Graphes

Plus courts chemins

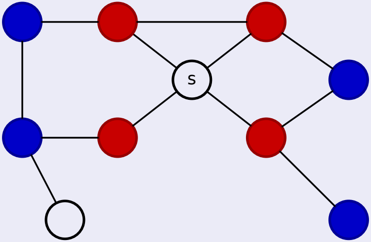
Mathilde Vernet
mathilde.vernet@univ-lehavre.fr

Master Informatique, Master Mathématiques
Université Le Havre Normandie

Automne 2020

UNIVERSITÉ LE HAVRE NORMANDIE

Qu'est-ce qu'un plus court chemin à partir de s ?



- Sommets à distance 1 de s
- Sommets à distance 2 de s

Comment parcourir le graphe ?

- En visitant tous les voisins d'abord

BFS s'impose !

Quelles applications ?

- Problèmes de tournées
- Routage
- Ordonnancement
- Programmation dynamique
- Problèmes d'investissement
- Problèmes de gestion de stocks
- ...

Contraintes

- Poids des arêtes unitaires
- Poids des arêtes positifs
- Poids des arêtes quelconques
- Graphe sans circuit
- Graphe quelconque
- ...

Quels types de problèmes ?

- Le plus court chemin entre deux sommets
- Le plus court chemin entre un sommet et tous les autres (*single-source shortest path problem*)
- Le plus court chemin entre toutes les paires de sommets (*all-pairs shortest path problem*)

Quels algorithmes de plus courts chemins ?

Parcours en largeur

- Plus court chemin entre un sommet et tous les autres
- Poids unitaires

Algorithme de Dijkstra

- Plus court chemin entre un sommet et tous les autres
- Poids positifs

Algorithme de Bellman-Ford

- Plus court chemin entre un sommet et tous les autres
- Poids quelconques

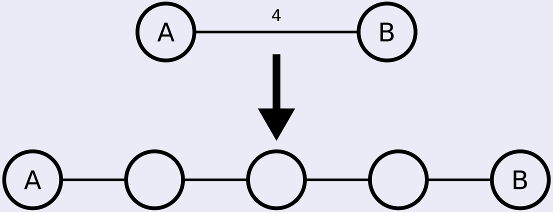
Algorithme de Floyd-Warshall

- Plus court chemin entre toutes les paires de sommets
- Poids quelconques

Avec BFS

On trouve des chemins pour des arêtes ayant des longueurs unitaires

Et avec des longueurs données sur les arêtes ?



Inconvénient

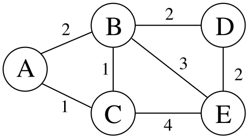
On ajoute beaucoup de sommets !

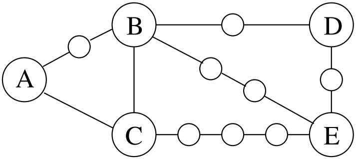
Problème à résoudre

- Entrée :
  - Graphe $G = (V, E)$ (orienté ou non)
  - Poids des arêtes (ou arcs) $w : E \to \mathbb{R}^+$
  - Sommet de départ $s \in V$
- Sortie :
  - Les plus courts chemins de s à tous les autres sommets de V

Adaptation du parcours en largeur

Chaque arête $e = (u, v) \in E$ est remplacée par w_e (le poids de e) arêtes de poids 1 en introduisant $w_e - 1$ sommets artificiels entre u et v.

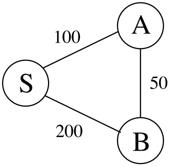


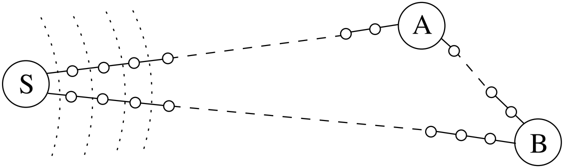


Images de *Dasgupta, Papadimitriou and Vazirani*

Comment peut-on voir cette adaptation ?

- Comme des alarmes : « On avance sur les arêtes jusqu'à ce qu'une alarme sonne »





Images de *Dasgupta*, *Papadimitriou and Vazirani*

Remarque

- Il ne s'agit nécessairement de distances euclidiennes :
  - Temps de trajet
  - Cout de déplacement
  - ...

Algorithme

Mettre l'alarme de s en temps 0
TantQue il reste des alarmes
  Dormir jusqu'à la prochaine alarme
  Soit t le temps et u le sommet du prochain déclenchement
  La distance entre s et u est t
  Pour $(u, v) \in E$ **Faire**
    Si v n'a pas d'alarme ou son alarme est plus tard que $t + w_{uv}$ **Alors**
      Mettre l'alarme de v en temps $t + w_{uv}$
    FinSi
  FinPour
FinTantQue

Comment efficacement implémenter les alarmes ?

Avec des files à priorités : structure de données permettant de stocker des éléments triés en fonction de leur « priorité ». Opérations :

- f.init() : crée un file vide
- f.empty() : vérifie si la file est vide
- f.extractMin() : récupère l'élément de priorité minimum
- f.add(e, p) : ajoute l'élément e de priorité p ou modifie sa priorité.

