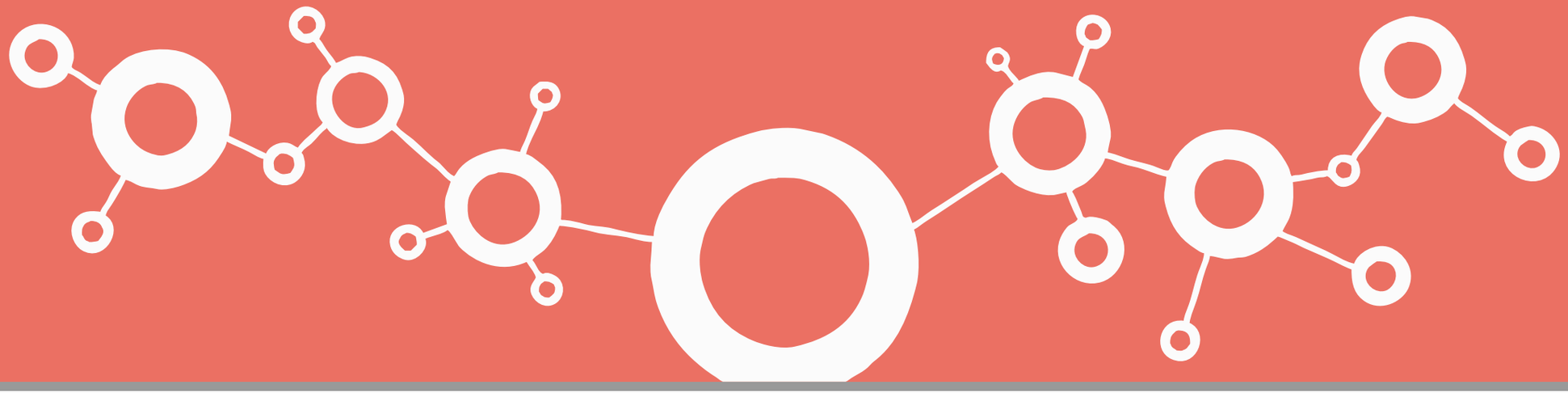


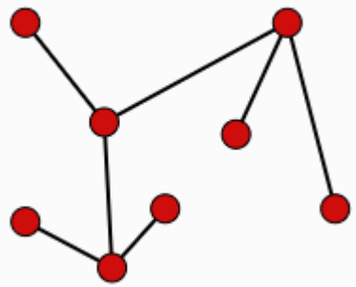
Ch. 7 : Parcours, arbres couvrants et plus courts chemins

7.1 Parcours de graphes

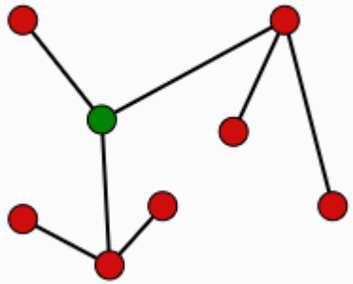
Rappels : arbres

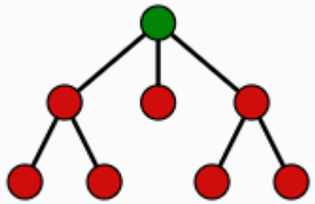
Arbre

- mathématiques : *graphe* connexe (= « en un morceau ») et sans cycle
- informatique : arbre *enraciné* ou *arborescence*



arbre





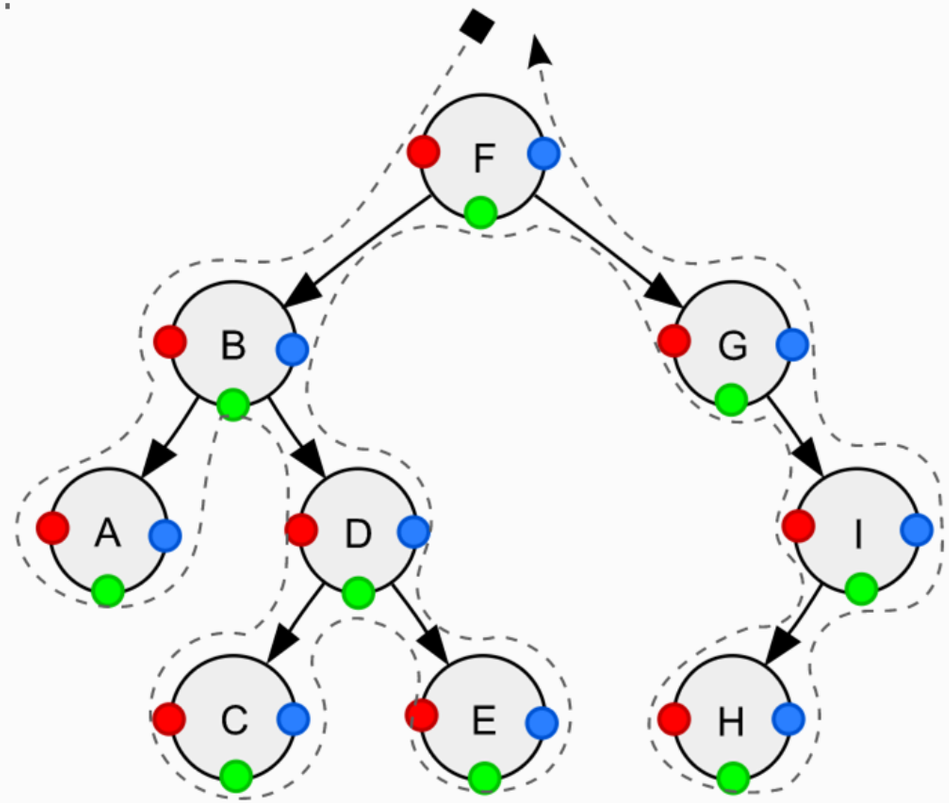
arbre enraciné ou arborescence

💡Un arbre est une généralisation d'une liste chaînée

Rappels : arbres

Il existe plusieurs manières de parcourir les nœuds d'un arbre :

- en *largeur* (càd par distance depuis la racine)
- en *profondeur* (càd en descendant le plus possible d'abord)



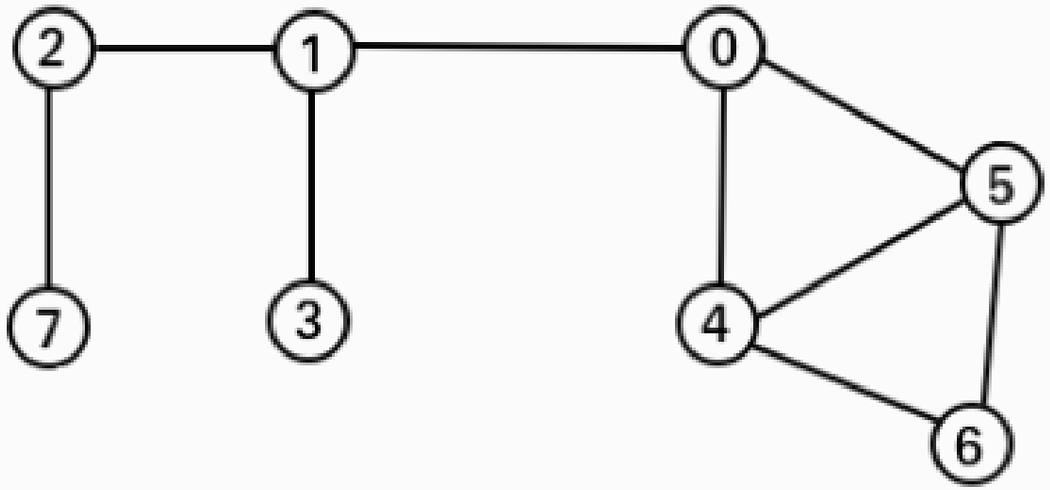
On peut généraliser ces deux types de parcours aux graphes

