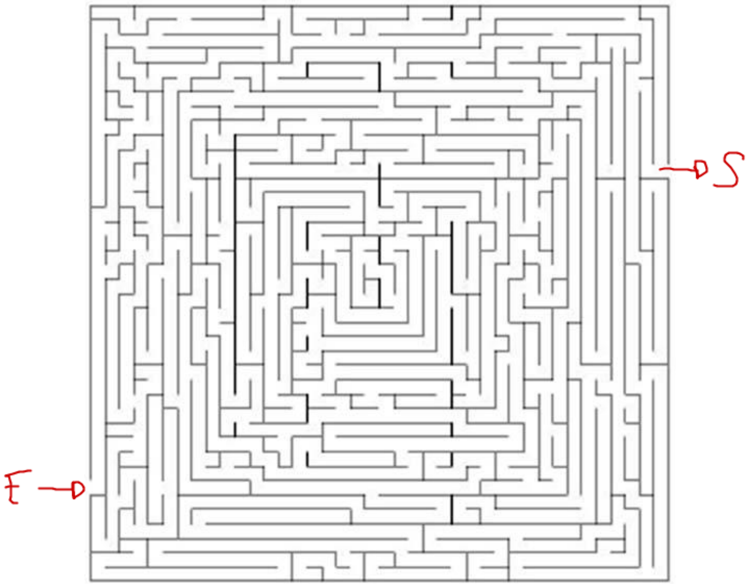
Des algorithmes dans les graphes

stage IREM - Nov./Déc. 2010

Plan

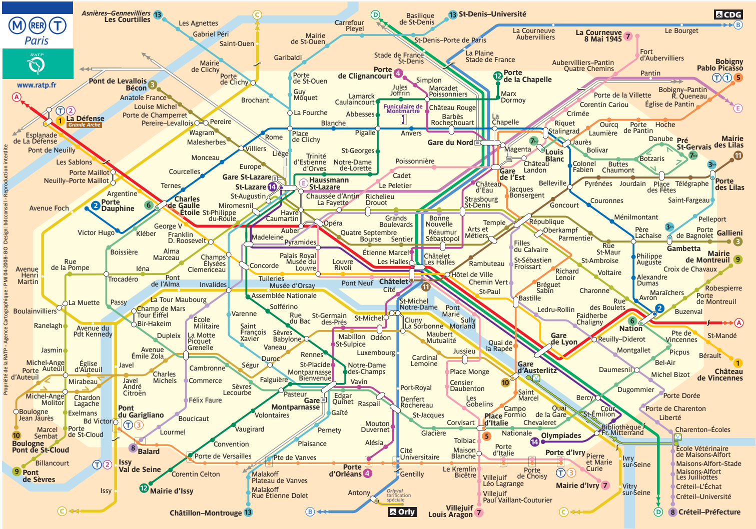
Plan

Comment sortir ?



Trouver un chemin optimal

Comment aller de Chevaleret à Porte de Bagnolet ?



Trouver un chemin optimal

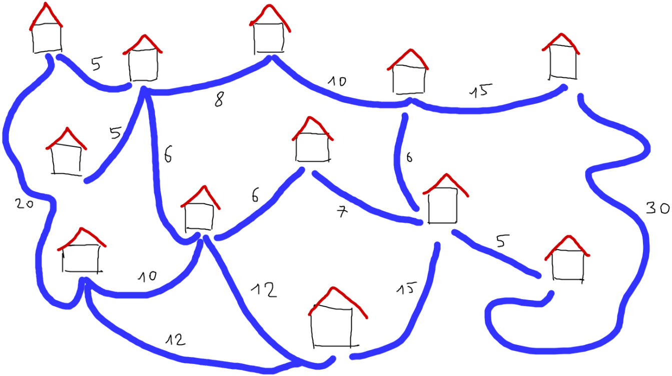
Comment aller de Paris à Morlaix ?

- en minimisant la distance ?
- en minimisant la durée ?
- en minimisant le coût ?
- ...

Calculs de plus courts chemins dans un graphe !