On a besoin de :

- Connaitre le plus courte distance à s pour chaque sommet
- Se souvenir du *parent* de chaque sommet

Procédure `Dijkstra(`G`,`s`)`
  v.dist $\leftarrow \infty$
  v.parent $\leftarrow$ null pour $v \in V$
  s.dist $\leftarrow 0$
  f.init()
  f.add(s, 0)
  TantQue non f.empty() **Faire**
    $u \leftarrow$ f.extractMin()
    Pour $(u, v) \in E$ **Faire**
      Si v.dist $>$ u.dist $+ w_{uv}$ **Alors**
        v.dist $\leftarrow$ u.dist $+ w_{uv}$
        v.parent $\leftarrow u$
        f.add(v, v.dist)
      FinSi
    FinPour
  FinTantQue
FinProcédure

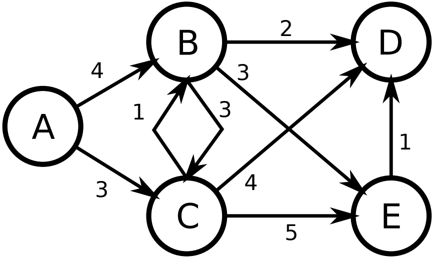
Dijkstra, c'est aussi :

- Maintenir
  - T : ensemble de sommets traités
  - v.dist, $v \in V$: longueur du plus court chemin de s à v qui ne passe que par des sommets de T
- À chaque itération
  - Choisir le sommet non-traité le plus proche de s
  - L'ajouter à T
  - Mettre à jour les attributs `dist` de ses voisins

Procédure `Dijkstra(`G`,`s`)`
    v.dist $\leftarrow \infty$ pour $v \in V$
    s.dist $\leftarrow 0$
    $T \leftarrow \emptyset$
    TantQue $T \neq V$ **Faire**
        $u \leftarrow \mathrm{argmin}\{v.\mathrm{dist} \ : \ v \in V \setminus T\}$
        $T \leftarrow T \cup \{u\}$
        Pour $(u, v) \in E$ **Faire**
            $v.\mathrm{dist} \leftarrow \min\{v.\mathrm{dist}, u.\mathrm{dist} + w_{uv}\}$
        FinPour
    FinTantQue
FinProcédure

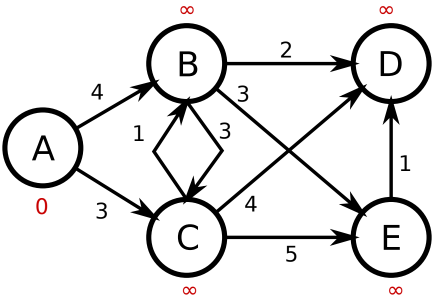
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



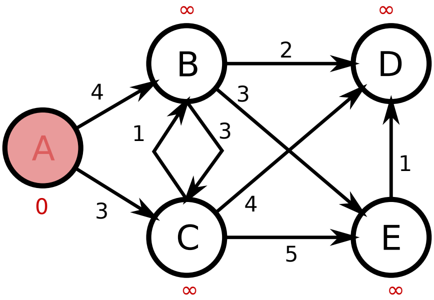
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



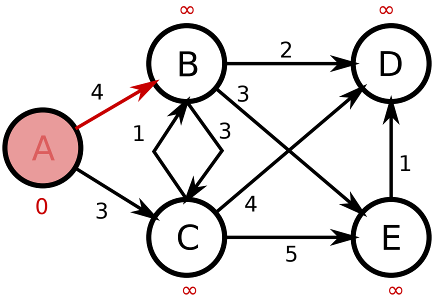
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



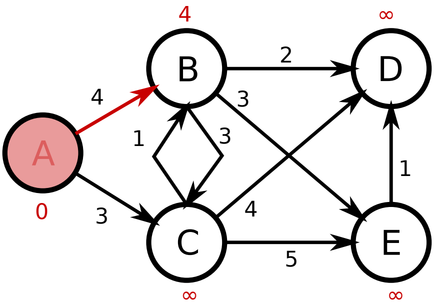
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



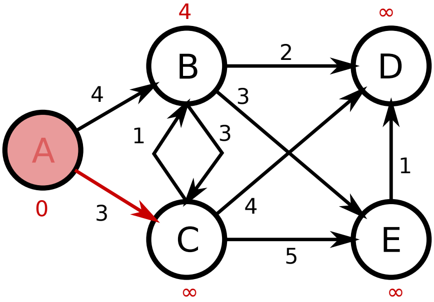
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



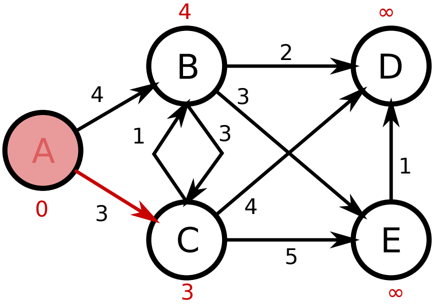
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



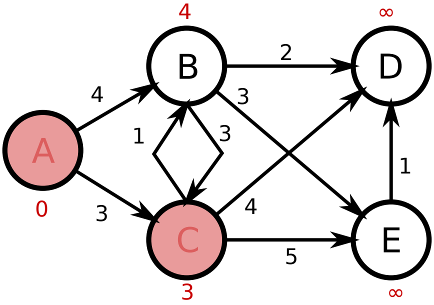
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



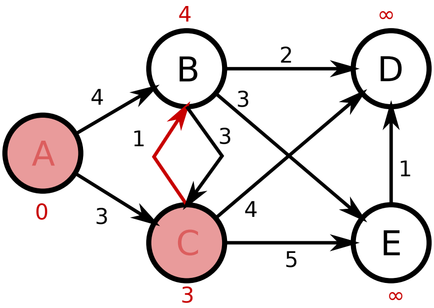
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



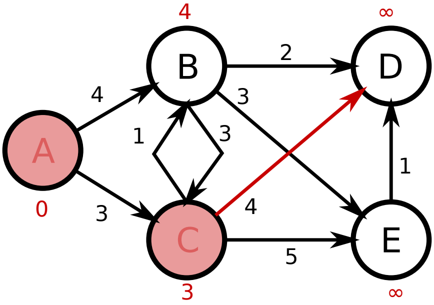
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



