ASD2

- la taille c'est le nombre d'arêtes
- l'ordre est le nombre de sommet
- graphe simple : pas de boucle et pas plus d'une arête par paire de sommet
- graphe partiel: on supprime quelques arêtes
- sous graphe: on supprime quelques sommets
- des sommets sont **adjacents** s'il y a une arête entre eux
- Graphe complet si tous les commets sont connecté entre eux

Composantes connexes

La relation de connectivité est réflexive, transitive et symétrique.

Cycles d'Euler et Hamilton

Un graphe connexe admet un cycle Eulérien si et seulement si il n'a pas de sommet de degré impair. Le premier et le dernier sommet peuvent être di érents, et dans ce cas de degré impair.

Un cycle Eulérien passe une et une seule fois par chaque arrête et un cycle Hamiltonien passe une et une seule fois par chaque sommet.

Chaîne Eulérienne: Le premier et le dernier sommet peuvent être di érents, et dans ce cas de degré impair.

Expressions Lambdas

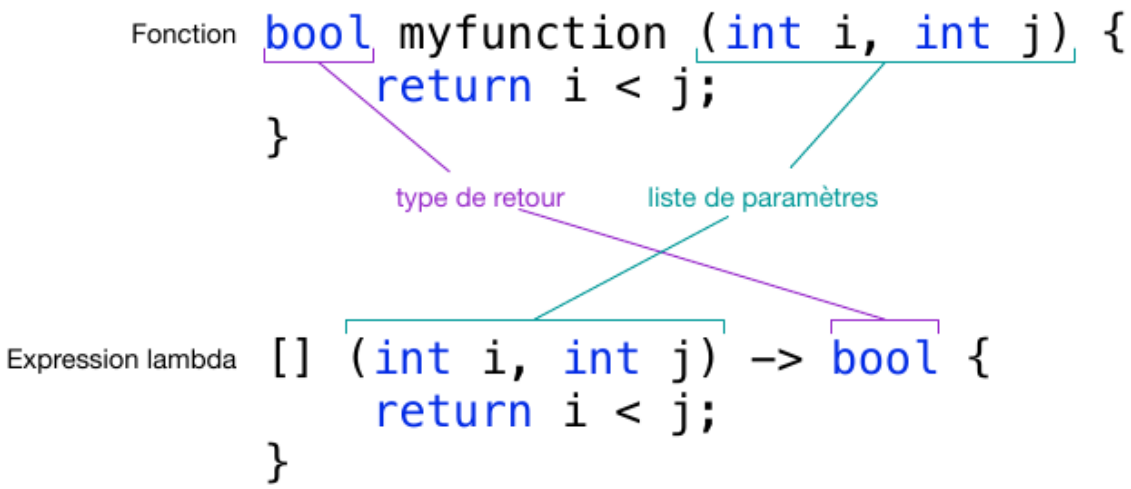


Figure 1:

Graphe orienté

Représentation par matrice d'adjacence **O(V)** efficace seulement si le graphe est complet. Par listes d'adjacences **O(degré(v))**.

Parcours avec un DFS pour savoir quels sont les sommets atteignables depuis un sommet. Pour savoir le plus court chemin il faut faire un BFS

- sommet demi-degré intérieur : flèches entrantes

- sommet demi-degré extérieur : flèches sortantes
- Tri topologique : trier les sommets pour que les arcs pointent vers le haut
- connectivité forte : chemin entre tous les couples de sommets La matrice n'est pas symétrique.
- composantes fortement connexe: est un ensemble de sommet fortement connectés
- il y a deux types d'arcs:
  - ceux qui restent dans la même CFC
  - Ceux qui mènent d'une CFC à une autre
- Si on regroupe tous les sommets d'une même CFC on obtient un graphe acyclique

Tri topologique

Trier les sommets de manière à ce que tous les sommets pointent dans la même direction. N'est possible que pour un graphe acyclique. Se trouve avec un DFS post-ordre inversé.