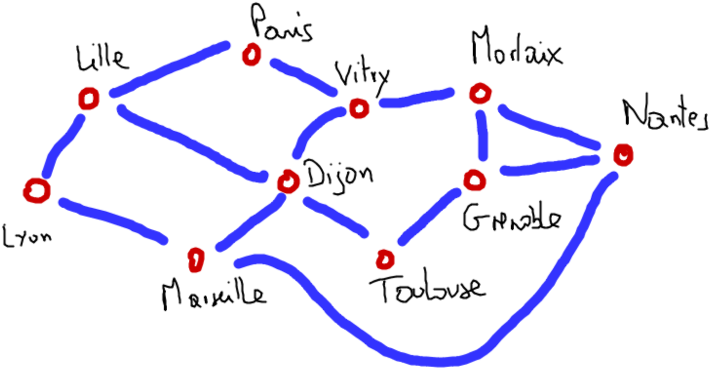
Le problème de la ville embourbée

Problème : Paver suffisamment de rues pour que tous les habitants puissent se rendre n'importe où les pieds au sec. . . mais en utilisant le moins de pavés possible.



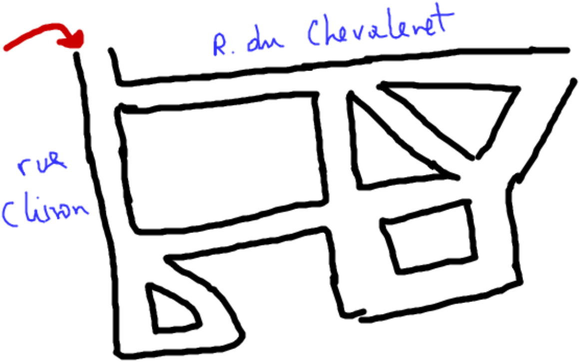
Le problème des bandits

Problème : Des bandits veulent visiter toutes les villes d'une région sans jamais repasser par une ville.



Le problème de la goudronneuse

Problème : On veut faire passer une goudronneuse une et une seule fois par chaque rue.
(on s'autorise à repasser aux croisements. . .)



Plan

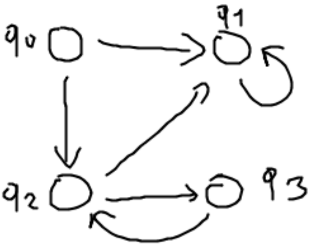
Plan

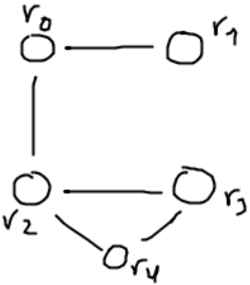
Les graphes

$G = (S, A)$:

- S est un ensemble fini de sommets.
- A est un ensemble fini d'arêtes (ou arcs, ou transitions. . .).

On va considérer des graphes orientés ou non-orientés, et parfois valués.



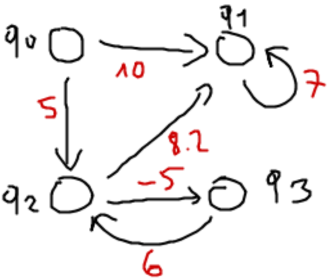


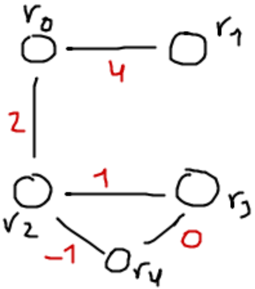
Les graphes

$G = (S, A)$:

- S est un ensemble fini de sommets.
- A est un ensemble fini d'arêtes (ou arcs, ou transitions. . .).

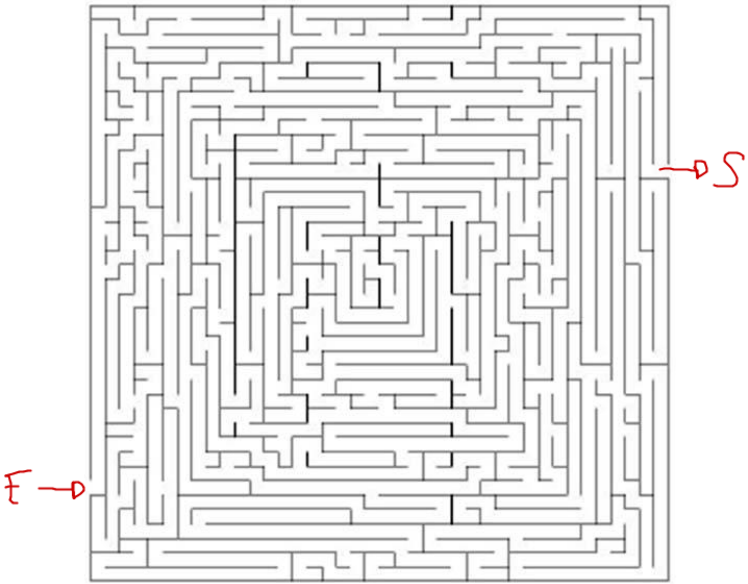
On va considérer des graphes orientés ou non-orientés, et parfois valués.