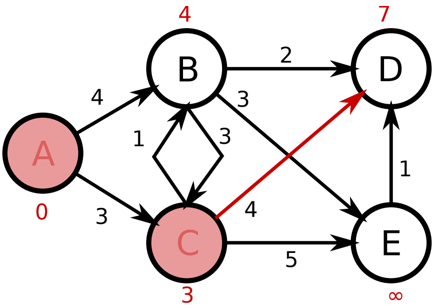
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



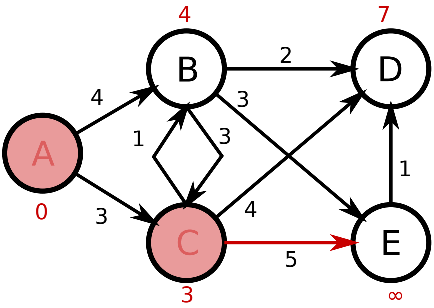
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



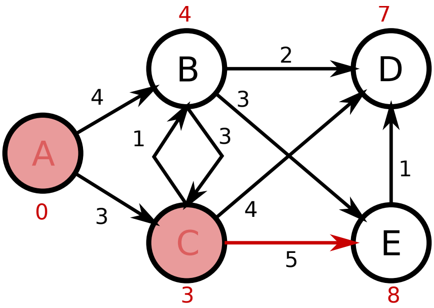
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



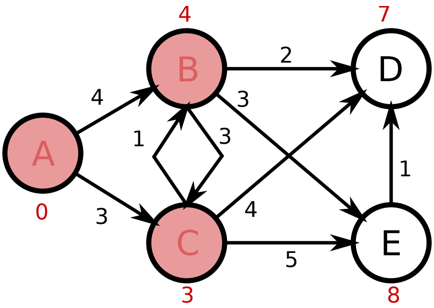
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



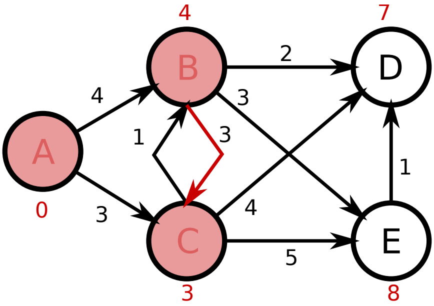
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



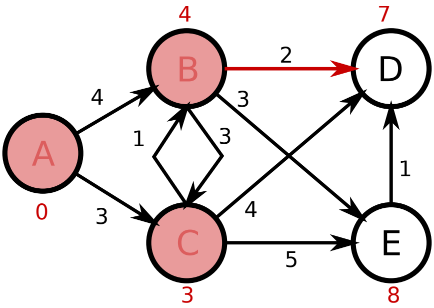
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



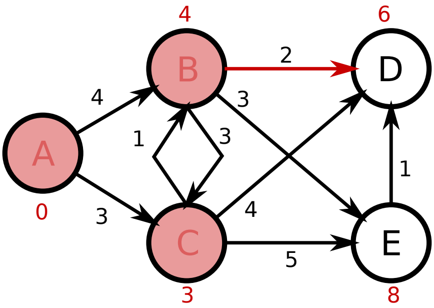
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



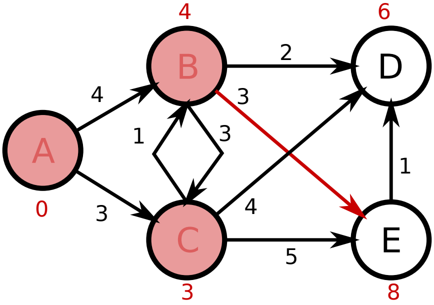
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



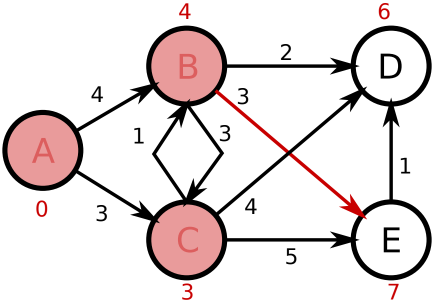
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



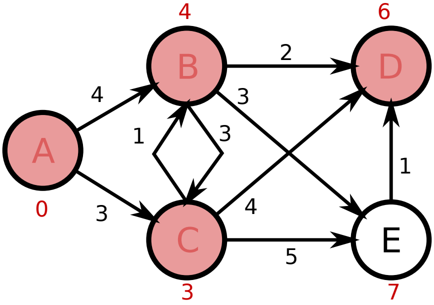
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



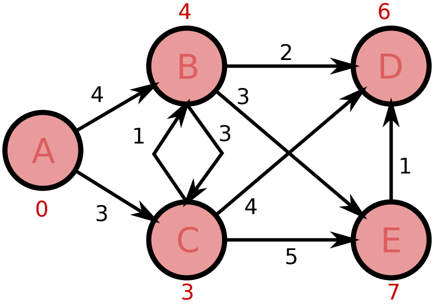
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



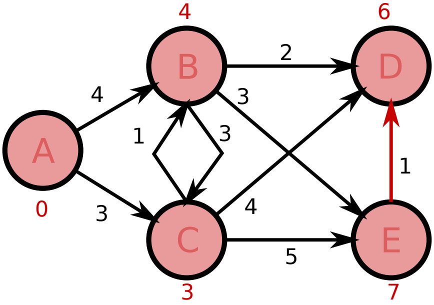
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



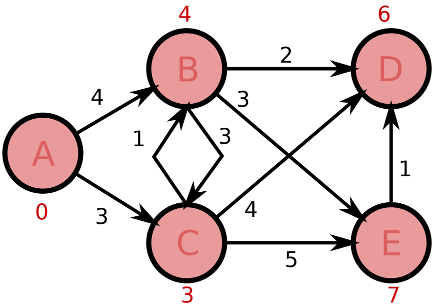
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



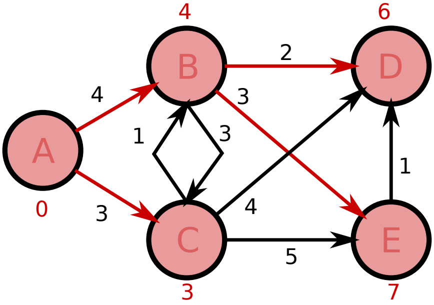
Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



Exercice

Appliquer l'algorithme de Dijkstra sur le graphe suivant à partir du sommet A.



