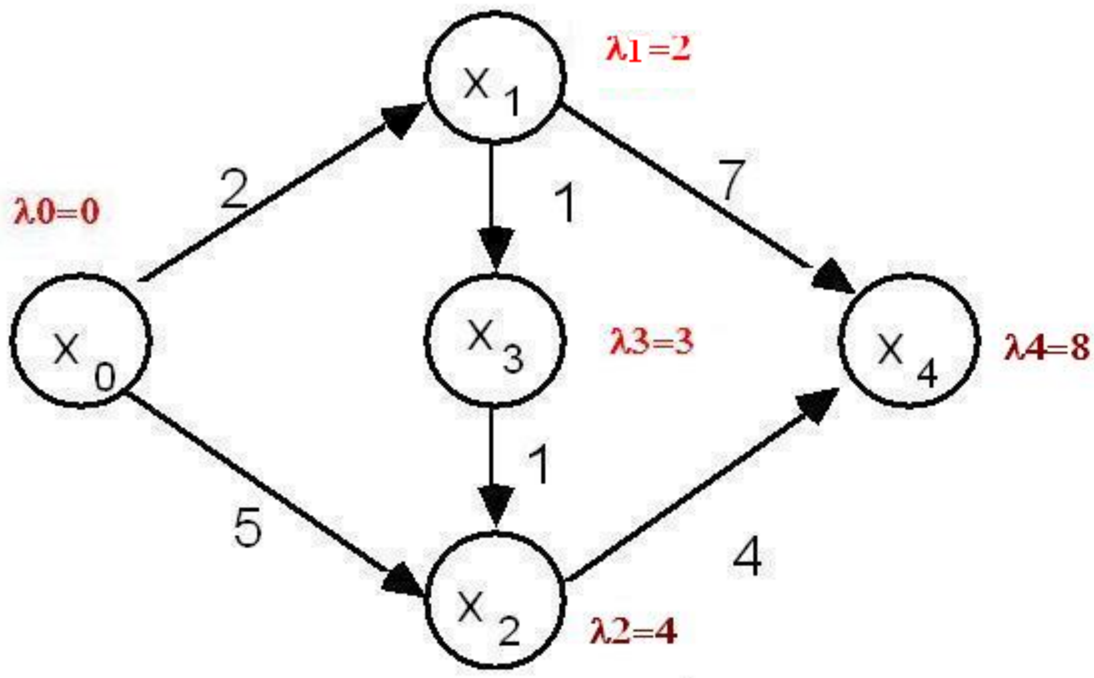
DIJKSTRA Algorithm
an example

End of the forth iteration



DIJKSTRA Algorithm
an example

End of the last iteration



Validity and complexity of Dijkstra algorithm

Theorem 4. λ_i obtained at the end of the algorithm are the shortest path values from x_0.

(valuations ≥ 0 : there are no negative cycles)

Proof by recurrence :

Invariant:

At the end of step **k**,

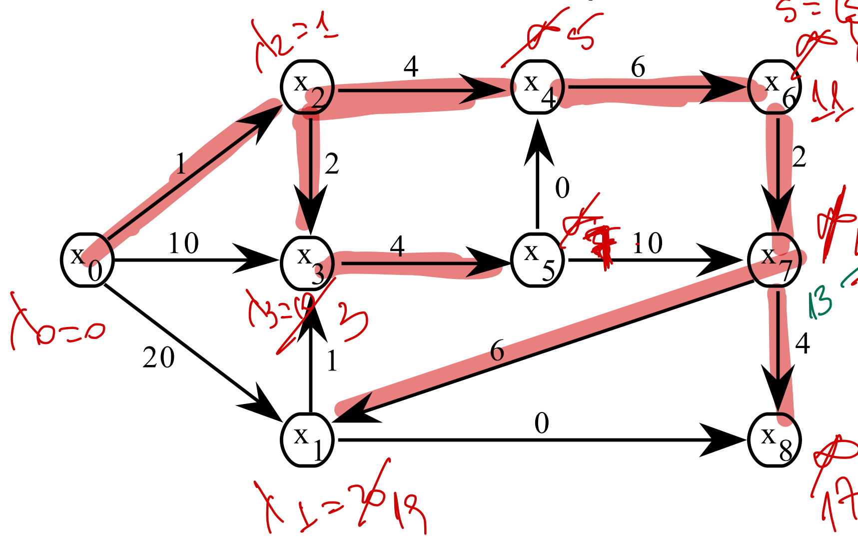
1- if $x_i \in S$: $\lambda_i = \lambda^*_i$.

