RO03
Problèmes de cheminement

Partie 2

Recherche des chemins de valeur minimale de x_0 vers tout sommet x_i, $i \in \{0, \dots, n-1\}$

prop 1 : tout sous-chemin d'un chemin de valeur minimale est de valeur minimale.

prop 2 : il est nécessaire et suffisant qu'il n'existe pas de circuit de valeur négative dans le graphe pour qu'il existe un chemin de valeur minimale de x_0 à tout autre sommet x_i, $i \in \{0, \dots, n-1\}$.

prop 3 : si $\{\lambda_i \mid i \in \{0, \dots, n-1\}\}$ est un ensemble de valeurs de chemins, avec λ_i la valeur d'un chemin de x_0 à x_i, alors il est nécessaire et suffisant que $\lambda_0 = 0$ et $\forall (x_i, x_j) \in U$ $\lambda_j \leq \lambda_i + v_{ij}$ pour que $\lambda_i = \lambda_i^*$, $\forall i \in X$, i.e. λ_i est la valeur d'un chemin de valeur minimale allant de x_0 à x_i.

Algorithmes de cheminements

Chercher un chemin de valuation minimale de x_0 à tout sommet x_i :

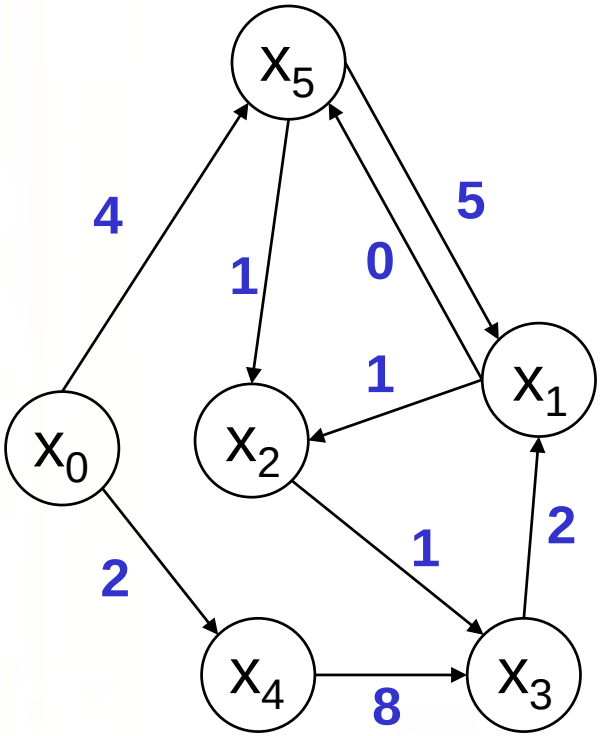
- Algorithme de FORD :
  - Valide avec des valuations quelconque
  - O(n m)
  - Algorithme à correction d'étiquettes

- Algorithme de DIJKSTRA :
  - Valide uniquement avec des valuations positives ou nulles
  - $O(n^2)$
  - Algorithme à fixation d'étiquette

- Algorithme de BELLMAN :
  - Valide avec des valuations quelconque mais uniquement s'il n'existe pas de circuit
  - O(m)
  - Algorithme à fixation d'étiquettes