Comment montrer que l'algorithme se termine et qu'il est correct ?

- Par récurrence sur la taille de l'ensemble T avec les hypothèses suivantes :
  - Pour $v \in T$, v.dist est la longueur du plus court chemin de s à v
  - Pour $v \notin T$, v.dist est la longueur du plus court chemin de s à v <u>ne passant que par sommets de T</u> pour arriver en v

Complexité

- Elle dépend de l'implémentation des files de priorités
- Sachant qu'on a :
  - $O(n)$ opérations extractMin()
  - $O(m)$ opérations add()

Implémentation naïve

Liste

- extractMin() : $O(n)$
- add() : $O(1)$
- Total : $O(n^2)$

Liste triée

- extractMin() : $O(1)$
- add() : $O(n)$
- Total : $O(nm)$

Structures performantes

Arbres binaires équilibrés et tas binaires

- extractMin() : $O(\log n)$
- add() : $O(\log n)$
- Total : $O((n+m)\log n)$

Tas de Fibonacci

- extractMin() : $O(\log n)$
- add() : $O(1)$ (amorti)
- Total : $O(n\log n + m)$

Exemple de comparaison

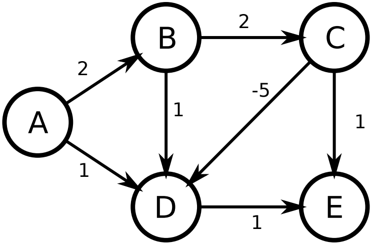
Dans un graphe où $n = 1000$ et $m = 10\ 000$

- Implémentation naïve ($O(n^2)$) : $\approx$ 1 000 000 opérations
- Implémentation efficace ($O(m + n\log n)$) : $\approx$ 20 000 opérations
- Accélération : $\approx \times 50$