2- if $x_i \notin S$: $\lambda_i = \min_{z \in U-(i) \cap S} \lambda_z + v_{zi}$

Lemma 3. Dijkstra algorithm is of complexity $O(n^2)$.

Exercise

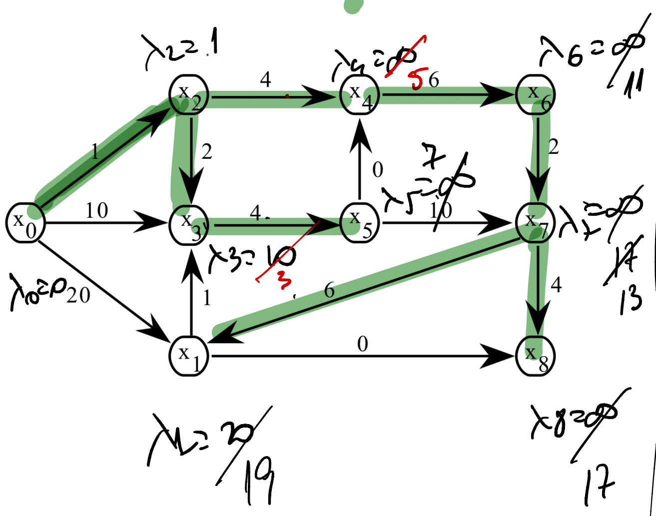
We wish to find the values of the minimal paths from x0.



S	λ_0	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
$S=\{x_0\}$	0	20	1	10	∞	∞	∞	∞	∞
$S=(x_0, x_2)$	0	20	1	3	5	∞	∞	∞	∞
$S=\{S+x_3\}$	0	20	1	3	5	7	∞	∞	∞
$\{S+x_4\}$	0	20	1	3	5	7	11	∞	∞
$\{S+x_5\}$	0	20	1	3	5	7	11	17	∞
$\{S+x_6\}$	0	20	1	3	5	7	11	13	∞
$\{S+x_7\}$	0	19	1	3	5	7	11	13	17
$\{S+x_8\}$	0	19	1	3	5	7	11	13	17
$\{S+x_1\}$	0	19	1	3	5	7	11	13	17

Apply DIJKSTRA algorithm. Write down the successive values of λi (if DIJKSTRA may be used), as well as a tree of minimal paths from x0.

Are the minimal paths unique? Justify your answer.



∃ (x_i, x_j) telle que

$$\boxed{\lambda_j - \lambda_i = v_{ij}}$$

$S \backslash \lambda$	λ_0	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
$\{x_0\}$	0	20	1	10	∞	∞	∞	∞	∞
$+\{x_2\}$	0	20	1	3	5	∞	∞	∞	∞
$+\{x_3\}$	0	20	1	3	5	7	∞	∞	∞
$+\{x_4\}$	0	20	1	3	5	7	11	∞	∞
$+\{x_5\}$	0	20	1	3	5	7	11	17	∞
$+\{x_6\}$	0	20	1	3	5	7	11	13	∞
$+\{x_7\}$	0	19	1	3	5	7	11	13	17
$+\{x_8\}$	0	19	1	3	5	7	11	13	17
$+\{x_1\}$	0	19	1	3	5	7	11	13	17

PROBLEM: SECOND SHORTEST (I)

Second shortest algorithm:

Begin

$O(m^2)$ $O(m^2)$ 1) Apply the Dijkstra algorithm to obtain the tree A of the shortest paths from 0 to i and the potentials λ (i) from 0 to i for all the vertices i of G.

(Note: We shall note γ (r, s) the path, if it exists, from r to s in A)

$O(m)$ 2) Determine γ (0, n-1) = (y0 = 0, y1, ..., yp = n-1).