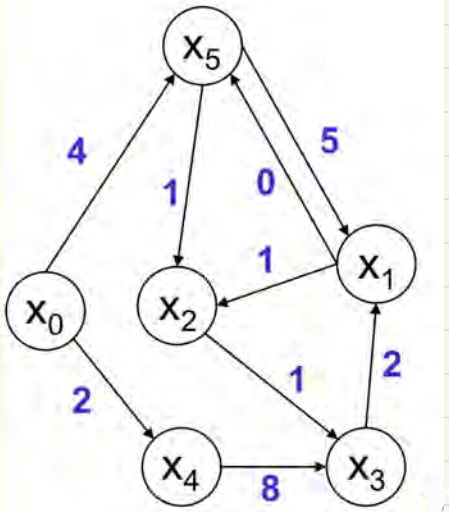
Algorithme de DIJKSTRA

Attention : valide uniquement si les valuations sont ≥ 0




```
pour i←0 à n-1 faire {
    λ[i] ← ∞; visité[i] ← faux; P[i] ← -1;
}
λ[0] ←0; P[0] ←0; nb_visités←0;
tant que (nb_visités < n){
    min ← -1;
    pour i←0 à n-1 faire {
        si (visité[i]=faux et (min=-1 ou λ[i] < λ[min] )
        alors min ← i;
    }→ Sélection du sommet (min) de plus petit λ parmi ceux non encore visités
    visité[min] ←vrai; nb_visités← nb_visités +1;
    pour j ∈ U+(min) faire {
        si (visité[j]=faux et λ[j]> λ[min]+v_min,j) alors {
            λ[j] ← λ[min]+v_min,j;  P[j] ←min;
        } → ajustement des λ des successeurs si meilleur chemin trouvé à partir de min.
    }
}
```

Par rapport à l'algorithme repaté dans le poly :
visité[i]= vrai <⇒ xi ∈ S



i	λ[i]	P[i]	visité[i]
0	~~∞~~ 0	~~x~~ 0	~~faux~~ vrai
1	~~∞~~ ~~8~~ 8	~~x~~ ~~8~~ 3	~~faux~~ vrai
2	~~∞~~ 5	~~x~~ 5	~~faux~~ vrai
3	~~∞~~ ~~6~~ 6	~~x~~ ~~2~~ 2	~~faux~~ vrai
4	~~∞~~ 2	~~x~~ 0	~~faux~~ vrai
5	~~∞~~ 4	~~x~~ 0	~~faux~~ vrai

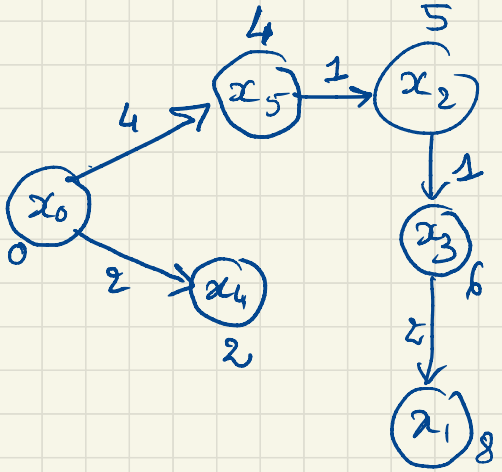
```
pour i←0 à n-1 faire {
    λ[i] ← ∞; visité[i] ← faux; P[i] ← -1;
}
λ[0] ←0; P[0] ←0; nb_visités←0;
tant que (nb_visités < n){
    min ← -1;
    pour i←0 à n-1 faire {
        si (visité[i]=faux et (min=-1 ou λ[i] < λ[min] )
        alors min ← i;
    }
    visité[min] ←vrai;  nb_visités← nb_visités +1;
    pour j ∈ U+(min) faire {
        si (visité[j]=faux et λ[j]> λ[min]+v_min,j) alors {
            λ[j] ← λ[min]+v_min,j;  P[j] ←min;
        }
    }
}
```

itération 3 min = 5

itération 4 min = 2

itération 5 min = 3

itération 6 min = 2



Algorithme de DIJKSTRA

- valuations ≥ 0 : il n'existe pas de circuit absorbant

- Démonstration de la validité de l'algorithme par récurrence :

Invariant:

A la fin de l'itération **k**,

1- Si $x_i \in S$ (visité[i]=vrai) : $\lambda[i] = \lambda^*_i$.