Plan

Comment sortir ?



Quel algorithme ?

Quel est le problème ?

Quel algorithme ?

Quel est le problème ?

Longer le mur sur la gauche ?

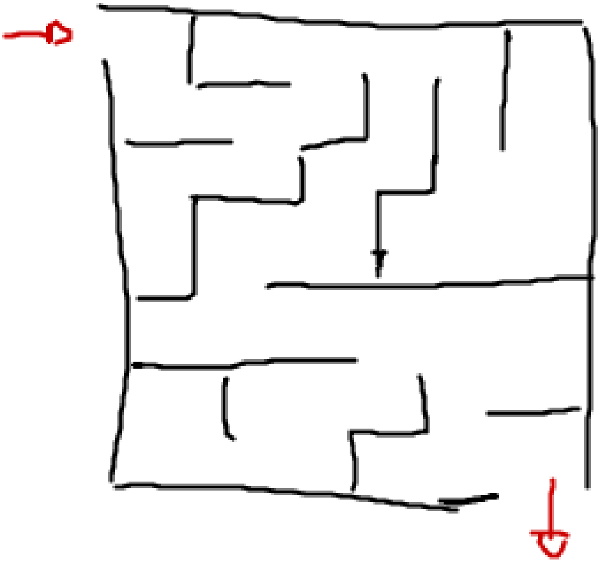
Quel algorithme ?

Quel est le problème ?

Longer le mur sur la gauche ?

Utiliser un fil d'Ariane ?

Exemple



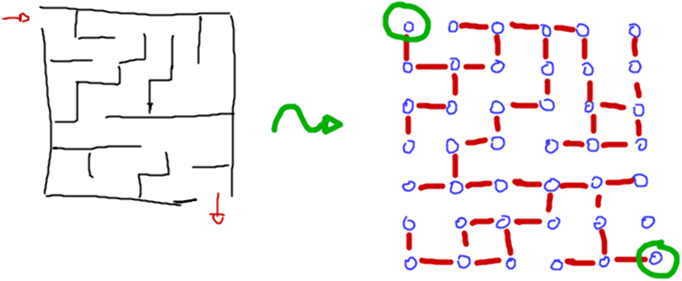
Parcourir sans boucler

Commencer par la première case du labyrinthe. . .

1. Si la case courante est la sortie, alors c'est fini !
2. (Sinon :) marquer la case
3. pour chaque case C atteignable en "un coup" :

   Si la case C n'est pas marquée, l'explorer en recommençant au point **1**

Du labyrinthe au graphe



Parcours depuis une origine

Initialisation : Couleur$[r] \leftarrow$ blanc et $\Pi[r] \leftarrow$ nil $\forall r \in S$.
Puis appel de PP-Visiter(G, q_{init}) :

PP-Visiter(G, q)

```
begin
    Couleur[q] ← noir;
    pour chaque (q, q') ∈ A faire
        si Couleur[q'] = blanc alors
            Π[q'] ← q;
            PP-Visiter(G, q');
end
```

(variante simplifiée du parcours en profondeur)

Recherche d'un chemin de q_{init} à q_F

Initialisation : Couleur$[r] \leftarrow$ blanc et $\Pi[r] \leftarrow$ nil $\forall r \in S$.
Puis appel de Chemin(G, q_{init}, q_S) :

Chemin(G, q, q_S)

---

begin
- **si** $q = q_S$ **alors retourner** Vrai;
- Couleur$[q] \leftarrow$ noir;
- **pour chaque** $(q, q') \in A$ **faire**
  - **si** $Couleur[q'] = blanc$ **alors**
    - $\Pi[q'] \leftarrow q$;
    - `trouve` $\leftarrow$ `Chemin`(G, u);
    - **si** $\mathtt{trouve} = Vrai$ **alors**
      - **retourner** Vrai
- **retourner** Faux

end

Parcours en profondeur

Algorithme important !
A la base des algorithmes de recherche des composantes fortement connexes (par ex. algorithme de Tarjan) et de tri topologique.