Pour détecter un cycle on utilise la pile récursion. si dans le parcours on tombe sur un sommet se trouvant déjà dans la pile --> cycle.

Connectivité forte

Deux sommets sont fortement connectés s'il existe un chemin dans un sens et dans l'autre. C'est une relation d'équivalence si elle est réflexive, symétrique et transitive. Une CFC est un ensemble maximal de sommets fortement connectés.

On distingue deux types d'arcs:

- ceux qui restent dans la même CFC, Si on regroupe tous les sommets d'une même CFC on obtient un graphe acyclique. De cette manière on peut le trier topologiquement.
- ceux qui mènent dans une CFC à l'autre

Un graphe G et son graphe inverse ont les mêmes composantes fortement connexes. **L'algorithme de Kosaraju-Sharir calcul le post ordre inverse du parcours en profondeur d'un graphe inverse.**

Algorithme de kosaraju-sharir

Calcul du post-ordre inverse DFS sur le graphe inverse.

Union find

Recalcul des composantes connexes

Lorsque le graphe n'a aucune arrête, chaque sommet forme une CC différente qu'on étiquette par le numéro du sommet. Chaque sommet pointe sur une sommet représentatif

- quick find pour savoir la CC d'un sommet **O(1)**
- connected savoir si deux sommets sont de la même CC
- union unir deux CC **O(V)**

Faire pointer chaque sommet sur un autre sommet de la CC et identifier chaque sommet par un sommet représentatif qui pointe sur lui-même.

complexité en log*(n) augmente très lentement

Minimum spanning Trees

Sur un graphe dont les arrêtes ont des poids positifs. Un arbre couvrant est un sous-graphe qui est **Connexe, acyclique et qui inclus tous les sommets**. Trouver un arbre couvrant tous les sommets d'un poids minimum.

- sans cycle et avec n-1 arrêtes
- connexe et admet n-1 arrêtes
- sans cycle, en ajoutant une arrête on crée un et un seul cycle élémentaire
- connexe, en supprimant une arrête quelconque, il n'est plus connexe.
- il existe une chaîne et une seule entre 2 sommets.

Algorithme glouton

Un algorithme qui suit le principe de faire, étape par étape, un choix optimum local.

La coupe d'un graphe est une partition en deux sous ensemble disjoints et exhaustifs. C'est aussi un ensemble des arêtes ayant un sommet dans chacune des deux partitions.

- On choisi une coupe sans arrête déjà dans le MST recommencer jusqu'à que V-1 arrêtes soient rouges.

Algorithme de Kruskal

Fait croître jusqu'à couverture totale. On considère les arrêtes par liste de poids croissant. si l'arrête ne fait pas de cycle, on l'ajoute. On teste si l'arrête ajoutée crée un cycle avec des union-find avec une complexité de O(log*V) qui augmente très lentement. Quand on ajoute une arrête avec les sommets v et w on appel union(v,w) si ils ne sont pas connected(v,w). Puis on utilise une priority queue pour trier les arrêtes.

Le calcul du MST se fait en temps proportionnel de E * log(E) mais si les arrêtes sont déjà ordonné, cela ne prend que E * log(V) qui est linéaire en E.

Algorithme de Prim

Version paresseuse

- On fait une PQ avec toutes les arrêtes connectées à l'arbre courant par au moins un sommet.

Version stricte

- On fait une PQ avec les sommets, la priorité de ces derniers est le poids le plus petit parmi les arêtes connectant à ce sommet à l'arbre.

Quand on sort un sommet v de la PQ, on ajoute l'arête v-w associée à l'arbre et l'on considère les
arêtes w-x adjacentes. Si x est dans l'arbre, on l'ignore Si x est absent de la PQ, on l'ajoute avec la priorité du poids de w-x Sinon, on abaisse la priorité de x si le poids de w-x est plus bas que le poids dans la PQ

Chemin les plus courts

Terminologie : raccourcir le chemin = "relâcher" un ressort imaginaire qui va du sommet source à w, parce qu'on a trouvé un chemin plus court qu'auparavant

Graphes acyclique

Traiter les sommets par ordre topologique.