valeur optimale

2- Si $x_i \notin S$ (visité[i]=faux) : $\lambda[i] = \min_{z \in U^-(i) \cap S} \lambda[z] + v_{zi}$

ensemble des prédécesseurs de i qui sont déjà visités

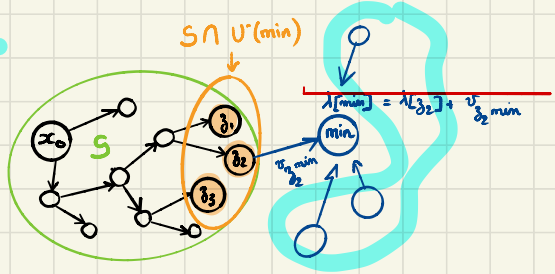
- **Complexité** : $O(n^2)$

Invariant : à la fin de l'itération k, si visité$[i]$ = vrai alors $\lambda[i] = \lambda_i^*$

si visité$[i]$ = faux alors $\lambda[i] = \min_{z \in U^-(i) \cap S} \lambda[z] + v_{zi}$

L'invariant est vrai initialement : il n'y a aucun sommet visité et supposons l'invariant vrai à la fin de la k^e itération.

A la $(k+1)^e$ itération, soit min un sommet non visité de plus petit λ : $\lambda[min] = \min_{z \notin S} \lambda[z]$.

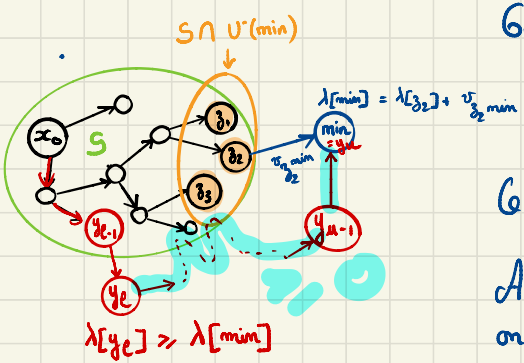


Par hypothèse, $\boxed{\lambda[min] = \min_{z \in U^-(min) \cap S} \lambda[z] + v_{z\,min}}$

« $\lambda[min]$ est donc la valeur d'un chemin de valeur minimale qui n'utilise que des sommets $\in S + \{x_i\}$ »

Supposons que $\lambda[min] > \lambda^*_{min}$: alors il existe un chemin $\mu = [y_0 = x_0, \dots, y_u = x_{min}]$ avec $v(\mu) = \lambda^*_{min}$

tel qu'au moins un sommet y_ℓ de ce chemin $\notin S$: $\exists\, \ell \in \{1, \dots, u\}$ tel que $y_{\ell-1} \in S$ et $y_\ell \notin S$.



Or, $\lambda^*_{min} = v(\mu) = v([y_0, \dots, y_\ell]) + v([y_\ell, \dots, y_u = x_{min}]) \geq \lambda[min]$

$\lambda[y_\ell] \geq \lambda[min]$

≥ 0 car les valuations sont positives

Ce qui contredit le fait que $\lambda[min] > \lambda^*_{min}$.

Après avoir exécuté $\lambda[min] \leftarrow$ vrai et ajusté les $\lambda[j]$, $j \in U^+(min) / S$, on a $\forall i \notin S$ $\lambda[i] = \min_{z \in U^-(i) \cap S} \lambda[z] + v_{zi}$

Algorithme de DIJKSTRA

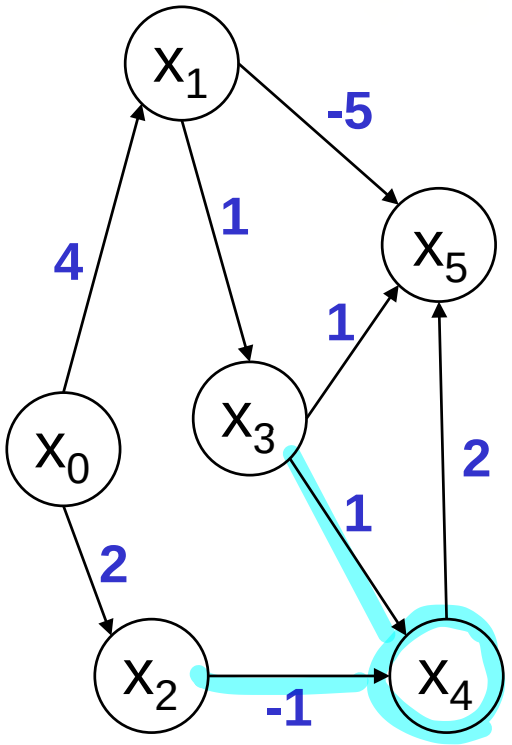
```
pour i←0 à n-1 faire {
    λ[i] ← ∞; visité[i] ← faux; P[i] ← -1;
}
λ[0] ←0; P[0] ←0; nb_visités←0;
tant que (nb_visités < n){
    min ← -1;
    pour i←0 à n-1 faire {
        si (visité[i]=faux et (min=-1 ou λ[i] < λ[min] )
        alors min ← i;
    }
    visité[min] ←vrai;  nb_visités← nb_visités +1;
    pour j ∈ U+(min) faire {
        si (visité[j]=faux et λ[j]> λ[min]+v_min,j) alors {
            λ[j] ← λ[min]+v_min,j;  P[j] ←min;
        }
    }
}
```