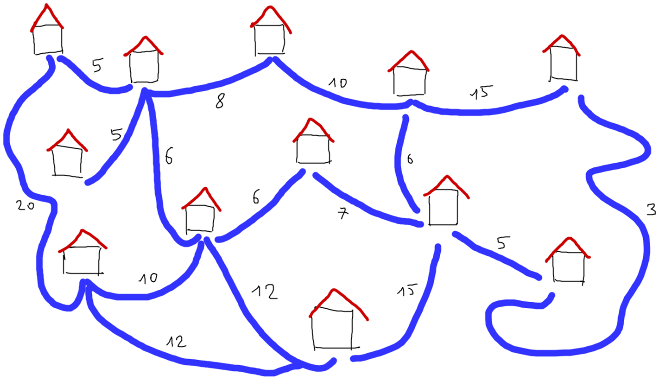
Algorithme de type “backtracking”.

Un autre parcours classique est le parcours en largeur.

Plan

Le problème de la ville embourbée

Problème : Paver suffisamment de rues pour que tous les habitants puissent se rendre n'importe où les pieds au sec. . . mais en utilisant le moins de pavés possible.



NB : graphe non-orienté, valué et connexe.

Algorithme 1

1. $S = \emptyset$
2. On trie les arêtes par poids croissant.
3. Pour chaque arête (x, y) :

   Si (x, y) ne crée pas de cycle,

   alors on l'ajoute à S

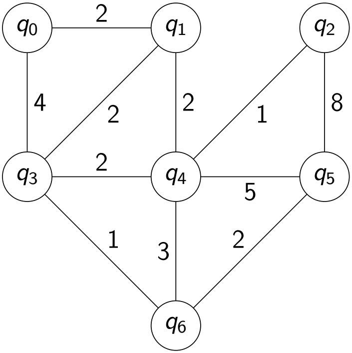
S est une solution !

Algorithme 2

1. $S = \emptyset$
2. On choisit un sommet de départ s.
3. On ajoute à S une arête (s, x) de poids minimal,
4. On ajoute à S une arête (q, q') reliant un sommet isolé à l'arbre en construction et de poids minimal,
5. recommencer le point **4** jusqu'à la connection de tous les sommets.

S est une solution !

Exemple



Arbres couvrants minimaux

Plus formellement...

$G = (S, A, w)$: non-orienté, connexe et valué ($w : A \to \mathbb{R}$).

Définitions :

- un arbre couvrant de G est un graphe $T = (S, A')$ avec $A' \subseteq A$, connexe et acyclique.
- un arbre couvrant $T = (S, A')$ est dit minimal lorsque :

$$w(A') \;=\; \min\{w(A'') \mid T' = (G, A'')\,\text{est un AC de}\; G\}$$

avec $w(A) \stackrel{\text{def}}{=} \sum_{(x,y)\in A} w(x, y)$

Propriété :Existence d'un ACM

Tout graphe non-orienté, valué et connexe admet un ou plusieurs ACM.

Construire des ACM

Algorithme 1 : algorithme de Kruskal.

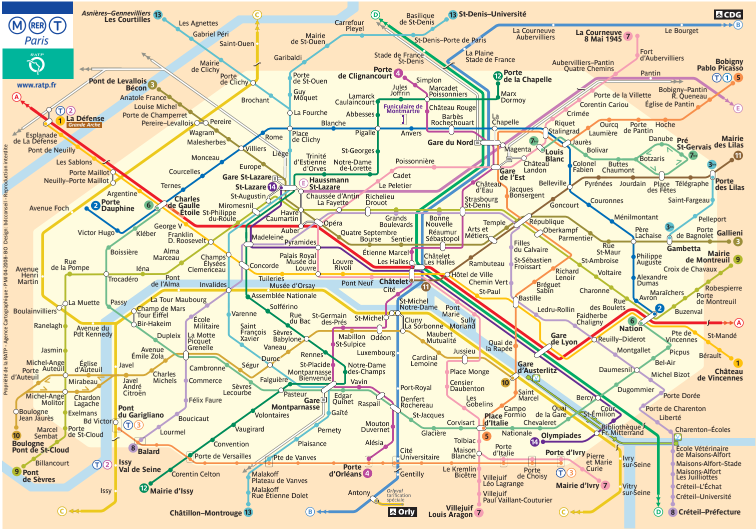
Algorithme 2 : algorithme de Prim.