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

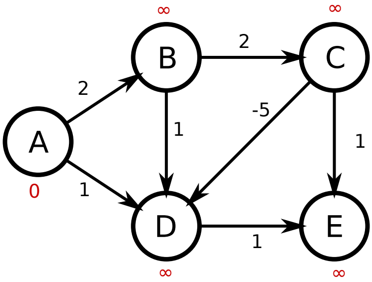


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

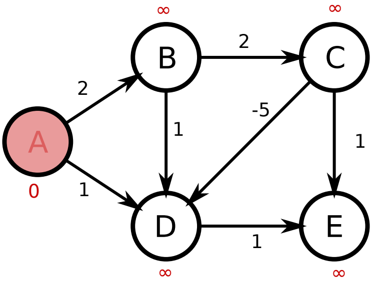


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

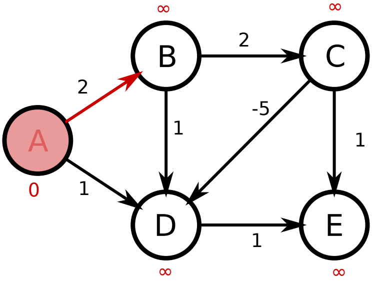


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

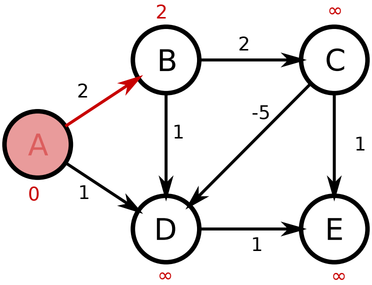


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

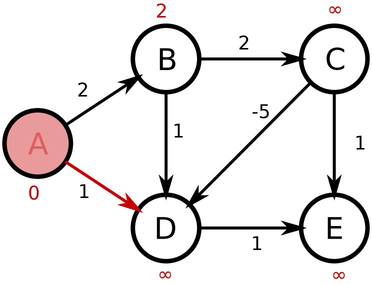


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

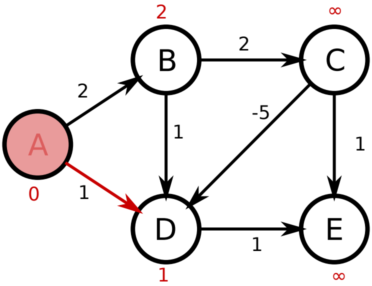


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

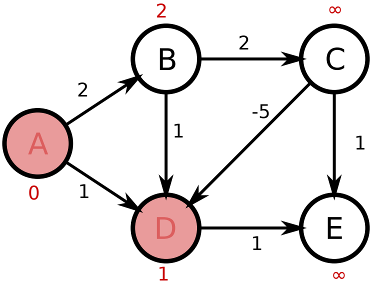


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

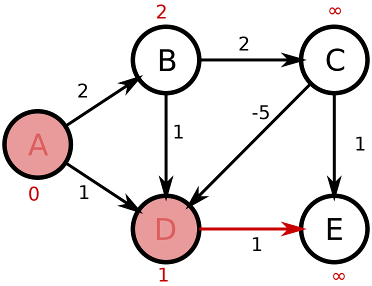


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

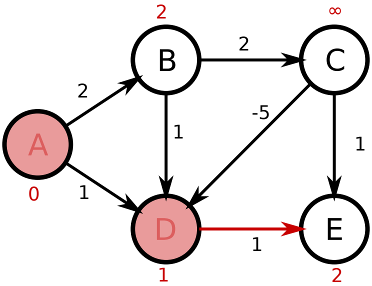


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

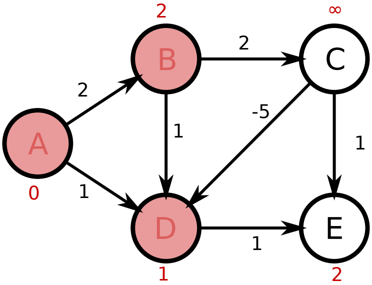


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

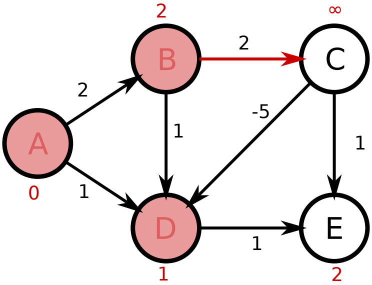


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

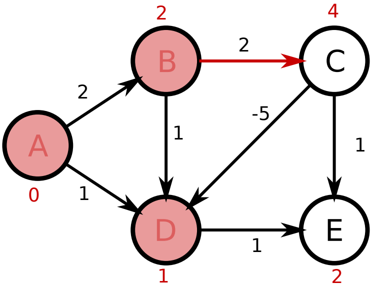


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

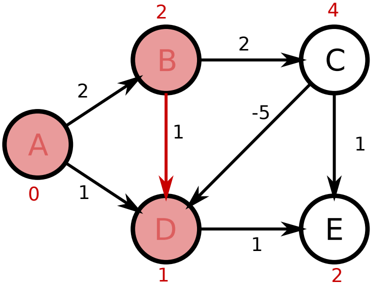


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

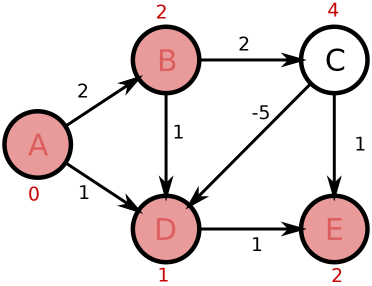


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

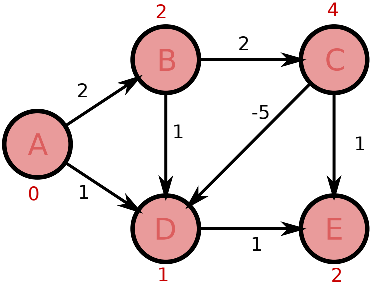


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

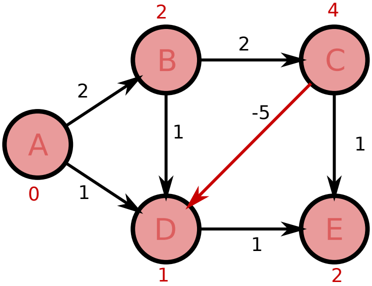


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

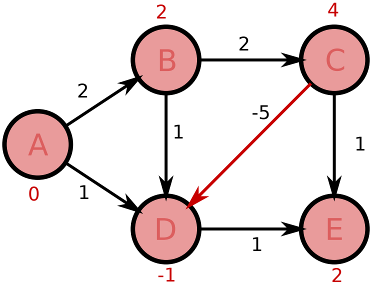


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

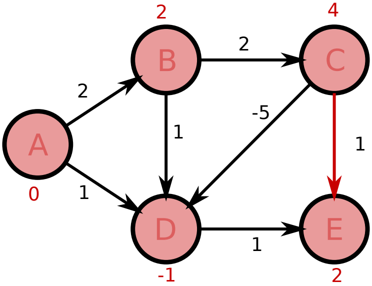


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

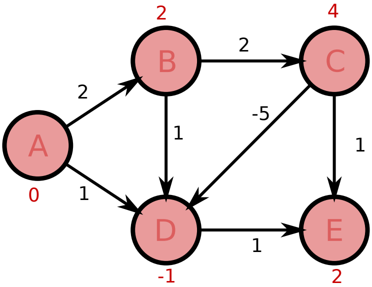


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Question

- L'algorithme de Dijkstra ne fonctionne pas dans le cas de poids négatifs. Donner un exemple qui montre ceci.

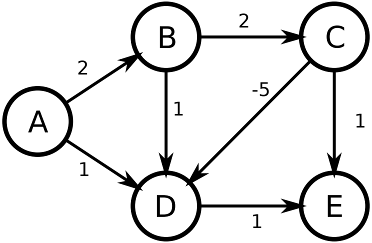


Problème : La distance de A à E est 0 et non pas 2 !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