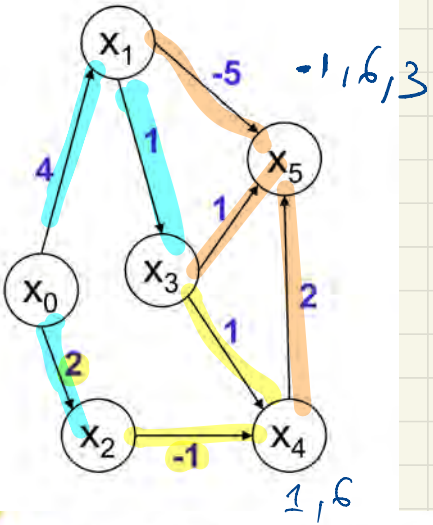
COMPLEXITÉ ?

Algorithme de BELLMAN

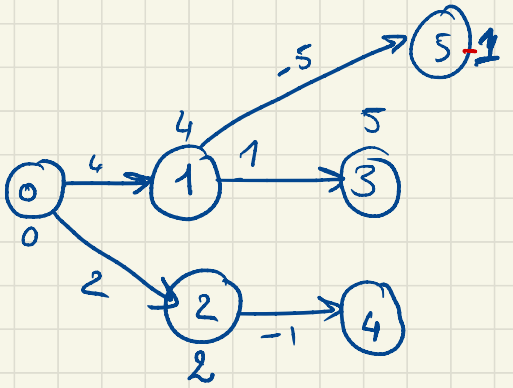
Attention : valide uniquement si le graphe est sans circuit

Numéroter les sommets du graphe;
$\lambda[0] \leftarrow 0$; $P[0] \leftarrow 0$;
pour $j \leftarrow 1$ à $n-1$ **faire {**
    $\lambda[j] \leftarrow \infty$;
    pour chaque prédécesseur i de j **faire {**
        si ($\lambda[j] > \lambda[i]+v_{ij}$) **alors {**
            $\lambda[j] \leftarrow \lambda[i]+v_{i,j}$;
            $P[j] \leftarrow i$;
        }
    }
}





i	λ[i]		P[i]
0	0		0
1	~~∞~~	4	0
2	~~∞~~	2	0
3	~~∞~~	5	1
4	~~∞~~	1	2
5	~~∞~~	-1	1



Numéroter les sommets du graphe;
λ[0] ←0; P[0] ←0;
pour j←1 à n-1 **faire {**
    λ[j] ←∞;
    pour chaque prédécesseur i de j **faire {**
        si (λ[j]> λ[i]+v_{ij}) **alors {**
            λ[j] ← λ[i]+$v_{i,j}$;
            P[j] ←i;
        }
    }
}

Algorithme de Bellman

- le graphe est sans circuit donc sans circuit absorbant

- Démonstration de l'algorithme par récurrence :

Invariant:

A la fin de l'itération **j**, $\lambda[j] = \lambda^*_j$.

- **Complexité** : O(m)

Invariant : À la fin de la j^e itération $\lambda[j] = \lambda^*_j$.