Des algorithmes très efficaces : en $O(|A| \cdot \log(|A|))$ et $O(|A| \cdot \log(|S|))$.
(mais le choix des structures de données est important).

Plan

Trouver un chemin optimal

Comment aller de Chevaleret à Porte de Bagnolet ?



Plus courts chemins

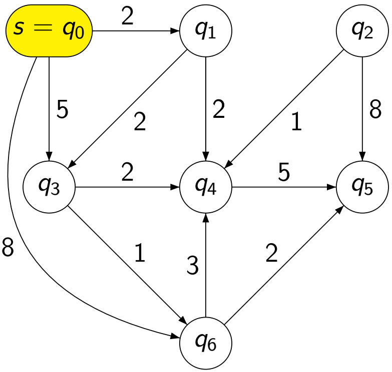
Objectif : Etant donné un graphe orienté et valué, et un sommet s, on veut trouver :

- la distance minimale séparant s de chaque sommet, et
- des plus courts chemins reliant s aux autres sommets.

Plus courts chemins

Objectif : Etant donné un graphe orienté et valué, et un sommet s, on veut trouver :

- la distance minimale séparant s de chaque sommet, et
- des plus courts chemins reliant s aux autres sommets.



Plus courts chemins

Objectif : Etant donné un graphe orienté et valué, et un sommet s, on veut trouver :

- la distance minimale séparant s de chaque sommet, et
- des plus courts chemins reliant s aux autres sommets.

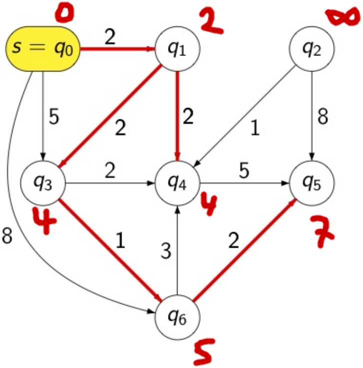
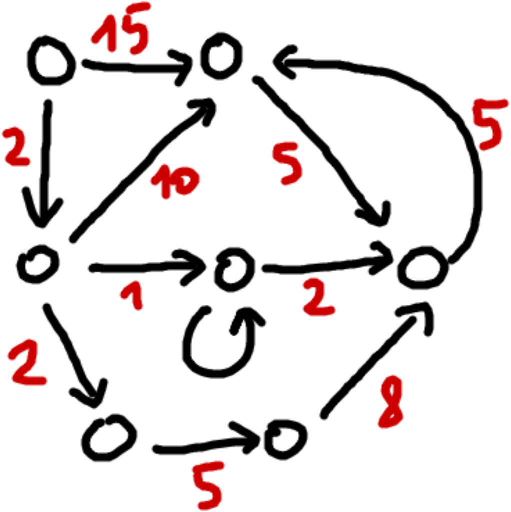


Table des prédécesseurs

$$
\begin{aligned}
\Pi[q_0] &= \text{nil} \\
\Pi[q_1] &= q_0 \\
\Pi[q_2] &= \text{nil} \\
\Pi[q_3] &= q_1 \\
\Pi[q_4] &= q_1 \\
\Pi[q_5] &= q_6 \\
\Pi[q_6] &= q_3
\end{aligned}
$$

Exemple



L'algorithme de Dijkstra

$G = (S, A, w)$: orienté et valué ($w : A \to \mathbb{R}_+$).
s un sommet initial.

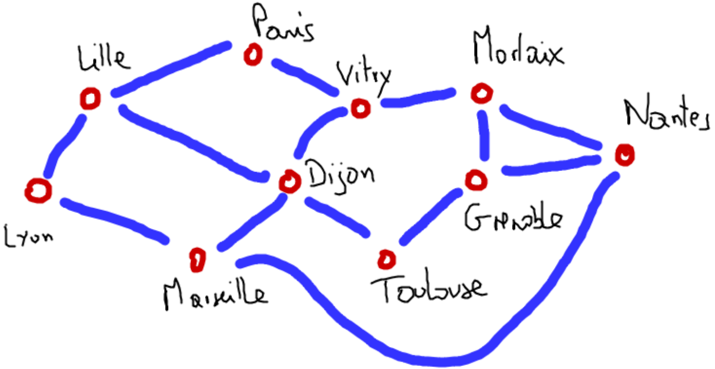
On va construire un arbre des plus courts chemins...
et calculer les distances $\delta(x)$: distance d'un PCC de s à x...