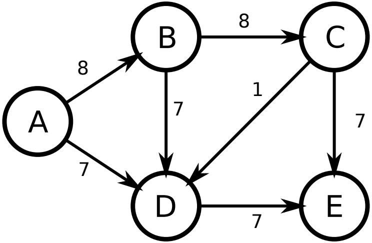


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

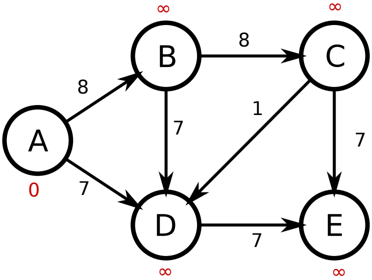


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

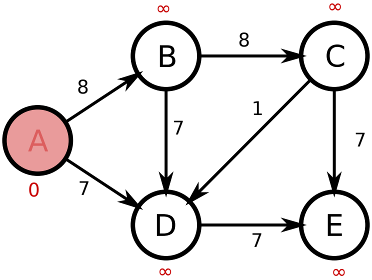


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

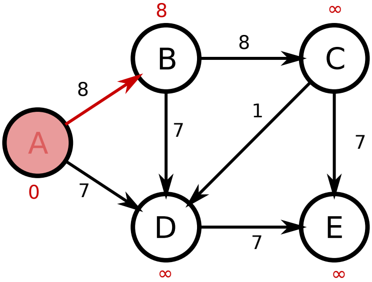


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

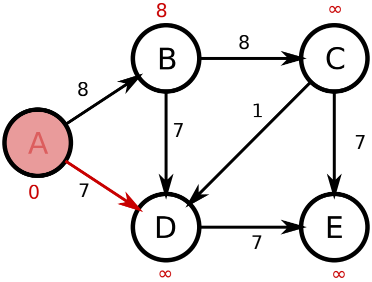


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

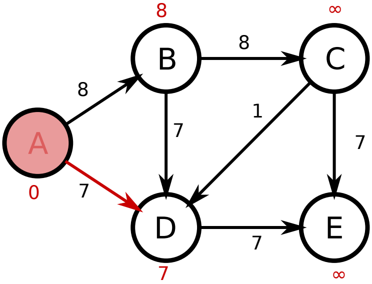


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

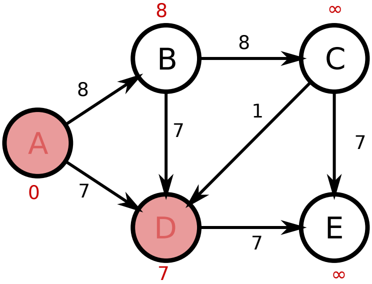


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

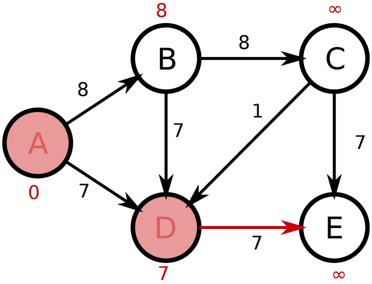


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

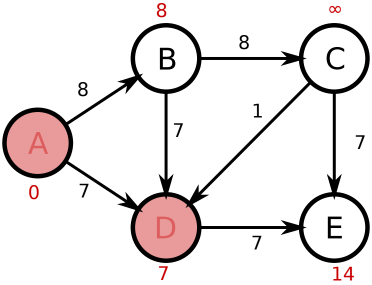


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

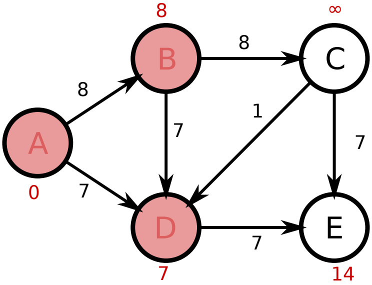


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

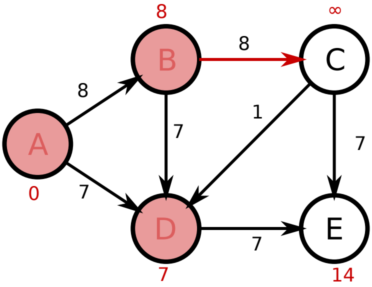


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

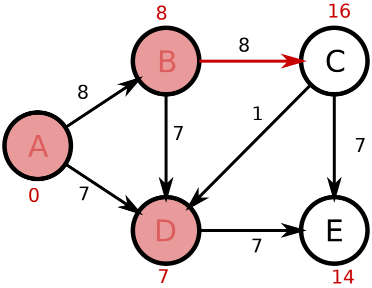


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

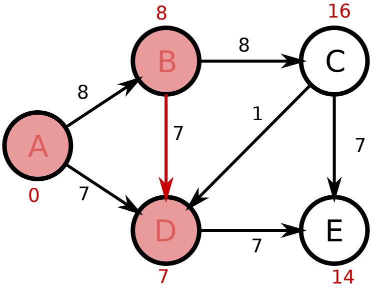


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

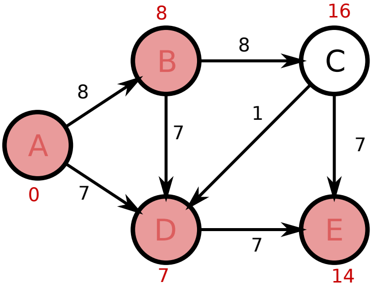


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

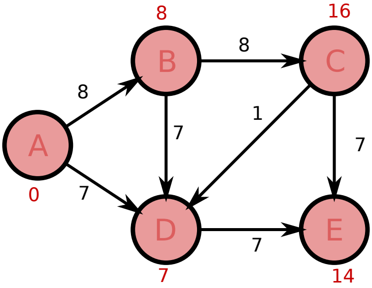


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

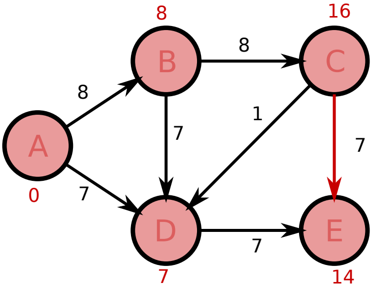


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

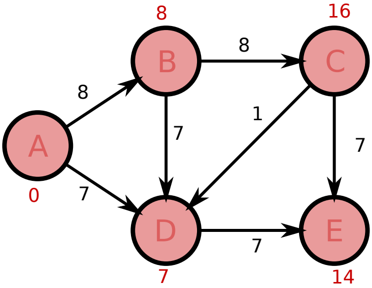


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

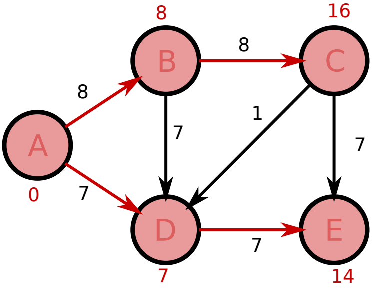


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.