3) Set value: = $+\infty$;

$O(n^2)$ For j: = 1 to p do $\leq n-1$

$O(n)$ for all k $\in$ (U-(yj) - (yj-1)) do

Begin

α: = λ (k) + v (k, yj) + (λ(n-1) - λ(yj));

if α < value then

value: = α; pivot1: = k; pivot2: = yj;

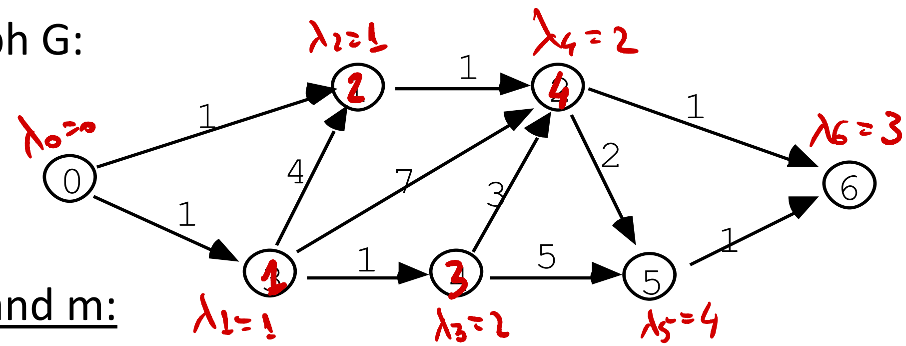
end;

$O(n)$ 4) **Second**: = γ (0, pivot1) + (pivot1, pivot2) + γ (pivot2, n-1).

end.

PROBLEM: SECOND SHORTEST (II)

1) Apply the algorithm to the following graph G:



2) Analysis of the complexity function of n and m:

It is assumed that the graph is coded by the queue of predecessors and successors.

2.1) What are the complexities of the phases 1, 2, 3 and 4 of the algorithm? Conclude as to the total complexity.

2.2) What improvements could be proposed to reduce this complexity?

3) Proof of the algorithm:

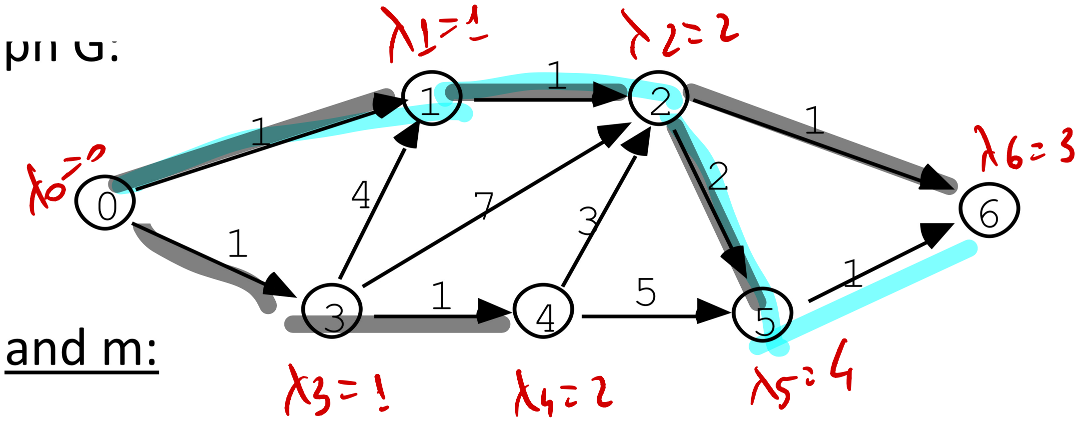
We note **first** γ = (0, n-1) = (y0 = 0, y1, ..., yp = n-1) the shortest path obtained in the phase 1) of the algorithm and **second** = (z0 = 0, z1, ..., zq = n-1) a second shortest path from 0 to n-1.

3.1) Show that there is an integer r such that yq = zp, yp-1 = 1 ,..., yp-r = zq-r and yp-r-1 ≠ zq-r-1.

3.2) What is the remarkable property of the path (0 = z0, z1, ..., zq-r-1) ?

3.3) Deduce the validity of the algorithm.

1)