Algorithme de Dijkstra

On traite une seule fois chaque sommet, on traite les sommets par ordre de distance croissante. **Seulement pour les poids non-négatifs**. O(E*log(V))

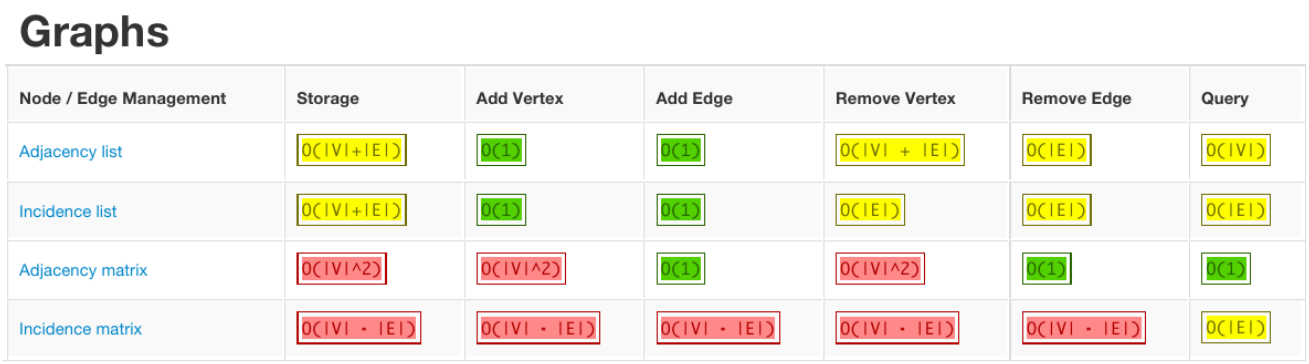
Algorithme de Bellman-Ford

On s'arrête à V-1 passages. Faire attention au circuit absorbant car on peut avoir des poids négatifs. Si après V-1 passage on peut toujours diminuer une distance entre deux sommets, il y a un circuit absorbant. On relâché1 plusieurs fois chaque sommet. O(E*V)

Complexité

Le parcours en largeur sur matrice d'adjacence O(V*V) et par liste d'adjacence O(V+E).

Méthode reverse: O(N). insertion set O(log(N)),



Graphs

Node / Edge Management	Storage	Add Vertex	Add Edge	Remove Vertex	Remove Edge	Query
Adjacency list	O(|V|+|E|)	O(1)	O(1)	O(|V| + |E|)	O(|E|)	O(|V|)
Incidence list	O(|V|+|E|)	O(1)	O(1)	O(|E|)	O(|E|)	O(|E|)
Adjacency matrix	O(|V|^2)	O(|V|^2)	O(1)	O(|V|^2)	O(1)	O(1)
Incidence matrix	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|E|)

Figure 2:

- L'algorithme de Kruskal calcule le MST en un temps proportionnel à *E log E*

- Si les arêtes sont déjà ordonnées, l'algorithme de Kruskal ne nécessite que *E log* V*, ce qui en pratique est linéaire en *E*

Opération	Fréquence	Coût
pq.push(e)	E	log E
pq.pop()	E	log E
uf.Union(v,w)	V	Log* V
uf.Connected(v,w)	E	Log* V

Figure 3:

Dijkstra

- Avec une mise en œuvre stricte, on relaxe chaque arc une seule fois, et gérer la queue de priorité coûte log(*V*) par opération, pour une complexité totale O(*E**log(*V*))

Figure 4:

Quiz

Pour un graphe creux, il est préférable de choisir une implémentation par listes d'adjacence.