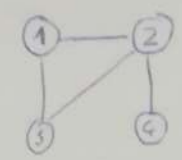
Accessibilités et chemins dans un graphe.

I Définitions et représentations des graphes

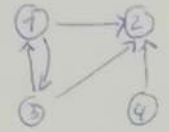
Définition : $G=(S,A)$ graphe non orienté
S ensemble des sommets.
A ensemble des arêtes $a \in A$ et $\exists a=(u,v)$ $u,v \in S$ $u \neq v$

Définition : $G=(S,A)$ graphe orienté
A ensemble des arcs
distinction entre (u,v) et (v,u)



G graphe non orienté

Graphe complet : graphe non orienté tel que
toute paire de sommet est reliée par une arête.



G graphe orienté

degrés d'un sommet : on note $d(u)$ $u \in S$
le nombre de sommet voisin de u dans G non orienté
$d(u) = |\{v \in S \mid (u,v) \in A\}|$
$d^+(u)$, $d^-(u)$ degrés sortant, entrant de u.

Chaine / chemin : $x_1, x_2 \dots x_{n-1}\ x_n$ tel que $\forall 1 \leq i \leq n-1$ $(x_i, x_{i+1}) \in A$.
chemin simple : chemin tel que les sommets sont différents (sauf ci dessus).

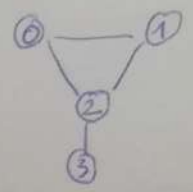
Connexité : $G=(S,A)$ non orienté est connexe si il existe un chemin pour tout $u, v \in S$. $G=(S,A)$ orienté si chaine entre u et v et v et u pour tout $u, v \in S$.

Fonction de pondération : $w : (u,v) \mapsto \mathbb{R}$ on munit chaque arête d'un poids. Ex : Distance entre deux sommets.

Représentation :

Matrice d'adjacence :

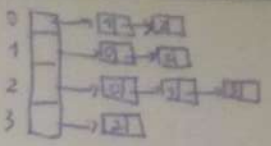
$$M_{i,j} = \begin{cases} 1 \text{ si } (i,j) \in A \\ 0 \text{ sinon} \end{cases}$$



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

taille $n^2 = |S|^2$

Liste d'adjacence :
tableau de $|S|$ liste.
$Adj[u]$ liste des sommets v voisin de u
taille $|A|$



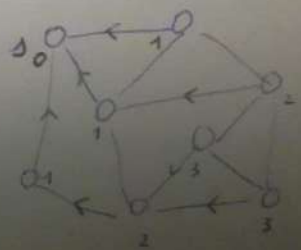
Exemple d'application :
Problème d'Euler : Trouver un chemin qui passe par toutes les arêtes une fois pour arête
Propriété : G un graphe connexe, G est un graphe eulérien si et seulement si il n'admet aucun sommet de degré impair.

II Parcourt dans les graphes

Parcourt en largeur

étant donnés un graphe $G=(S,A)$ et s un sommet origine
le parcourt en largeur parcourt systématiquement les arcs de G pour trouver l'ensemble des sommets accessible depuis s.

Parcourt-en-largeur (G,s) : complexité $O(S+A)$ si représentation par listes d'adjacence.
pour chaque sommet $u \in S$ $u \neq s$
couleur$[u] \leftarrow$ Blanc
$d[u] \leftarrow \infty$ # distance
$\pi[u] \leftarrow$ NIL # parents dans l'arborescence.
couleur$[s] \leftarrow$ Gris
$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ NIL
$F \leftarrow \{s\}$. # F : fifo
tant que $F \neq \emptyset$
$u \leftarrow$ tête$[F]$
Pour v tq $(u,v) \in A$:
Si couleur$[v]$ = Blanc
couleur$[v] \leftarrow$ Gris
$d[v] \leftarrow d[u]+1$
$\pi[v] \leftarrow u$
Enfiler(F,v)
couleur$[u] \leftarrow$ Noir.



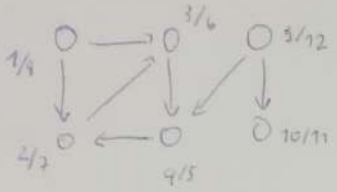
Parcours en profondeur

"On avance d'abord et va aller le plus loin possible !"

Créer une forêt de parcours en profondeur.

```
parcours-profondeur (G)
    pour chaque sommet u ∈ S
        couleur[u] ← Blanc
        π[u] ← Nil
    date ← 0
    pour chaque sommet u ∈ S
        Si couleur[u] = Blanc
            visiter-PP(u)

visiter-PP(u)
    couleur[u] ← Gris
    date ← date + 1
    d[u] ← date
    pour tout v tq (u,v) ∈ A
        Si couleur[v] = Blanc
            π[v] ← u
            visiter-PP(v)
    couleur[u] ← Noir
    f[u] ← date
    date ← date + 1
```



Tri topologique

$G = (S, A)$ orienté sans circuit. ordonne linéairement tous les sommets tq si $(u,v) \in A$. u apparaisse avant v dans le tri

```
Tri-topologique (G)
    parcours-profondeur (G)
    trier par f[u] par ordre décroissant
```

III Plus court chemin dans les graphes

Problème : Plus court chemin à origine unique.

On cherche la distance la plus courte entre s et tous les autres sommets du graphe.

→ En particulier on aura la distance la plus courte entre s et y deux sommets choisis

Dijkstra : algorithme qui résout le problème si les poids sont à valeur positive.

Dev 1	présentation de l'algorithme implémentation

Bellman-Ford : algorithme dans le cas général qui détecte les circuits absorbant s'il y en a. $O(SA)$

Autres problèmes possibles : obtenir la distance la plus courte entre toutes paires de sommet.

Algorithme de reconstruction du plus court chemin.

IV Autres applications intéressantes

Arbre couvrant de poids minimal :

Définition : $G'(S,A')$ sous-graphe ensemble d'arêtes de somme de poids minimal qui connecte tous les sommets

Algorithme de Kruskal permet de déterminer un arbre couvrant de poids minimal

Kruskal (G, w)
trier les arêtes par ordre de poids croissant
Tant que l'on n'a pas retenu $n-1$ arêtes
prendre la première arête non considérée
(u,v)
Si (u,v) ne crée pas de cycle avec les arêtes précédemment sélectionnées
sélectionner l'arête (u,v).

Problème du voyageur de commerce

problème De décision associé NP-Complet

Approximation dans le cas euclidien
2-approximation simple avec les arbres couvrant
3/2-approximation : Algorithme de Christofides

Modélisation en PLNE et variante du problème Dev 2