2) $\gamma[0, n-1] = [0, 1, 2, 6]$

3) value $= \emptyset$, $y_1 = 1$; $k = 3$

$\alpha = \lambda_3 + v_{31} + \lambda_6 - \lambda_1 = 7 \Rightarrow 7 < \infty$

$pivot_1 = 3$, $pivot_2 = 1$

$y_2 = 2$: $k \in \{3, 4\}$

$k = 3$: $\alpha = \lambda_3 + v_{32} + \lambda_6 - \lambda_2 = 9$

$k = 4$: $\alpha = \lambda_4 + v_{42} + \lambda_6 - \lambda_2 = 6$

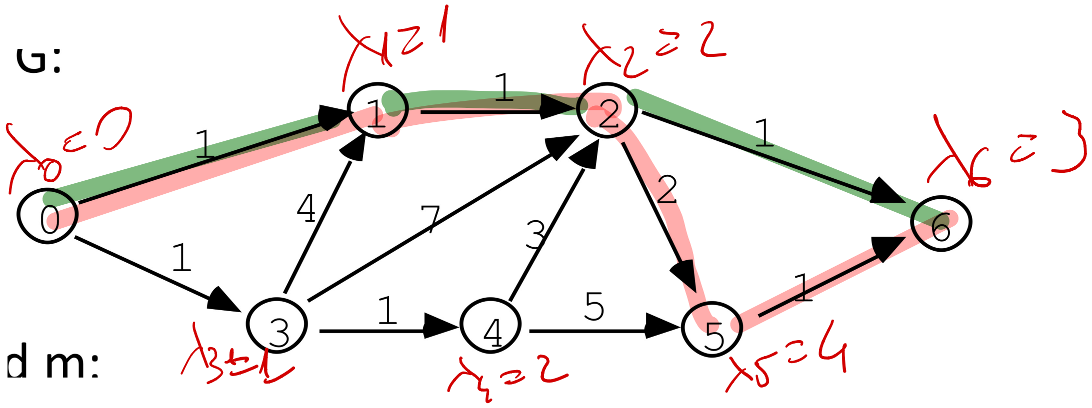
$pivot_3 = 4$ $pivot_2 = 2$.

$y_3 = 6$;

$k = 5$: $\alpha = \lambda_5 + \sqrt{56} + \lambda_6 - \lambda_6$

$= \lambda_5 + \sqrt{56} = 5$

$pivot_1 = 5$, $pivot_2 = 6$.



$y_0=0,\quad y_1=1 \quad y_2=2 \quad y_3=6$

$j=1 \qquad K=\{3\}. \quad \alpha=\lambda_3+v_{31}+\lambda_6-\lambda_1=$

$=1+4+3-1=7$; value $=7$, pivot$_1=3$

pivot$_2=1$

$j=2 \qquad K\in\{3,4\}.$ $K=3$: $\alpha=\lambda_3+v_{32}+\lambda_6-\lambda_2=$

$=1+7+3-2=9$

$K=4$: $\alpha=\lambda_4+v_{42}+\lambda_6-\lambda_2=2+3+3-2=6$

$\Rightarrow$ Value $=6$, pivot$_1=4$ pivot$_2=2$

$j=3$: $K=\{5\}$ $K=5$ $\alpha=\lambda_5+v_{56}+\lambda_6-\lambda_6=$

$=4+1+3-3=5$

$\Rightarrow$ Value $=5$, pivot$_1=5$, pivot$_2=6$,

Bellman algorithm

Algorithm:

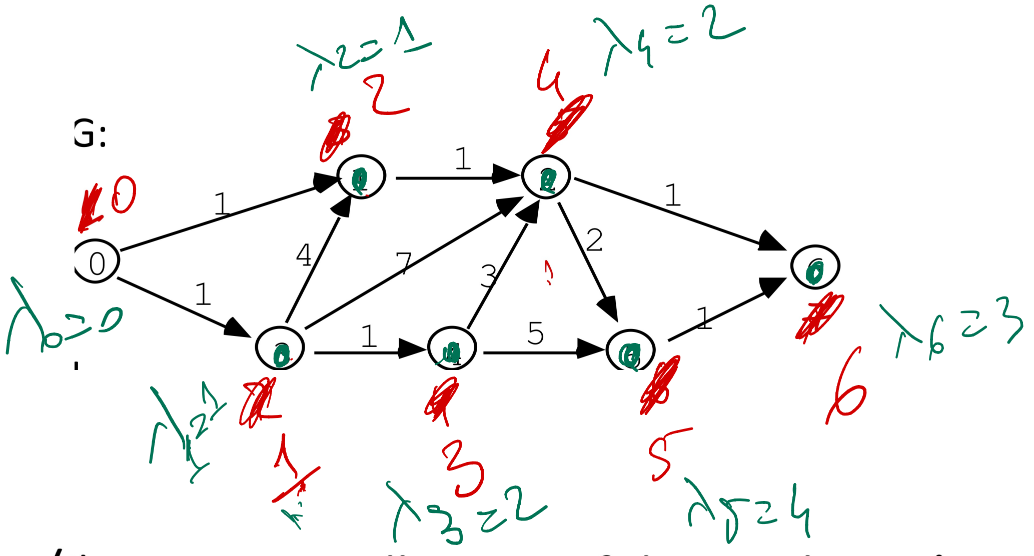
(i) **enumerate** all vertex of the graph, set $\lambda_0 = \cancel{0}$. 1

(ii) for j = 1 to n – 1 set : $\lambda_j = \cancel{\min}$ max $(\lambda_k * v_{kj})$ over the set of predecessors x_k of x_j.