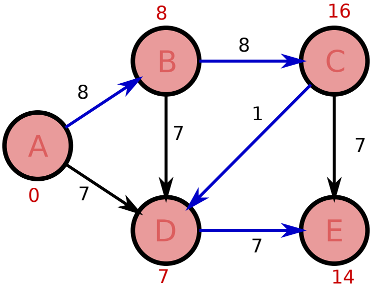


Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Exercice

Questions

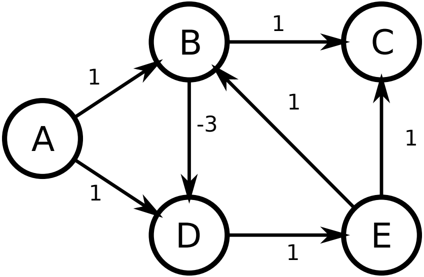
1. Comment faire pour gérer les poids négatifs ?
2. On propose l'algorithme suivant dans le cas de poids négatifs : On ajoute une grande constante aux poids des arcs de façon à ce que ceux-ci deviennent positifs. Ensuite on applique l'algorithme de Dijkstra. Cette méthode est-elle correcte ? Si oui, montrez-le, sinon, donnez un contre-exemple.



Problème : Le plus court chemin de A à E n'est pas le même dans ce graphe que dans le graphe initial !

Question

- Quel est le plus court chemin de A à E dans ce graphe ?



Si le graphe contient un cycle de poids négatif, un plus court chemin n'a pas de sens !

Remarques

- L'algorithme de Dijkstra ne fonctionne pas si des arcs ont des poids négatifs
- On ne peut pas adapter Dijkstra pour prendre en compte des poids négatifs en augmentant tous les poids
- De façon général, s'il y a un cycle de poids négatif dans un graphe, le plus court chemin peut ne pas exister

Le cas des poids négatifs

- On sait que Dijkstra est inutilisable pour les poids négatifs
- Comment faire ?

Procédure de mise à jour de la valeur de distance de Dijkstra

Procedure $\texttt{maj}((u, v) \in E)$
$v.\text{dist} \leftarrow \min\{v.\text{dist}, u.\text{dist} + w_{uv}\}$
FinProcedure

- Donne la valeur correcte de v.dist si u est le sommet avant v dans le plus court chemin de s à v et si la valeur de u.dist est correcte
- Quel que soit le nombre d'appels, v.dist reste une borne supérieure sur la distance entre s et v

Procedure $\texttt{maj}((u,v) \in E)$
v.dist $\leftarrow \min\{v.\text{dist}, u.\text{dist} + w_{uv}\}$
FinProcedure

Comment utiliser cette procédure ?

Considérons le plus court chemin entre s et v :

$$s - u_1 - u_2 - \cdots - u_k - v$$

- Si la séquence des mises à jour inclut $(s,u_1),(u_1,u_2),\ldots(u_k,v)$ dans cet ordre (mais pas forcement consécutivement), la distance de s à v sera correctement calculée.
- Comment faire sans connaitre l'ordre ? Mettre à jour tous les arcs $n-1$ fois.

Procedure $\texttt{BellmanFord}(G, s)$
  v.dist $\leftarrow \infty$ pour $v \in V$
  s.dist $\leftarrow 0$
  Répéter $n - 1$ fois
    Pour $e \in E$ **Faire**
      maj(e)
    FinPour
  FinRépéter
FinProcedure

Procedure $\texttt{maj}(e = (u, v))$
  v.dist $\leftarrow \min\{v.\text{dist}, u.\text{dist} + w_{uv}\}$
FinProcedure

Complexité

- $O(mn)$
  - Tous les arcs sont parcourus $n - 1$ fois

Quelle amélioration possible ?

- S'arrêter si pendant une itération il n'y a pas eu de mise à jour

Question

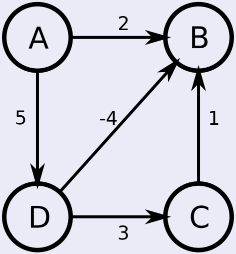
Comment peut-on adapter l'algorithme de Bellman-Ford pour détecter des cycles de poids négatif ?

Idée

- On sait que s'il n'y a pas eu de changements d'une itération à la suivante, cela signifie qu'on a terminé

En effectuant une $n^{\text{ième}}$ itération, et s'il y a encore des changements dans les valeurs de distance, alors il y a un cycle négatif

Quelle est la particularité de ce graphe ?



Observations

- C'est un DAG
- Donc tous les arcs vont « en avant »

Comment utiliser cette particularité ?

- On peut faire l'opération de mise à jour une seule fois pour chaque arcs
- On visite les sommets dans leur ordre topologique, en mettant à jour leurs arcs sortants

Procedure `PlusCourtCheminDAG`(G, s)
  v.dist $\leftarrow \infty$ pour $v \in V$
  s.dist $\leftarrow 0$
  trier les sommets de G en ordre topologique
  Pour $u \in V$ en ordre topologique **Faire**
    Pour $(u, v) \in E$ **Faire**
      maj$((u, v))$
    FinPour
  FinPour
FinProcedure

Procedure `maj`$(e = (u, v))$
  v.dist $\leftarrow \min\{v.\text{dist}, u.\text{dist} + w_{uv}\}$
FinProcedure

Complexité

- Si les sommets sont déjà triés : $O(m)$
  - Tous les arcs sont parcourus une fois

Pourquoi, en général, on n'utilise pas cet algorithme ?

« Le graphe est un DAG » est une hypothèse trop forte qu'on ne peut, en général, pas faire

Comment trouver les plus courts chemins entre toutes les paires de sommets ?