1. La distance de s à lui-même est 0 : $\delta(s) = 0$
   C'est la racine de l'arbre.
2. Trouver un sommet s' situé à une arête de l'arbre en construction et le plus proche possible de s.
3. Marquer l'arête (x, d, s') qui a permis de trouver s'
4. La distance séparant s et s' est : $\delta(x) + d$.
5. Recommencer au point **2** tant qu'il reste des sommets atteignables...

Plan

Le problème des bandits

Problème : Des bandits veulent visiter toutes les villes d'une région sans jamais repasser par une ville.



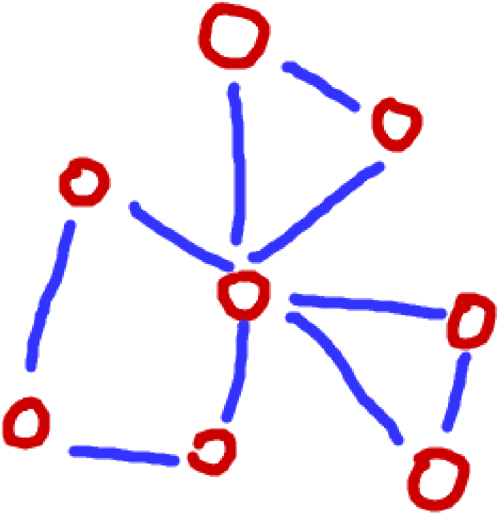
Aie !

On ne connaît pas d'algorithme efficace pour ce problème. . .

C'est un problème NP-complet !

La recherche de chemin Hamiltonien.

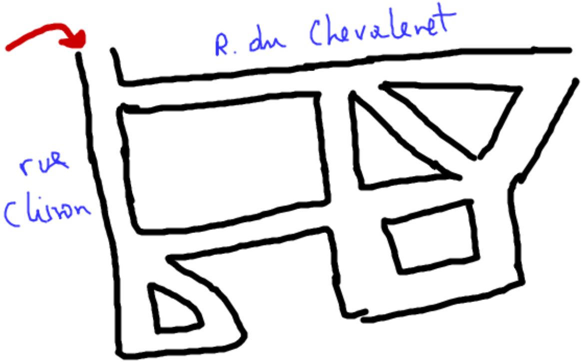
Le problème des bandits



Plan

Le problème de la goudronneuse

Problème : On veut faire passer une goudronneuse une et une seule fois par chaque rue.
(on s'autorise à repasser aux croisements. . .)



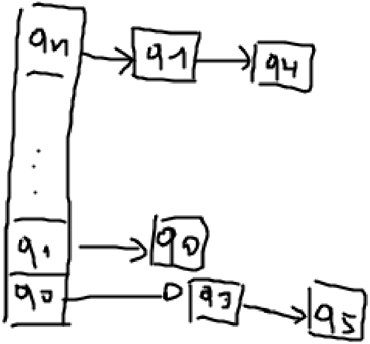
Chemin Eulérien. . .

Très facile !

Plan

Représentation des graphes - 1

Par liste d'adjacence :



Représentation des graphes - 1

Par matrice d'adjacence :

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Graphes valués

$G = (S, A) \quad + \quad w : A \to \mathbb{R}.$

La fonction w associe un poids, une distance, etc. à chaque arc ou arête de G.
(La fonction w s'étend naturellement à tout chemin (fini) de G en sommant le poids de chaque arc/arête.)

Les deux représentations précédentes s'étendent facilement aux graphes valués.

Représentation des graphes en PYTHON

```
S1=['q0','q1','q2','q3','q4','q5','q6']
A1={'q0': ['q1','q3'], 'q1': ['q3','q4'],'q2': ['q4','q5'],
'q3': ['q1','q3','q4','q6'],'q4':[ ],'q5': ['q4','q5'],
'q6': ['q4'] }
G = (S1,A1)
```