Ps. Vertex numeration is a function f : {V}->N s.t. for any arc (x_i, x_j) $f(x_j) < f(x_j)$.

Proof by recurrence…

Theorem 5: Bellman algorithm computes the shortest path values λ_i from x_0 in O(m).



(i) **enumerate** all vertex of the graph, set $\lambda_0 = 0$.

(ii) for j = 1 to n – 1 set : $\lambda_j = \min(\lambda_k + v_{kj})$ over the set of predecessors x_k of x_j.

$\lambda_0 = 0.$

$\lambda_1 = 0 + 1 = 1$

$\lambda_2 = \min\{0+1, 1+4\} = 1$

$\lambda_3 = 2$

$\lambda_4 = \min\{\overset{2}{\overbrace{\lambda_2+1}}, \overset{8}{\overbrace{\lambda_1+7}}, \overset{5}{\overbrace{\lambda_3+3}}\} = 2$

$\lambda_5 = \min\{\overset{7}{\overbrace{\lambda_3+5}}, \overset{4}{\overbrace{\lambda_4+2}}\} = 4$

$\lambda_6 = \min\{\overset{3}{\overbrace{\lambda_4+1}}, \overset{5}{\overbrace{\lambda_5+1}}\} = 3.$

Some path problems

- The longest path computation problem;
- The maximum probability path;
- The maximum capacity path value;

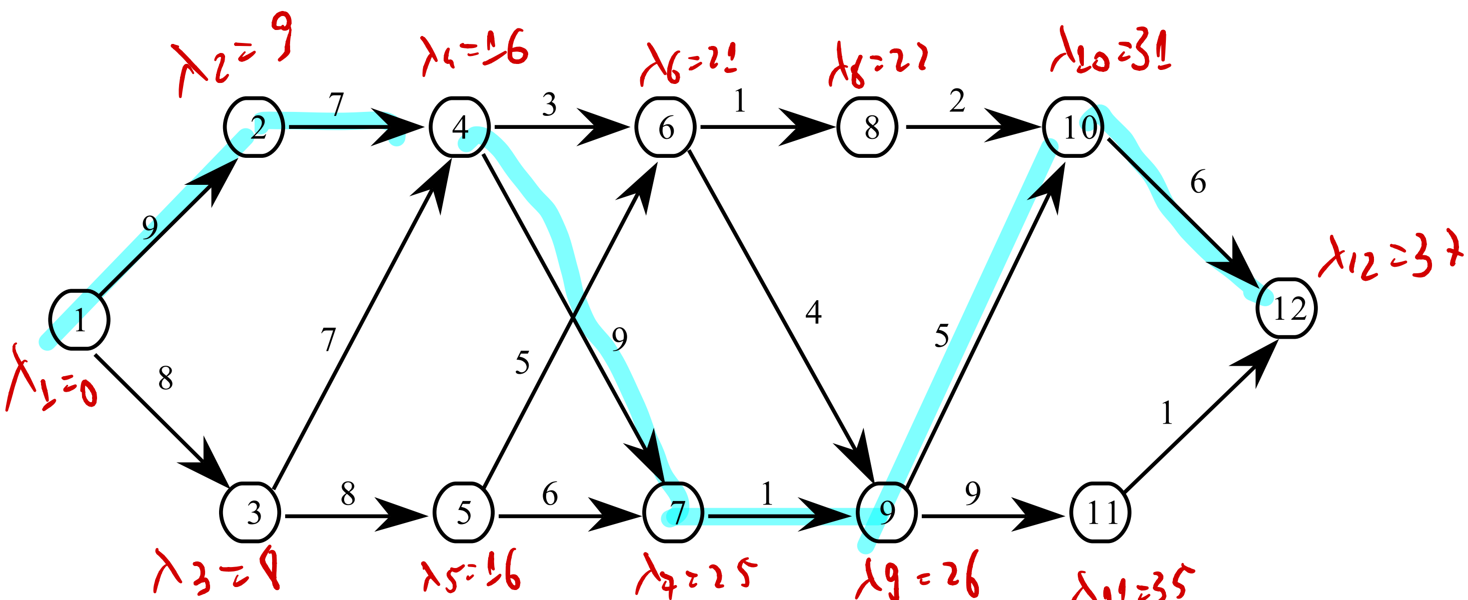
  - Exercise : compute the shortest path among these of maximum capacity.