- Utilisation d'un tableau où la case i, j contient la longueur du plus court chemin de i à j
- En utilisant le principe de la programmation dynamique
- L'idée est de calculer les plus courts chemins passant seulement par les k premiers sommets et d'augmenter k petit à petit

Principe

- d_{ij}^k : longueur du plus court chemin de i à j passant uniquement par les sommets $1, 2, \ldots, k$

Initialisation

$$d_{ij}^0 = \begin{cases} w_{ij} & \text{si } (i,j) \in E \\ 0 & \text{si } i = j \\ \infty & \text{sinon} \end{cases}$$

Relation de récurrence

d_{ij}^k (longueur du plus court chemin de i à j passant uniquement par les sommets $1, 2, \ldots, k$) vaut :

- d_{ij}^{k-1} : Le plus court chemin de i à j passant seulement par les sommets $1, 2, \ldots, k-1$
- $d_{ik}^{k-1} + d_{kj}^{k-1}$: Le plus court chemin de i à j passant forcément par le sommet k

Algorithme de Floyd-Warshall

Procedure `FloydWarshall`(G)
  $d_{ij} = \infty \; \forall \; i \in V, j \in V$
  $d_{ii} = 0 \; \forall \; i \in V$
  $d_{ij} = w_{ij} \; \forall \; (i,j) \in E$
  Pour k de 1 à n **Faire**
    Pour i de 1 à n **Faire**
      Pour j de 1 à n **Faire**
        $d_{ij} = \min\{d_{ij}, d_{ik} + d_{kj}\}$
      FinPour
    FinPour
  FinPour
FinProcedure

Complexité

- $O(n^3)$
  - Une opération en temps constant
  - Dans une boucle n fois
  - ...elle même dans une boucle n fois
  - ...elle même dans une boucle n fois

Comment trouver tous les plus courts chemins entre chaque paire de sommet ?

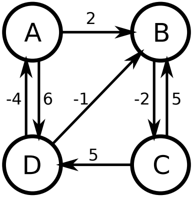
- Avec Floyd-Warshall
- En utilisant un algorithme de plus court chemin d'un sommet s vers tous les autres sommets du graphe. Comment ? En appelant cet algorithme à partir de chaque sommet du graphe

Comparaison de complexité

Algorithme	*Single source shortest path*	*All-pairs shortest path*
Dijkstra	$m + n \cdot \log(n)$	$n \cdot m + n^2 \cdot \log(n)$
Bellman-Ford	$n \cdot m$	$n^2 \cdot m$
Floyd-Warshall	n^3	n^3

Question

Appliquer l'algorithme de Floyd-Warshall sur le graphe suivant



0	A	B	C	D
A	0	2	∞	6
B	∞	0	-2	∞
C	∞	5	0	5
D	-4	-1	∞	0

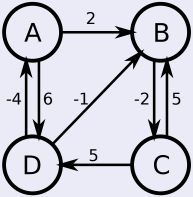
1	A	B	C	D
A	0	2	∞	6
B	∞	0	-2	∞
C	∞	5	0	5
D	-4	**-2**	∞	0

2	A	B	C	D
A	0	2	**0**	6
B	∞	0	-2	∞
C	∞	5	0	5
D	-4	-2	**-4**	0

3	A	B	C	D
A	0	2	0	**5**
B	∞	0	-2	**3**
C	∞	5	0	5
D	-4	-2	-4	0

4	A	B	C	D
A	0	2	0	5
B	**-1**	0	-2	3
C	**1**	**3**	0	5
D	-4	-2	-4	0

Exemple



	A	B	C	D
A	0	2	0	5
B	-1	0	-2	3
C	1	3	0	5
D	-4	-2	-4	0

Que remarque-t-on ?

- Il y a une diagonale de 0

Comment utiliser cette information pour détecter des cycles négatifs ?

- Si $\exists\ i, d_{ii} < 0$, alors cela signifie qu'il existe un cycle négatif

Comment trouver le chemin lui-même en plus de la valeur ?

P_{ij}^k : prédécesseur de j dans le plus court chemin de i à j passant par les sommets $1, \ldots k$

- Initialisation : prédécesseur de j dans le plus court chemin de i à j passant par aucun sommets :

$$P_{ij}^0 = \begin{cases} i & \text{si } (i,j) \in E \\ NULL & \text{si } i = j \\ NULL & \text{sinon} \end{cases}$$

- Récurrence : prédécesseur de j dans le plus court chemin de i à j passant par les sommets $1, \ldots k$. $d_{ij}^k =$
  - P_{ij}^{k-1} : prédécesseur de j dans le plus court chemin de i à j passant par les sommets $1, \ldots k-1$

  OU

  - P_{kj}^{k-1} : prédécesseur de j dans le plus court chemin de i à j passant forcément par le sommet k

Algo pour le *single-source shorstest path problem*

- Le plus efficace sur les DAG : $O(m)$
- Le plus efficace hors DAG : Dijkstra, $O(m + n \cdot \log(n))$, uniquement des poids positifs
- Le plus générique : Bellman-Ford, $O(m \cdot n)$, même avec des poids négatifs

Algo pour le *all-pairs shorstest path problem*

- Floyd-Warshall : $O(n^3)$

Quoi utiliser, quand ?

1. Je cherche les plus courts chemins depuis un sommet donné ?
   1. Le graphe est-il sans cycle ? Cas particulier de Bellman-Ford en $O(m)$
   2. Le graphe a-t-il seulement des poids positifs ? Dijkstra en $O(m + n \cdot \log(n))$
   3. Le graphe a-t-il des poids négatifs ? Bellman-Ford en $O(m \cdot n)$
2. Je cherche les plus courts chemins entre chaque paire de sommets ?
   Floyd-Warshall ou n fois un algo de *single-source shorstest path problem* selon les mêmes interrogations que précédemment