Parcours en largeur d'un graphe

OU BFS (BREADTH-FIRST SEARCH)

Principe : comme pour les arbres, on procède par *distance depuis un sommet de départ*

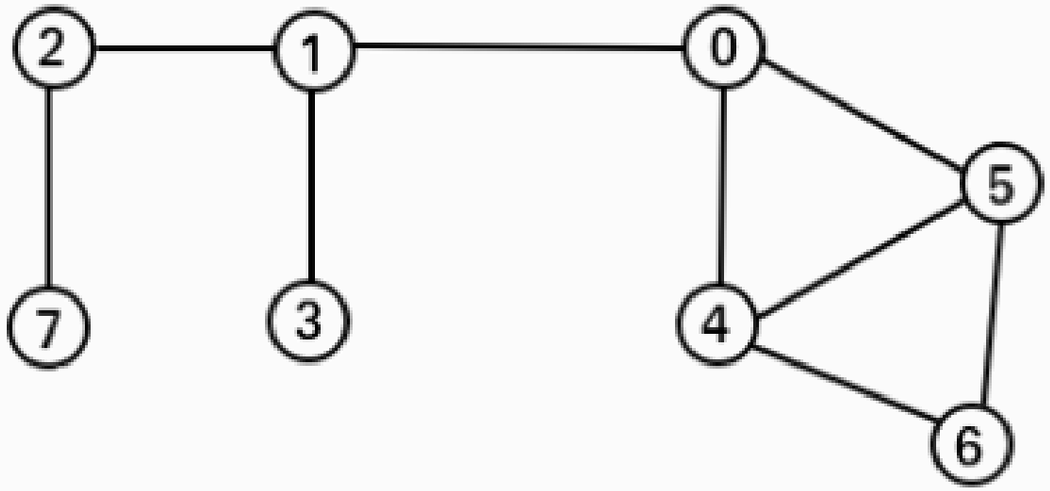
- on note un sommet de départ
- puis tous ses voisins
- puis tous les voisins de ses voisins
- etc.



Parcours en largeur

OU BFS (BREADTH-FIRST SEARCH)

Implémentation : on utilise une file d'attente



Exemple : parcours en largeur depuis le sommet 0

File d'attente

0	1	4	5	2	3	6	7

Sortie

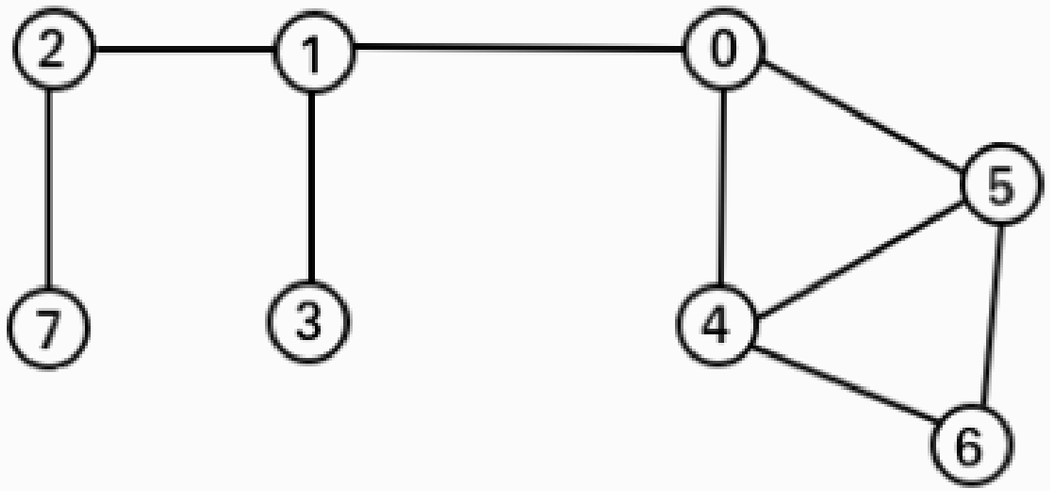
0 1 4 5 2 3 6 7

Parcours en profondeur

OU DFS (DEPTH-FIRST SEARCH)

Principe : comme pour les arbres, on procède en allant le plus loin possible avant de revenir en arrière jusqu'à pouvoir essayer un autre voisin (principe de la recherche de la sortie dans un labyrinthe) :

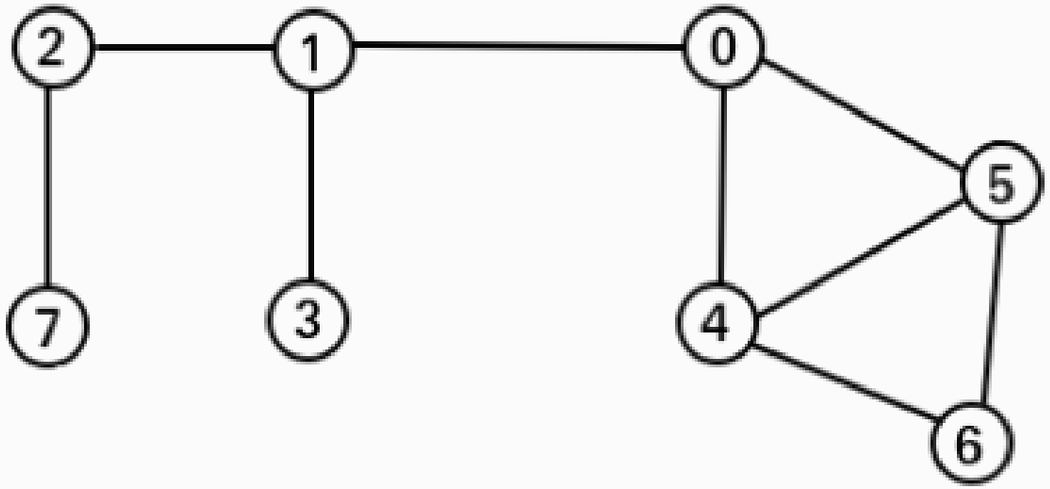
- on marque un premier sommet comme étant *visité*
- on considère son premier voisin non encore *visité*, et on recommence récursivement
- quand on a épuisé tous les voisins d'un sommet, on revient au sommet qu'on avait « mis en pause »



Parcours en profondeur

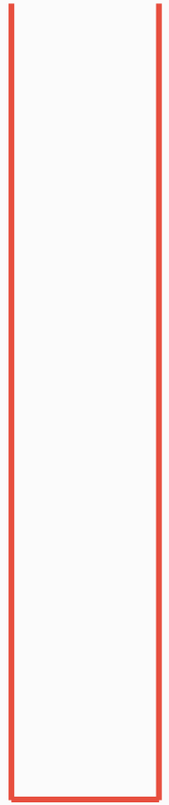
OU DFS (DEPTH-FIRST SEARCH)

Implémentation : on utilise une pile et un tableau pour mémoriser les sommets visités