Exercise: The itinerary of Michel Strogoff

Leaving from Moscow, Michel STROGOFF, courier of the tsar, was supposed to reach IRKUTSK. Before leaving, he had consulted a fortune teller who told him, amongst other things : "After KAZAN beware of the sky, in OMSK beware of the tartars, in TOMSK beware of the eyes, after TOMSK beware of water and, above all, always be careful of a large brown-haired person with black boots. " STROGOFF had therefore written on a map his "chances" of success for each route between two towns : these chances were represented by a number between 1 and 10 (measuring the number of chances of success out of 10). Ignoring probability calculation, he had therefore chosen his route by maximising the total sum of the chances.

The numbers of the cities are: MOSCOW (1), KAZAN (2), PENZA(3), PERM (4), OUFA (5), TOBOLSK (6), NOVO-SAIMSK (7), TARA (8), OMSK (9), TOMSK (10), SEMIPALATINSK(11), IRKOUTSK (12).

Exercise: The itinerary of Michel Strogoff



1. Determine the route of Michel Strogoff.
   $\mu = [1,2,4,7,9,10,12]$; $P(\mu) = \prod_{(i,j)\in\mu} P_{i,j} = 1,7\%$
2. What was the probability, with the assumption of the independence of the random variables, that Strogoff would succeed?
3. What would have been his route if he had known the principles of probability calculation?

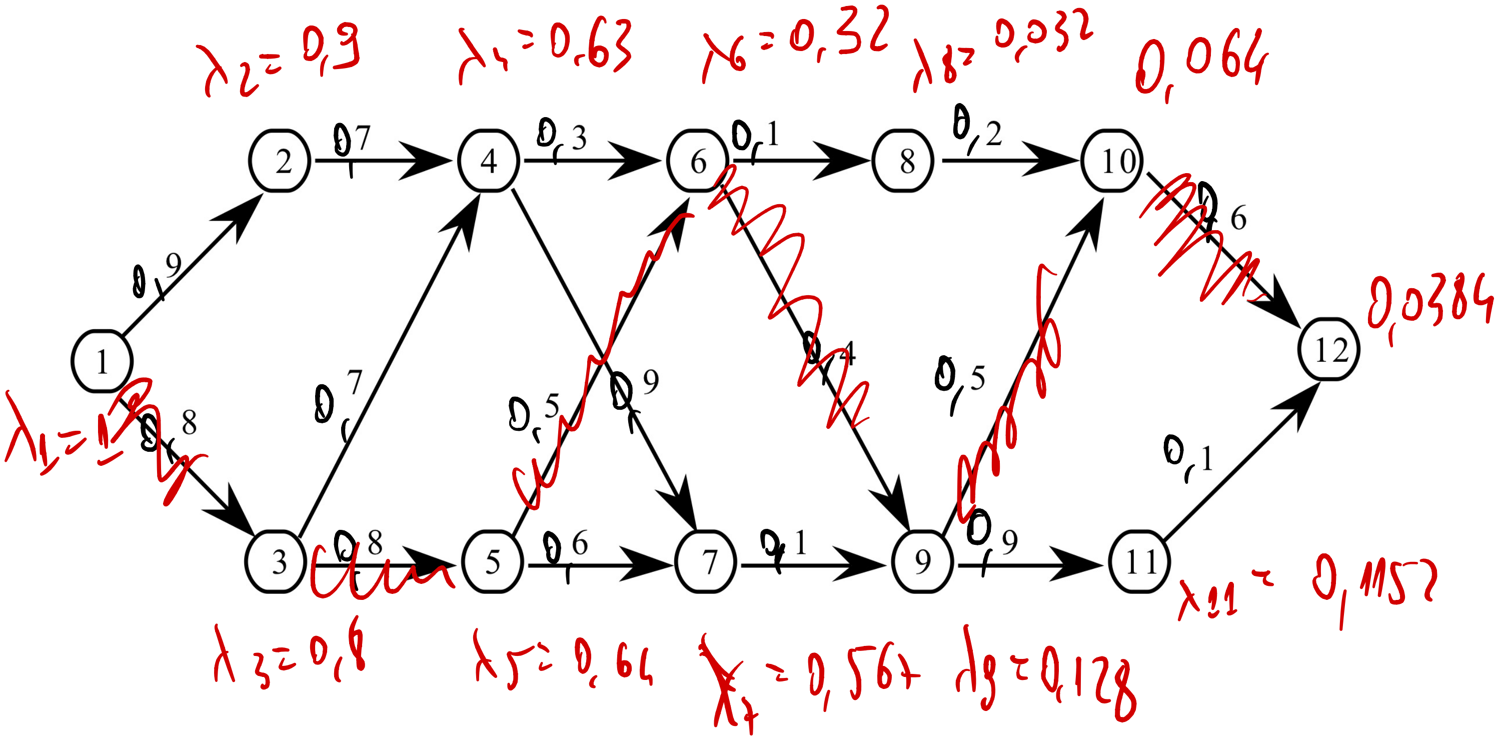
Matrix method (I)

```
for i←1 à n do {
    for j←1 à n do {
        if (j∈U⁺(i)) then V⁰[i][j] ← v_ij otherwise V⁰[i][j] ← ∞;
    }
}
for k←1 à n do {          O(n³)
    for i←1 à n do {
        for j←1 à n do {
            Vᵏ[i][j] ← min(Vᵏ⁻¹[i][j] , Vᵏ⁻¹[i][k]+Vᵏ⁻¹[k][j])
        }
    }
}
```

