INFO601 : algorithmique et graphes
TD 3 : plus court chemin dans un graphe pondéré

Pierre Hyvernat, Gérald Cavallini
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 17, poste : 94 22
email : Pierre.Hyvernat@univ-smb.fr
www : http://www.lama.univ-smb.fr/~hyvernat/

Exercice 1 : algorithme de Dijkstra

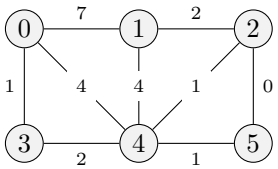
Question 1. Donnez un exemple de graphe pondéré où le chemin trouvé par le parcours en largeur depuis $s_0 = 0$ jusqu'à $s_1 = 1$ n'a pas le poids minimal. (Le chemin trouvé sera par contre forcément minimal en nombre d'arêtes utilisées.)

Question 2. L'algorithme de Dijkstra cherche un chemin de poids total minimal en modifiant le parcours basique.

1- En plus du tableau `Pred` pour sauvegarder les chemins trouvés depuis s_0, l'algorithme sauvegarde également le poids du chemin trouvé dans un tableau `D`.
2- Au début de la boucle, on choisit le sommet s dans `ATraiter` qui a la petite valeur de `D[`s`]`.
3- Lorsqu'on regarde les voisins v de s, on mets à jour les tableau `Pred` et `D` seulement si la nouvelle distance trouvée est plus courte :

```
for (v, d) in G[s]:      # d est le poids de l'arête s -> v
    if not Vu[v] and d + D[s] < D[v]:
        ...                  # on ajoute v dans ATraiter
        Pred[v] = s
        D[v] = d + D[s]
```

Faites tourner l'algorithme à la main sur le graphe suivant, en partant du sommet $s_0 = 0$.



Quelle est la distance trouvée pour aller de $s_0 = 0$ jusqu'au sommet 1 ?

Vérifiez que vous pouvez retrouver le chemin de $s_0 = 0$ jusqu'au sommet 1 à partir du tableau `Pred`.

Question 3. Quelle est la complexité de la recherche du sommet `s` avec `D[s]` minimal dans le tableau `ATraiter` ?

Comment peut on améliorer la complexité globale de l'algorithme en évitant de faire une recherche "naïve" en parcourant le tableau `ATraiter` ?

Question 4. Expliquez pourquoi le tableau `Vu` n'est pas nécessaire dans l'algorithme de Dijkstra.

Question 5. L'algorithme de Dijkstra ne donne pas toujours une bonne solution si le graphe contient des arêtes de poids négatif.

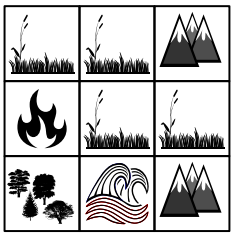
Donnez un contre-exemple.

Donnez également un exemple de graphe avec poids négatif où
- il y a un chemin de 0 à 1
- il n'y a pas de chemin de 0 à 1 *de poids minimal.*

Exercice 2 : algorithme A*

Lorsqu'on cherche un plus court chemin de s_0 à s_1 (par opposition à *tous* les plus courts chemin à partir de s_0), on peut essayer de "diriger" la recherche en donnant priorité aux sommets qui nous "rapprochent" de s_1.

Question 1. On considère un plateau de jeu avec la numérotation des cases données à droite



(0,0,)	(1,0)	(2,0)
(0,1)	(1,1)	(2,1)
(0,2)	(1,2)	(2,2)

Le coup de déplacement d'une case vers une des cases adjacentes est de :
- 1 pour sortir d'une case "herbe",
- 2 pour sortir d'une case "forêt",
- 3 pour sortir d'une case "montagne",
- 4 pour sortir d'une case "eau",
- 8 pour sortir d'une case "feu",
- on ne peut pas se déplacer en diagonale.

Donnez le graphe des déplacements possibles correspondant à ce plateau.

Question 2. Si la grille ne contenait que des cases "herbe", comment pourrait-on calculer la distance minimale $dist((x,y),(x',y'))$ pour aller d'une case de coordonnées (x,y) à une case de coordonnées (x',y').

Question 3. À chaque pas de l'algorithme de Dijkstra, `D[`s`]` contient la distance minimale réelle depuis s_0 connue à cet instant.

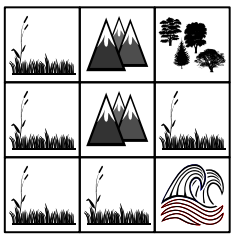
Que représente le nombre, `D[`s`]` $+ dist(s,s_1)$, où s_1 est le sommet à atteindre ?

Question 4. L'algorithme A* choisit le sommet s de `ATraiter` qui minimise `D[`s`]` $+ dist(s,s_1)$.
- Faites tourner l'algoritme à la main sur le graphe précédent, pour aller du sommet $(1,0)$ au sommet $(2,2)$.
- Modifiez le code de l'algorithme de Dijkstra pour faire ceci.

Question 5. Donnez des exemples de graphes (avec plus que 9 sommets !) où A* sera moins efficace que l'algorithme de Dijksta.

Question 6. Peut on faire la même chose sur un graphe arbiraire ?

Question 7. On considère maintenant le plateau suivant



avec des coups de déplacement diminués de 1. Autrement dit :
- 0 pour sortir d'une case "herbe",
- 1 pour sortir d'une case "forêt",
- etc.

Faites tourner l'algorithme A* pour chercher un chemin de $(1,0)$ à $(2,2)$. Qu'en pensez vous ?

Donnez une explication.