Exemple : parcours en profondeur depuis le sommet 0

0 1 2 7 3 4 5 6



Pile

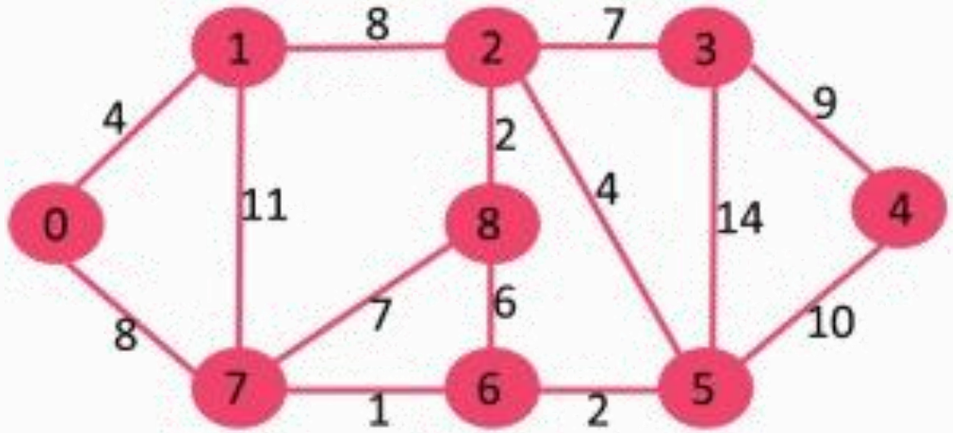
n°	Visité ?
0	
1	
2	
3	
4	
5	
6	
7	

7.2 Problème de l'arbre couvrant de poids minimum

Enigme du jour

APPLICATIONS

Une entreprise dispose d'un ensemble de centres, dont certains sont reliés entre eux par des routes (exprimées ici en dizaines de km) :



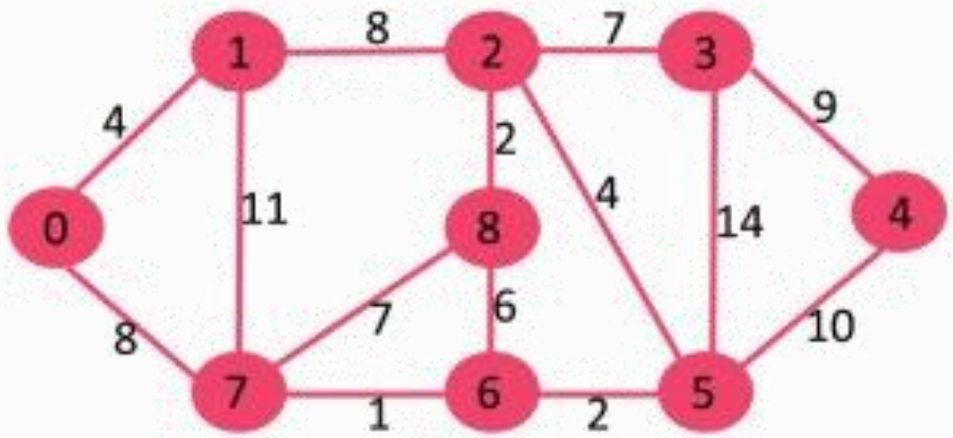
Cette entreprise souhaite connecter l'ensemble de ses centres par une liaison ultra rapide et hautement sécurisée. Chaque liaison ne peut se faire qu'en suivant les tronçons routiers (pour des raisons de coûts d'infrastructures) et le coût d'une liaison est proportionnel à la longueur de la route.

Quelle est la configuration optimale, càd celle qui coûtera le moins cher à l'entreprise ?

Arbre couvrant de poids minimal

MODELISATION

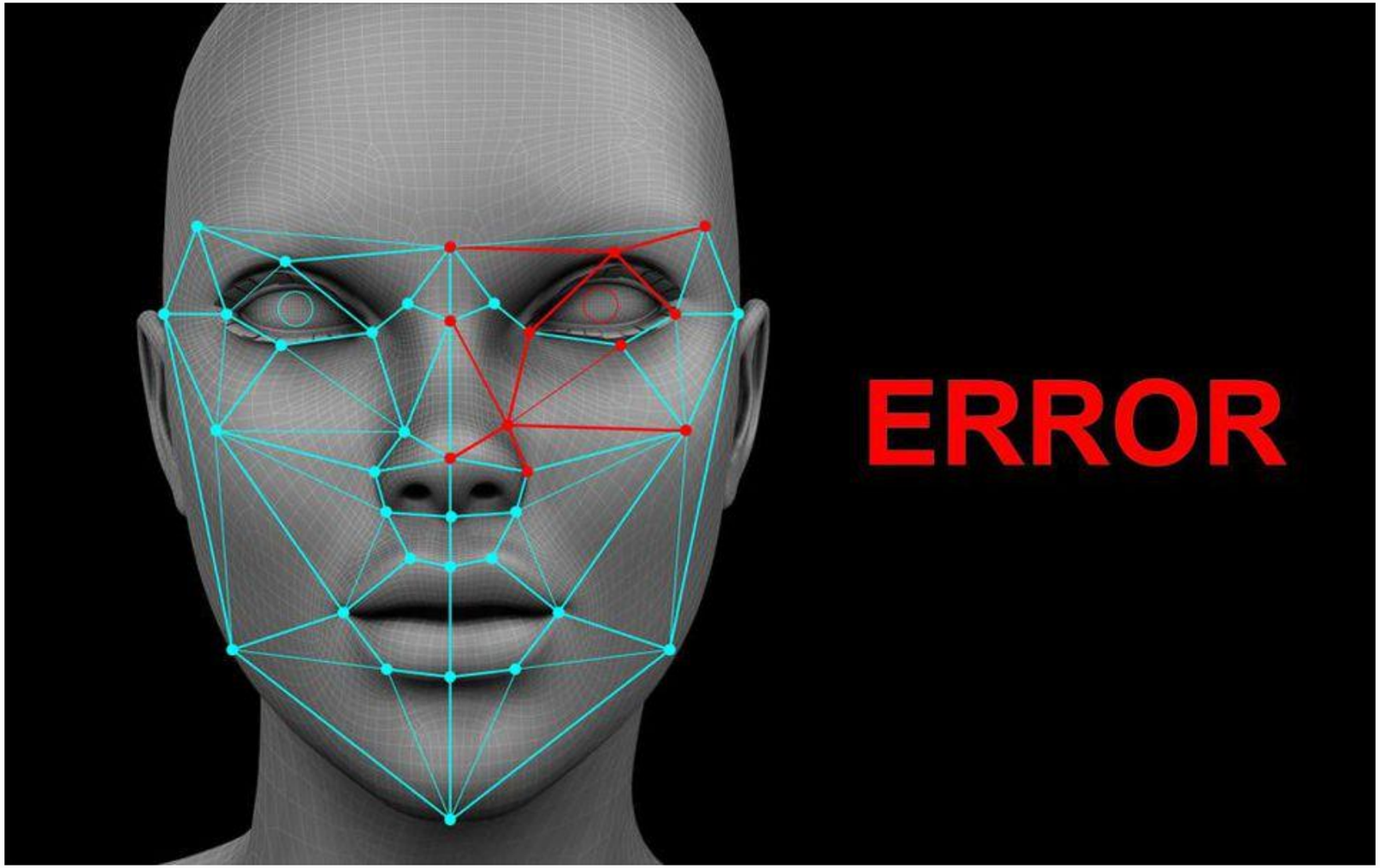
On modélise le problème par un graphe pondéré (ou *réseau*) *connexe* :



- Problème de l'arbre couvrant de poids minimum : trouver le sous-graphe *couvrant* (càd touchant tous les sommets) de poids minimum
- Pourquoi est-ce nécessairement un *arbre* ?