L'invariant est vrai initialement : $\lambda[0] = 0$. Supposons l'invariant vrai à la fin de la $(j-1)^e$ itération.

À l'itération j, tous les prédécesseurs $x_i \in U^-(x_j)$ sont tels que $i < j$.

Donc par hypothèse, $\forall x_i \in U^-(x_j)$, $\lambda[i] = \lambda^*_i$.

Comme le chemin de valeur minimale de x_0 à x_j n'est composé que de sous-chemins de valeur minimale, le chemin μ de x_0 à x_j de valeur minimale est composé d'un chemin de valeur minimale de x_0 à un des sommets $x_i \in U^-(x_j)$ et de l'arc (x_i, x_j).

Ainsi $\lambda^*_j = \min_{x_i \in U^-(x_j)} \lambda_i + v_{ij}$, ce que calcule l'algorithme à l'itération j.

Algorithme de BELLMAN

COMPLEXITÉ ???

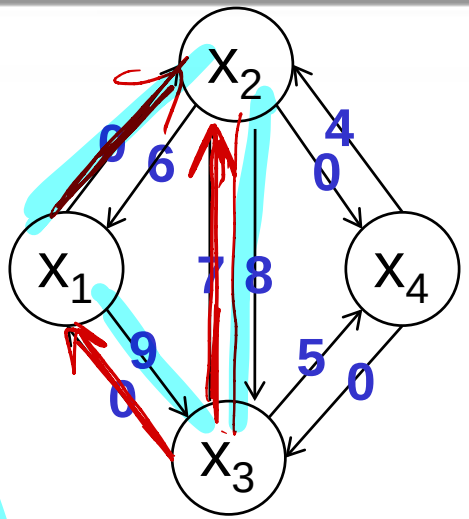
Attention : valide uniquement si le graphe est sans circuit

Numéroter les sommets du graphe;
$\lambda[0] \leftarrow 0$; $P[0] \leftarrow 0$;
pour j←1 à n-1 **faire {**
    $\lambda[j] \leftarrow \infty$;
    pour chaque prédécesseur i de j **faire {**
        si ($\lambda[j] > \lambda[i]+v_{ij}$) **alors {**
            $\lambda[j] \leftarrow \lambda[i]+v_{i,j}$;
            $P[j] \leftarrow i$;
        }
    }
}