Proof of validity of the algorithm by recurrence :

Hint : at the end of iteration **k**, **V^k[j][j]** gives the value of the shortest path from **i** to **j** going through vertices **{1, 2, …, k}+{i, j}.**

Complexity : $O(n^3)$



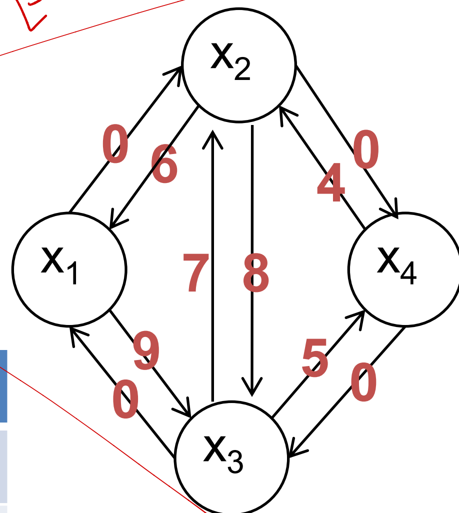
λ2 = 0,9
λ4 = 0.63
λ6 = 0,32
λ8 = 0,032
0,064
0,0384
λ1 = 1
λ3 = 0,8
λ5 = 0,64
λ7 = 0,567
λ9 = 0,128
λ11 = 0,1152

Matrix method (II)

$\min\{V^1_{[1,3]}, V^1_{[1,2]} + V^1_{[2,3]}\}$

$V^3_{[1,2]} = \min\{V^2_{[1,2]}, V^2_{[1,3]} + V^2_{[3,2]}\}$

4 0




```
for i←1 à n do {
      for j←1 à n do {
            if (j∈U⁺(i)) then V⁰[i][j] ← vij
      otherwise V⁰[i][j] ← ∞;
      }
}
for k←1 à n do {
      for i←1 à n do {
            for j←1 à n do {
                        Vᵏ[i][j] ← min(Vᵏ⁻¹[i][j] ,
      Vᵏ⁻¹[i][k]+Vᵏ⁻¹[k][j])
            }
      }
}
```

V^0

	1	2	3	4
1	0	0	9	∞
2	6	0	8	0
3	0	7	0	5
4	∞	4	0	0

V^1

	1	2	3	4
1	0	0	9	∞
2	6	0	8	0
3	0	0	0	5
4	∞	4	0	0

V^2

	1	2	3	4
1	0	0	8	0
2	6	0	8	0
3	0	0	0	0
4	10	4	0	0

V^3

	1	2	3	4
1	0	0	8	0
2	6	0	8	0
3	0	0	0	0
4	0	0	0	0

V^4

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

END.