Arbre couvrant de poids minimal

UNE AUTRE APPLICATION



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

Algorithme de Kruskal :

1. Trier les arêtes par poids croissant
2. Prendre l'arête de poids minimum, et vérifier si elle forme un cycle avec l'arbre déjà construit :
   - si *non*, l'ajouter à l'arbre
   - si *oui*, la rejeter
3. Répéter l'étape 2 jusqu'à ce que tous les sommets soient reliés, càd avoir $n - 1$ arêtes dans l'arbre

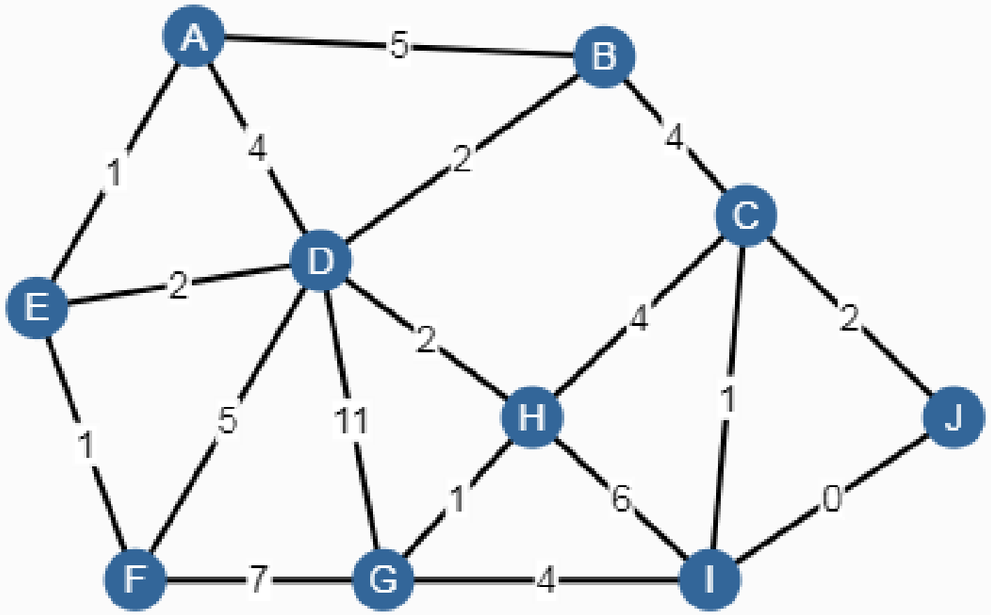
💡 Cas d'algorithme glouton optimal !

❓ Comment détecter automatiquement les cycles ?

Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

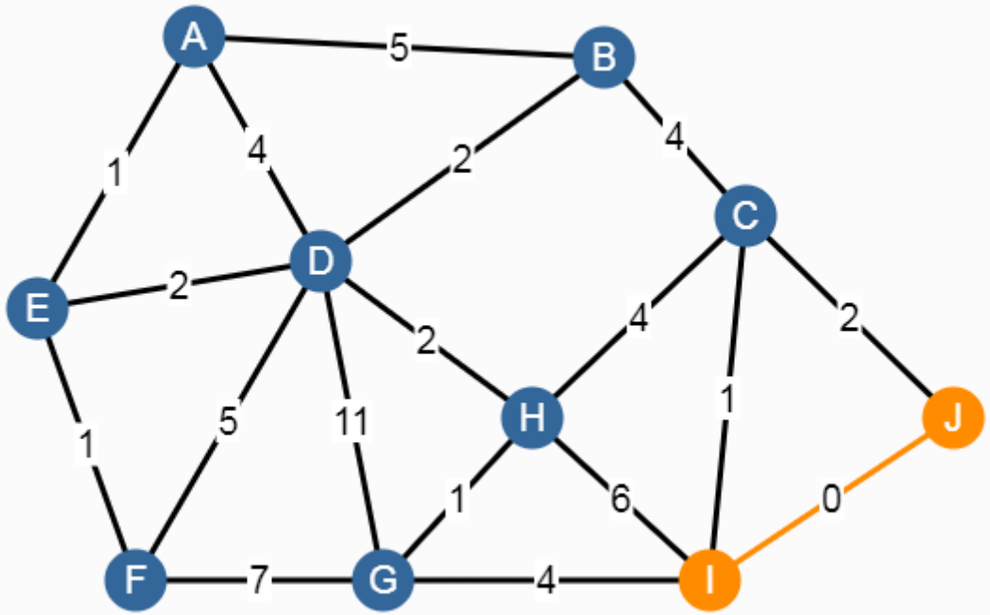
Exemple :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

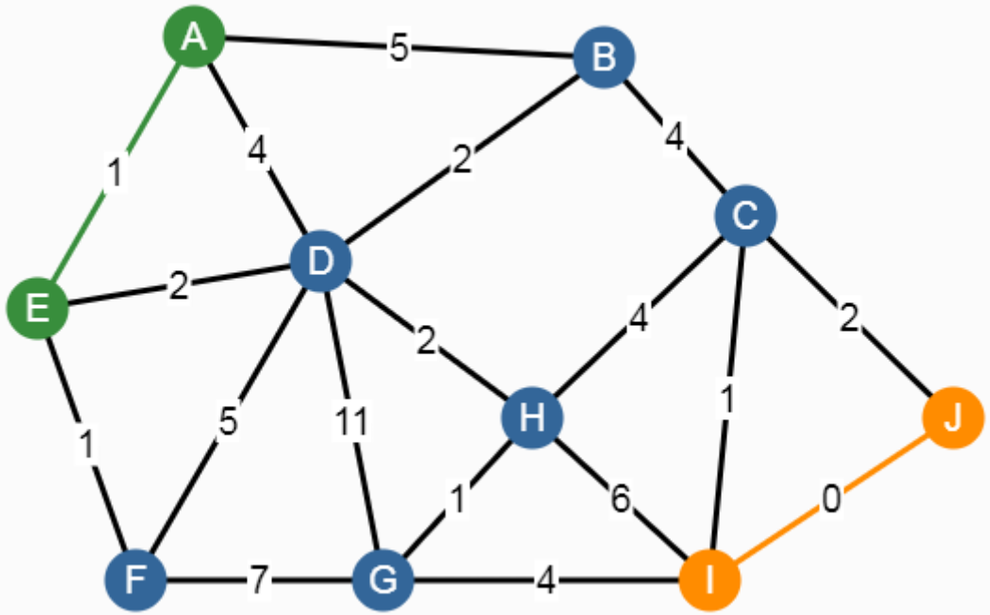
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

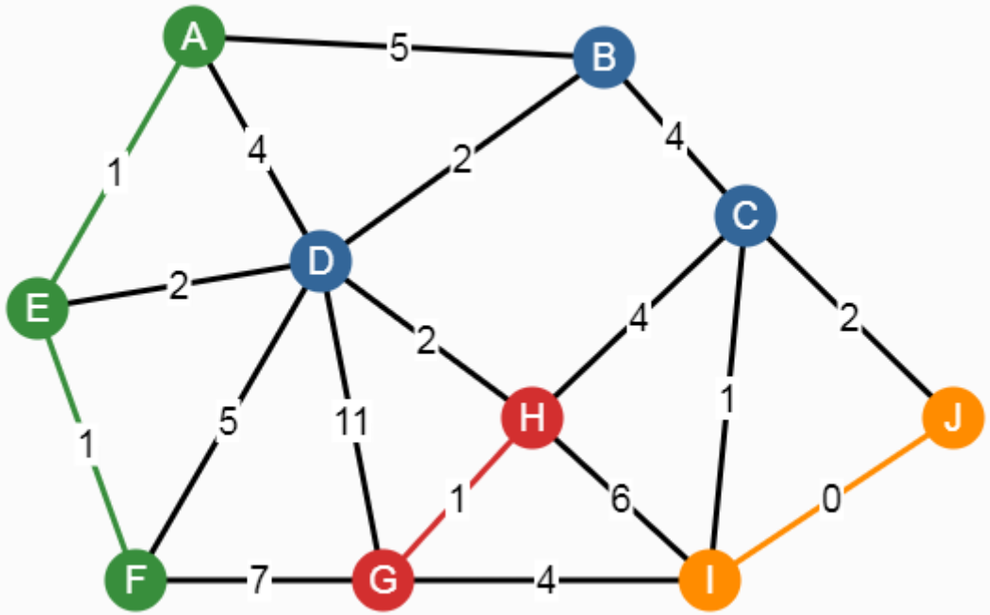
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