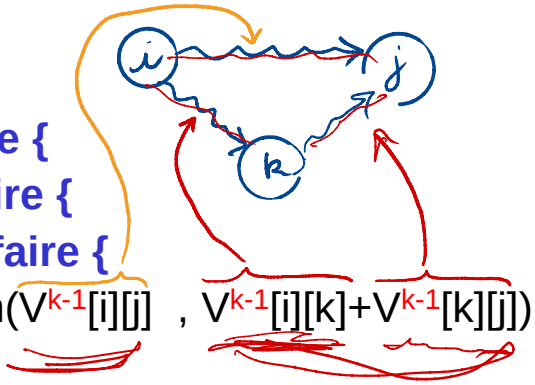
Méthode matricielle

Calculer les valeurs des chemins de valeur minimale entre toute paire de sommets.

```
pour i←1 à n faire {
    pour j←1 à n faire {
        si (j∈U+(i)) alors V0[i][j] ← vij sinon V0[i][j] ← ∞;
    }
}
pour k←1 à n faire {
    pour i←1 à n faire {
        pour j←1 à n faire {
            Vk[i][j] ← min(Vk-1[i][j] , Vk-1[i][k]+Vk-1[k][j])
        }
    }
}
```




```
pour i←1 à n faire {
    pour j←1 à n faire {
        si (j∈U+(i)) alors V0[i][j] ← vij
        sinon V0[i][j] ← ∞;
    }
}
pour k←1 à n faire {
  pour i←1 à n faire {
    pour j←1 à n faire {
      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

Méthode matricielle

- L'absence de circuit absorbant permet de se limiter à la recherche de chemins élémentaires de valeur minimale.

- Démonstration de l'algorithme par récurrence :

Invariant:

A la fin de l'itération **k**, **$V^k[j][j]$** est la valeur minimale d'un chemin élémentaire allant de **i** à **j** et ne passant que par les sommets **{1, 2, …, k}+{i, j}.**

- **Complexité** : $O(n^3)$

Invariant:

À la fin de l'itération **k**, $V^k[j][j]$ est la valeur minimale d'un chemin élémentaire allant de **i** à **j** et ne passant que par les sommets **{1, 2, …, k}+{i, j}**.

Invariant vrai initialement : $V^0[i][j] \neq \infty$ seulement si $(x_i, x_j) \in U$

On suppose l'invariant vrai à la fin de la $(k-1)^e$ itération.

À la k^e itération, on considère un chemin μ de valeur minimale de x_i à x_j qui n'emprunte que des sommets dans $\{x_0, \dots, x_k\} \cup \{x_i, x_j\}$ et soit λ_{ij}^{k*} sa valeur.

- Soit μ ne passe pas par x_k, il n'emprunte alors que des sommets $\in \{x_0, \dots, x_{k-1}\} \cup \{x_i, x_j\}$

  On a donc $\lambda_{ij}^{k*} = \lambda_{ij}^{k-1*} = V^{k-1}[i][j]$

- Soit μ passe par x_k : comme tout sous-chemin d'un chemin de valeur minimale est de valeur minimale, alors μ est constitué :

. d'un sous-chemin de x_i à x_k n'empruntant que des sommets dans $\{x_0, \dots, x_{k-1}\} \cup \{x_i, x_k\}$ de valeur $\lambda_{ik}^{k-1*} = V^{k-1}[i][k]$

. d'un sous-chemin de x_k à x_j n'empruntant que des sommets dans $\{x_0, \dots, x_{k-1}\} \cup \{x_j, x_k\}$ de valeur $\lambda_{kj}^{k-1*} = V^{k-1}[k][j]$

avec $\lambda_{ij}^{*k} = \lambda_{ik}^{k-1*} + \lambda_{kj}^{k-1*} = V^{k-1}[i][k] + V^{k-1}[k][j]$ ce que calcule l'algorithme…

Autres problèmes de cheminement

- Chemins de valeur maximale: FORD et BELLMAN sont valides en remplaçant **min** par **max**, en initialisant les λ_i à $-\infty$ et λ_0 à 0.
- Existence d'un chemin entre i et j : méthode matricielle ou puissance d'une matrice d'adjacence (CF TD)
- Dénombrement du nombre de chemins entre i et j.
- Chemin de capacité maximale qui est la plus petite des valuations du chemin. FORD et BELLMAN sont utilisables en remplaçant **+** par **max** et **min** par **max** et en initialisant les λ_i à -1 et λ_0 à ∞.

Chemins de fiabilité maximale

- Dans une région en proie à un conflit ou une catastrophe naturelle, un convoi doit être acheminé d'une ville s à une ville t.
- Le réseau routier est donné par un graphe valué:
  - les sommets : l'ensemble des villes
  - Les arcs : les tronçons de route
  - La valuation d'un arc (i,j) est la probabilité de parcourir sans dommage la route de i à j.
- On souhaite déterminer un itinéraire maximisant la probabilité pour le convoi d'arriver sans dommage en t.
- La valeur d'un chemin = le produit des valuations de ses arcs : adaptation des algorithmes ou prendre le log des valuations.

Le problème du caboteur

- Une compagnie maritime veut définir une ligne côtière desservant cycliquement N ports.
- On connaît pour chaque voyage d'un port x à un port y:
  - le bénéfice b(x,y);
  - la durée du trajet d(x,y).
- La compagnie souhaite maximiser le bénéfice par unité de temps, égal pour un parcours P au bénéfice cumulé en suivant P, divisé par la durée de P.
- Le problème consiste donc à trouver un cycle P maximisant ce rapport.
- La valeur du chemin est la valeur moyenne ou moyenne pondérée des chemins.