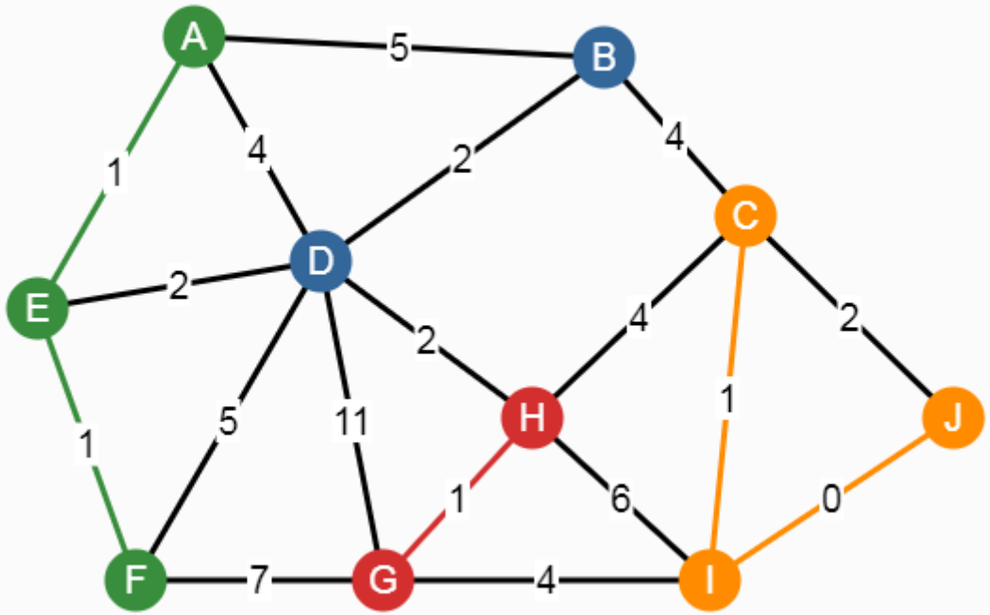
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

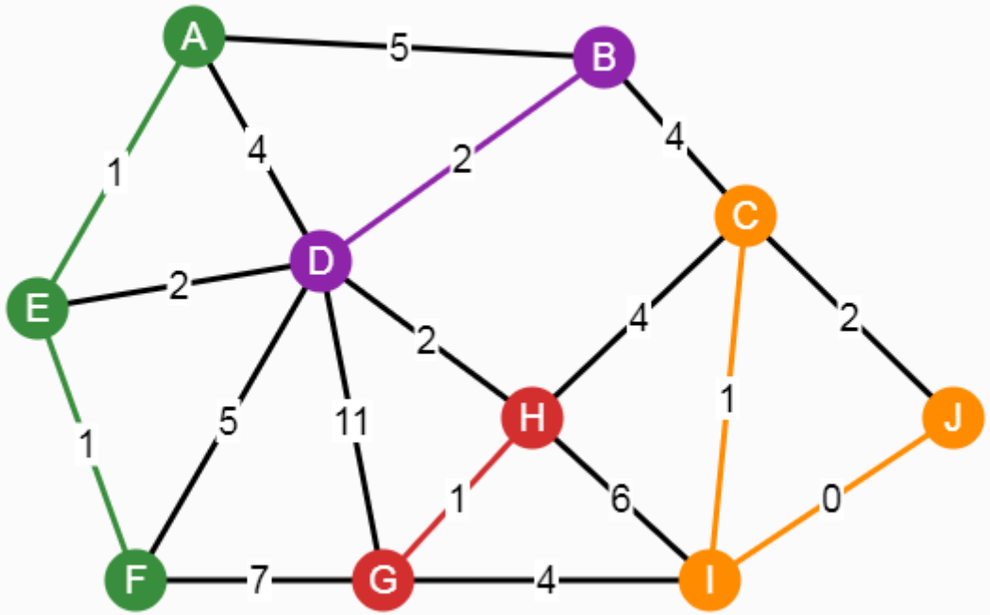
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

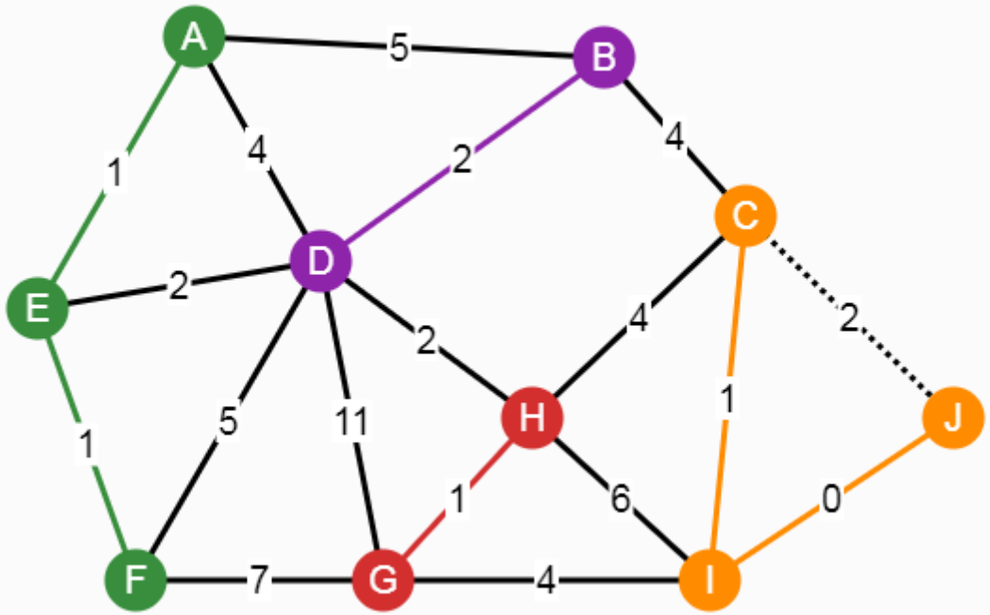
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

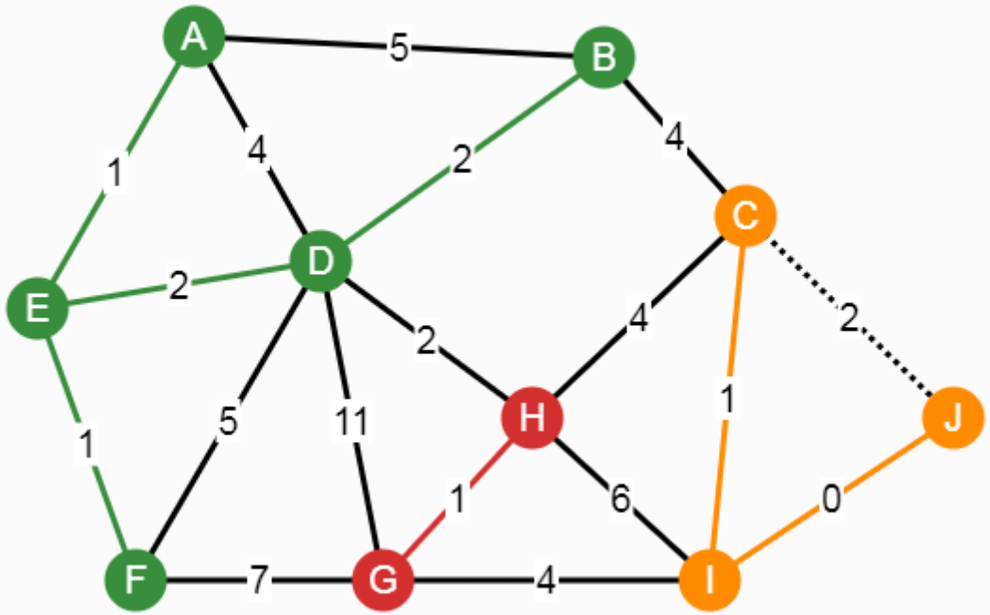
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

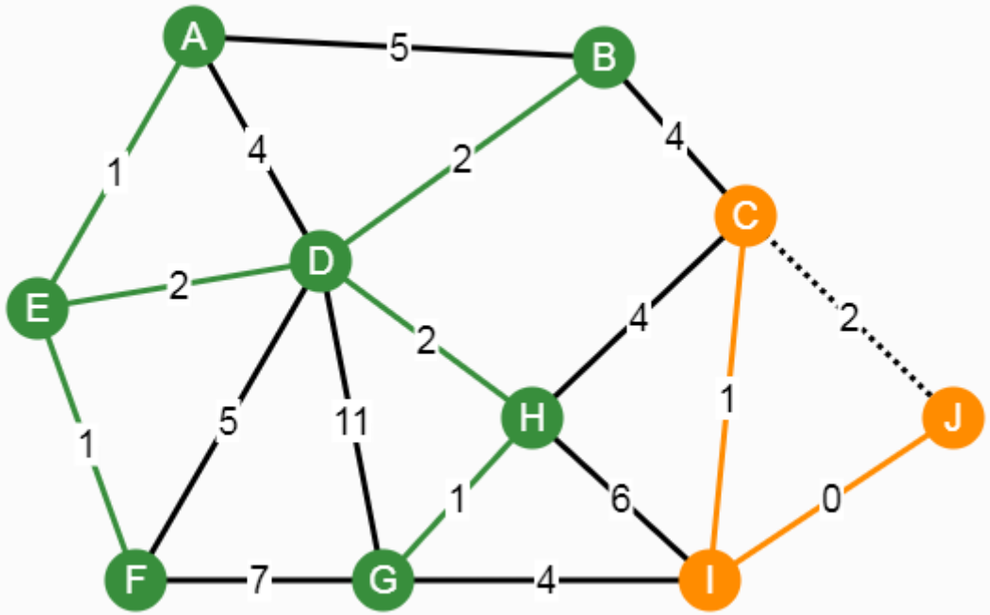
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

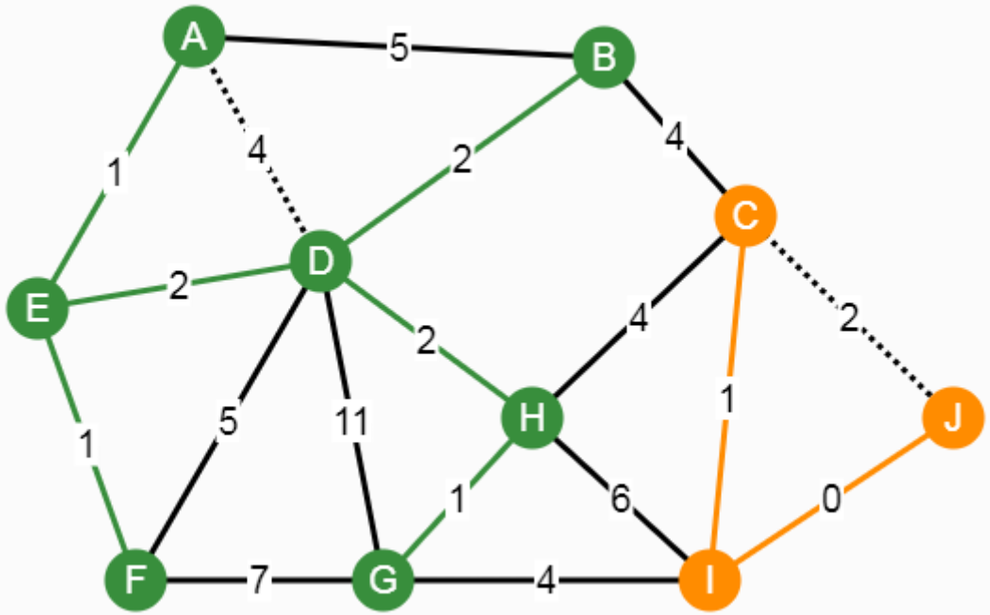
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

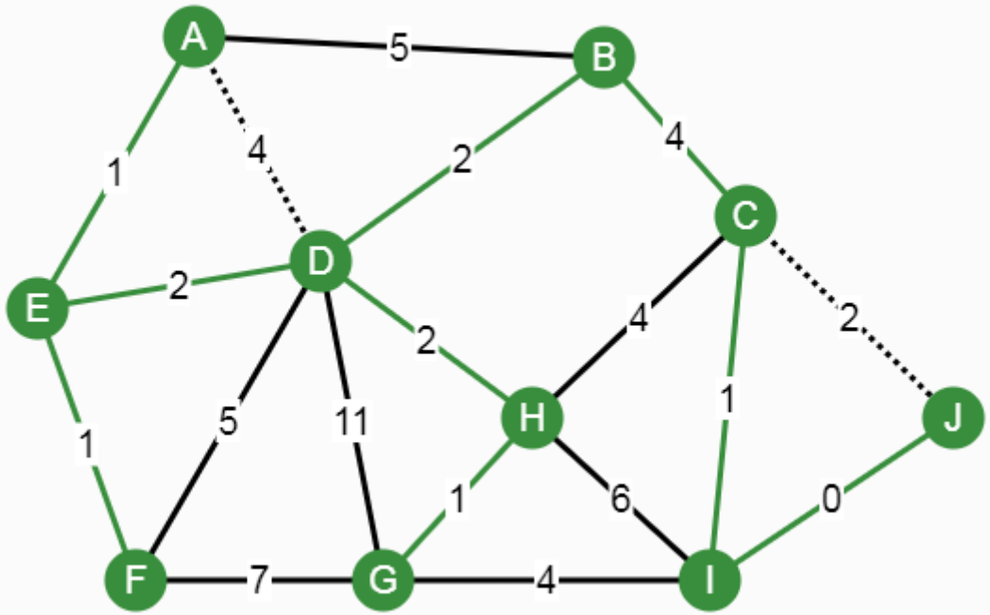
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

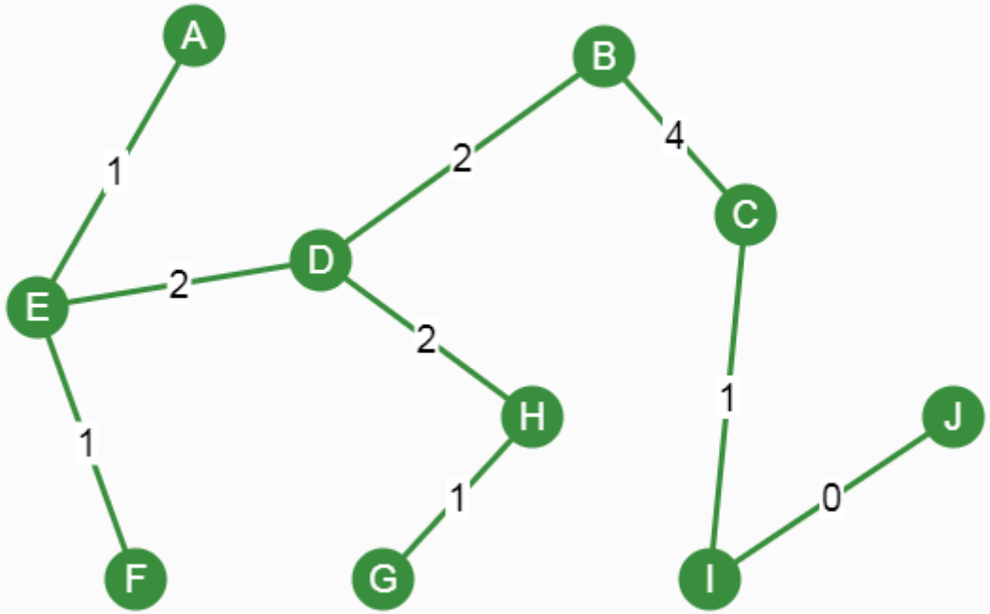
Solution :



Arbre couvrant de poids minimal

ALGORITHME DE KRUSKAL

Solution : arbre couvrant de poids 0 + 1 + 1 + 1 + 1 + 2 + 2 + 2 + 4 = 14



Arbre couvrant de poids minimal

ALGORITHME DE PRIM

Algorithme de Prim :

1. Initialiser l'arbre avec un sommet arbitraire
2. Faire croître l'arbre en prenant l'arête de poids minimal reliant un sommet de l'arbre et un sommet en dehors de l'arbre
3. Répéter l'étape 3 tant qu'il reste un sommet hors de l'arbre

Autre cas d'algorithme glouton optimal !

Arbre couvrant de poids minimal

ALGORITHME DE PRIM

Pseudocode :

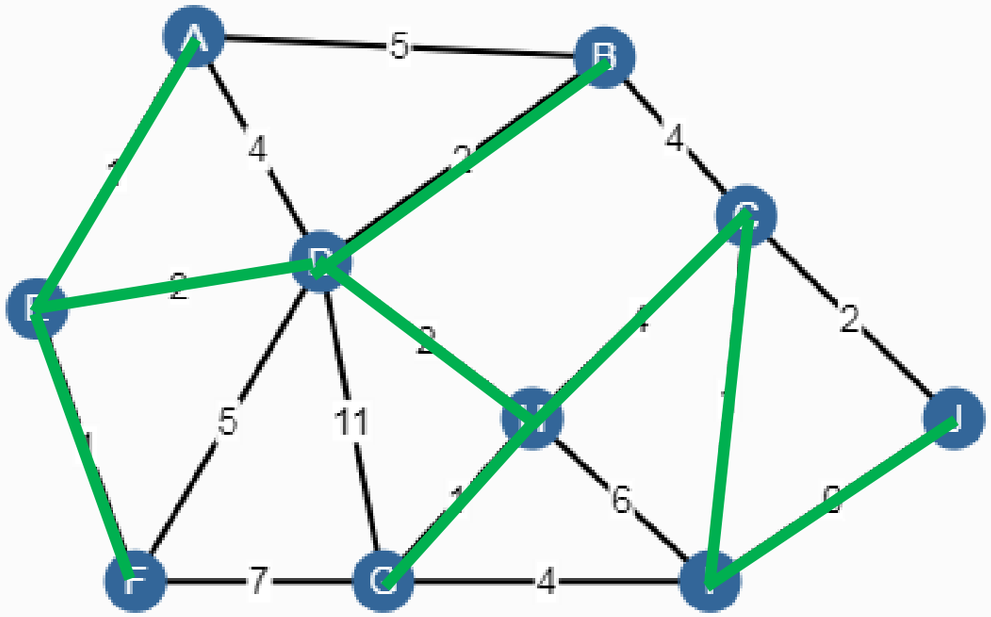
1. Créer un tableau *visités* afin de garder une trace des sommets visités, et un tableau *parents* afin de savoir quelles arêtes conserver
2. Attribuer à chaque sommet un *poids* égal à $+\infty$ sauf pour le sommet de départ, initialisé à 0
3. Tant qu'il existe un sommet qui n'est pas dans *visités*
   a) Prendre un sommet *v* absent de *visités* ayant le poids minimal
   b) L'ajouter à *visités*
   c) Mettre à jour les poids des voisins de *v* :

      pour chaque arête v-w, si le poids de cette arête est inférieur au poids actuel de w, mettre à jour le poids de w avec le poids de l'arête, et *parents[w]* = *v*

Arbre couvrant de poids minimal

ALGORITHME DE PRIM

Exemple : on part du sommet D

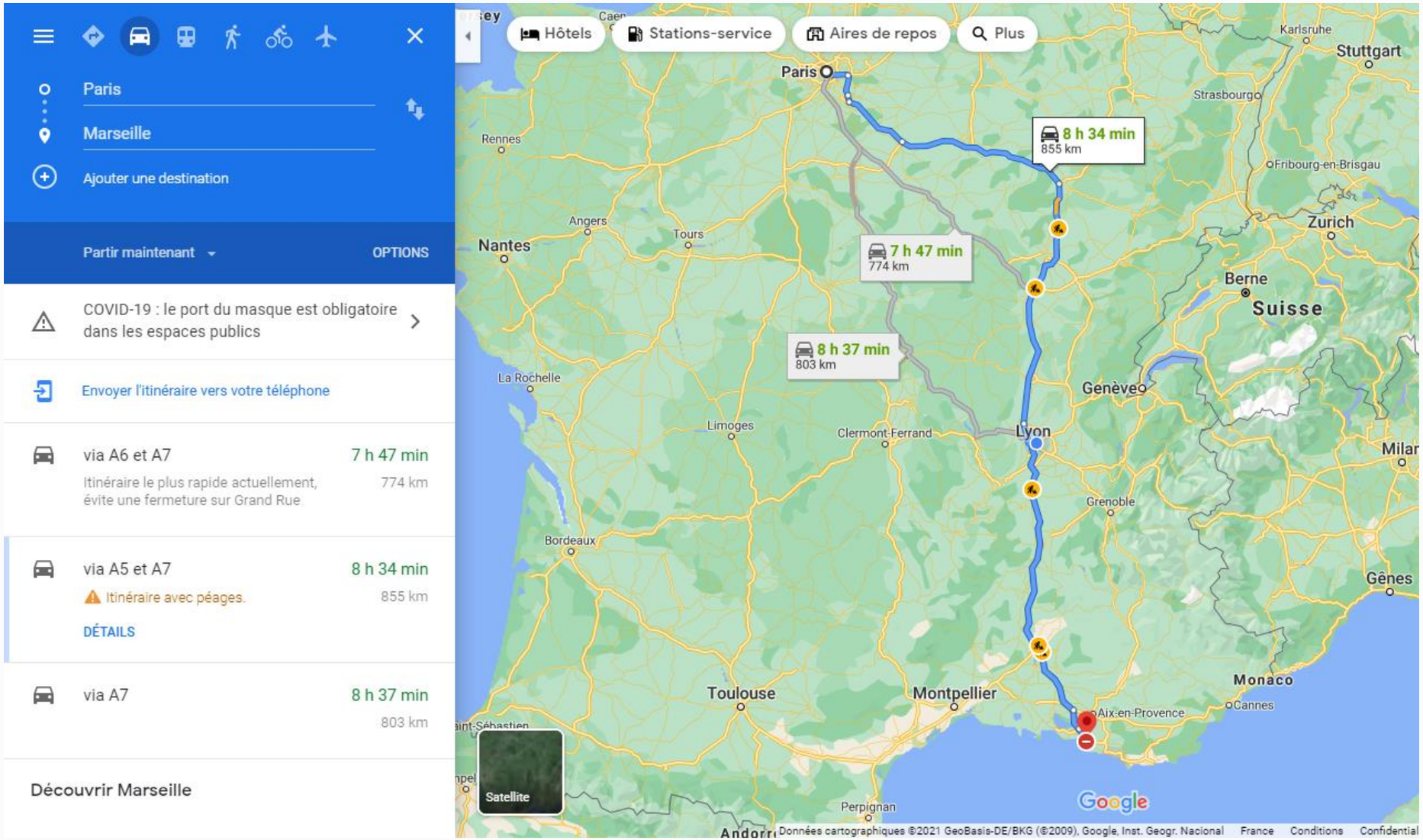


	A	B	C	D	E	F	G	H	I	J
Poids	∞	∞	∞	0 / D	∞	∞	∞	∞	∞	∞
Poids	4/D	2/D	∞	0	2/D	5/D	11/D	2/D	∞	∞
Poids	4/D	2/D	4/B	0	2/D	5/D	11/D	2/D	∞	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	11/D	2/D	∞	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	11/D	2/D	∞	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	7/F	2/D	∞	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	1/H	2/D	6/H	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	1/H	2/D	4/G	∞
Poids	1/E	2/D	4/B	0	2/D	1/E	1/H	2/D	1/C	2/C
Poids	1/E	2/D	4/B	0	2/D	1/E	1/H	2/D	1/C	0/I
Poids	1/E	2/D	4/B	0	2/D	1/E	1/H	2/D	1/C	0/I

7.3 Problème du plus court chemin entre deux sommets

Problème du plus court chemin

COMMENT FONCTIONNE GOOGLE MAPS ?



Problème du plus court chemin

ALGORITHME DE DIJKSTRA

Algorithme de Dijkstra :

1. Créer un tableau *PCC* afin de garder une trace des sommets visités
2. Attribuer à chaque sommet une *distance* $+\infty$, sauf pour le sommet de départ, initialisée à 0
3. Tant qu'il existe un sommet qui n'est pas dans PCC
   a) Prendre un sommet *v* absent de PCC ayant la distance minimale
   b) L'ajouter à PCC
   c) Mettre à jour les clés des voisins de *v* :

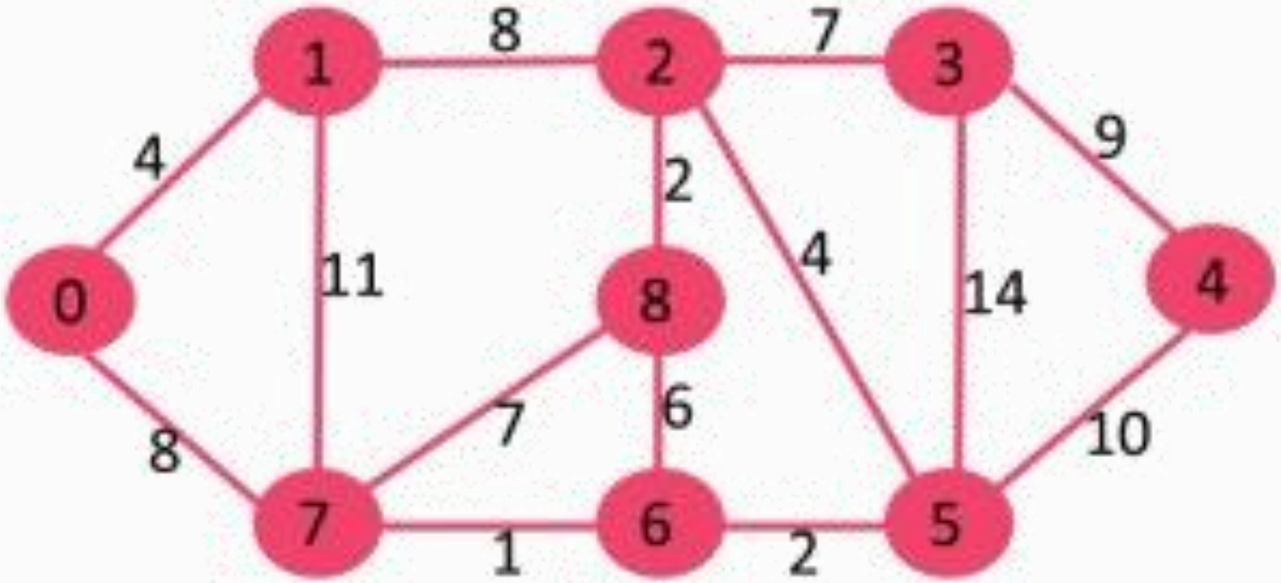
      pour chaque arête v-w, si $d(w) \geq d(u) + d(u, v)$, mettre à jour w

- Très similaire à l'algorithme de Prim !
- Donne le plus court chemin d'un sommet vers tous les autres

Problème du plus court chemin

ALGORITHME DE DIJKSTRA

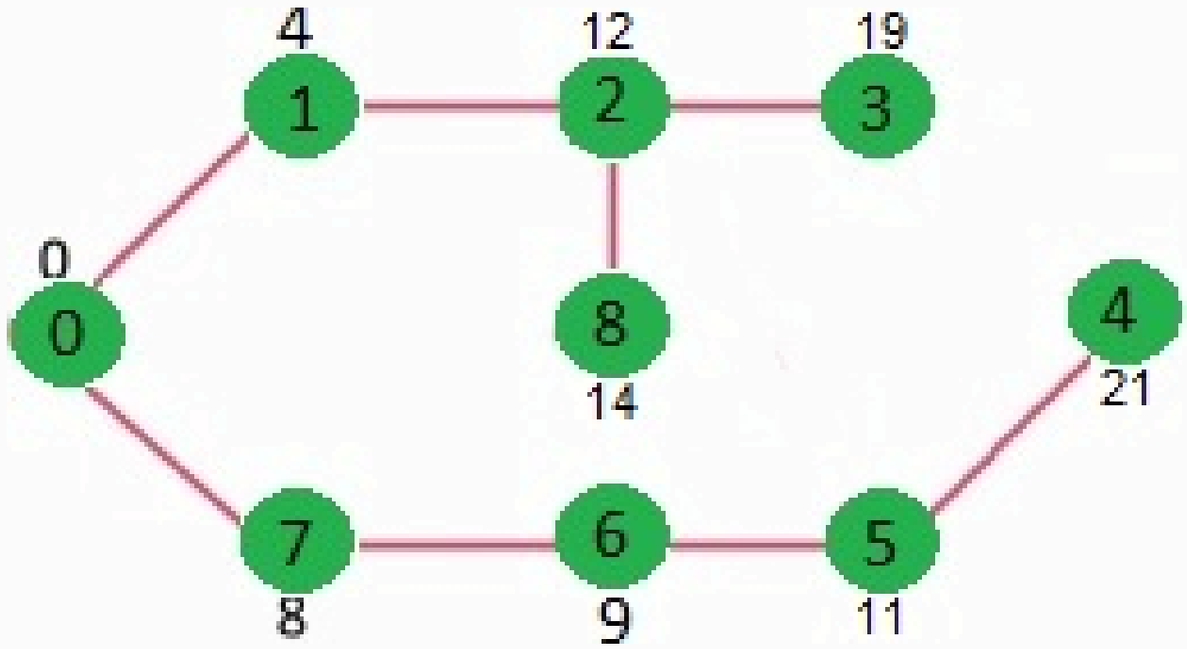
Exercice :



Problème du plus court chemin

ALGORITHME DE DIJKSTRA

Solution :



Problème du plus court chemin

ALGORITHME DE DIJKSTRA

? Comment reconstituer l'itinéraire d'un sommet *origine* vers un sommet *destination*

=> comme avec l'algorithme de Prim, on doit conserver une trace du *parent* de chaque sommet

! L'algorithme de Dijkstra ne fonctionne pas toujours si le graphe comporte des arêtes de poids *négatif*

(peut arriver dans certaines applications)