université de BORDEAUX

Année 2023-2024 — DS novembre 2023
Épreuve de : Recherche Opérationnelle
Date : 24 novembre 2023 — **Durée :** 1h30
Épreuve de : O. Baudon — **Nb. pages : 7**

Une attention toute particulière sera portée à la présentation et à la rédaction de la copie.

Dans tous les exercices, on désignera par $V(G)$ et $E(G)$ respectivement l'ensemble des sommets et l'ensemble des arêtes d'un graphe G. Les variables n et m désigneront respectivement le nombre de sommets et d'arêtes.

Dans tout le sujet, on conviendra que, dans les différentes structures de données modélisant les graphes, les sommets sont rangés dans l'ordre croissant de leur nom.

Les algorithmes utilisés dans les exercices sont disponibles à la fin du sujet.

1 Structures de données et complexité

Dans un graphe simple orienté G, on appelle ***puits universel*** tout sommet v tel que :
— pour tout sommet $u \neq v$, il existe un arc uv ;
— v n'a pas de successeur.

1.1) Dessiner un graphe à 4 sommets possédant un puits universel. Quel est le nombre maximum de puits universels que peut contenir un graphe ? (Justifier votre réponse).

1.2) Pour chacune des deux représentations :

1. listes de successeurs,
2. matrice d'adjacence,

écrire un algorithme pour déterminer si le graphe contient un puits universel et en étudier la complexité. *Attention de présenter les algorithmes de façon lisible !*

Solution Q1
Le graphe à 4 sommets avec un puits universel est dessiné ci-dessous dans la Figure 1. Le puits est le sommet étiqueté *puits*.
Un graphe ne peut contenir qu'au plus un puits universel. En effet, tous les autres sommets doivent avoir ce sommet comme successeur et un puits universel n'a pas de successeur.

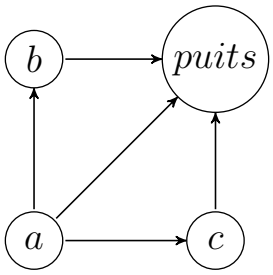

FIGURE 1 – Graphe avec un puits

Solution Q2

La fonction `Puits` renvoie `Faux` si le graphe ne possède pas de puits universel, ou le sommet puits sinon.

Pour les deux représentations, l'algorithme consiste à sélectionner un sommet candidat n'ayant aucun successeur, puis à vérifier que tous les autres sommets sont bien des prédécesseurs de ce candidat.

Version avec liste de successeurs (G est un graphe représenté par ses listes d'adjacence) :

```
0   Puits(G) :
1   candidat <- Nil
2   pour tout sommet v de V(G) faire
3           si Adj(v) = ∅ alors
4                   candidat <- v
5                   break
6   si candidat == Nil alors
7           retourner Faux
8   pour tout sommet v de V(G) faire
9           si v != candidat alors
10                  pred <- Faux
11                  pour tout sommet u de Adj(v) faire
12                          si u = candidat alors
13                                  pred <- Vrai
14                                  break
15                  si pred = Faux alors
16                          retourner Faux
17  retourner candidat
```

Version avec matrice d'adjacence (A est une matrice d'adjacence de taille $n \times n$) :

```
0   Puits(A) :
1   pour i de 1 à n faire
2           candidat <- i
3           pour j de 1 à n faire
4                   si A[i, j] != 0
5                           candidat <- -1
6                           break
7           si candidat != -1 alors
8                   break
9   si candidat = -1 alors
10          retourner Faux
11  pour i de 1 à n faire
12          si i != candidat et A[i, candidat] = 0 alors
13                  retourner Faux
14  retourner candidat
```

La complexité de la version avec les listes d'adjacences est en $O(n+m)$. En effet la boucle 2-5 est en $O(n)$, les boucles 8-16 et 11-16 combinées sont en $O(n+m)$.
La complexité de la version avec la matrice d'adjacence est en $O(n^2)$. En effet la boucle 1-8 est en $O(n^2)$, la boucle 11-13 est en $O(n)$.

2 Flot Maximum

2.1) En utilisant l'algorithme de Ford-Fulkerson rappelé ci-dessous (Algorithme **??**), trouver le flot maximum entre les sommets s et t du réseau de la figure 2. Le flot possède déjà une valeur initiale donnée par le premier nombre porté sur les arcs, le second étant la capacité de l'arc. Par exemple, l'arc `(s,A)` possède un flot initial de 2 et une capacité de 4. On donnera pour chaque exécution de la procédure de marquage le chemin augmentant obtenu et l'augmentation correspondante.

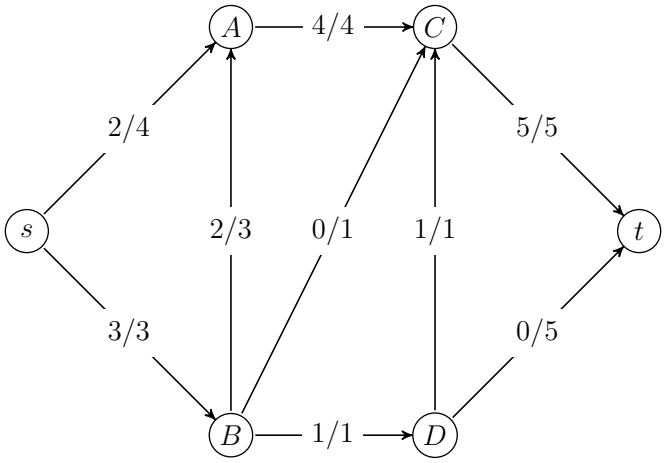

FIGURE 2 – Réseau

Solution Q1

Il n'y a qu'un seul chemin augmentant, la valeur de l'augmentation étant $+1$: $sABCDt$. On notera que l'utilisation des arcs AB et CD se font par marquage indirect. La valeur du flot maximum est donc de 6

2.2) Donner la coupe minimum obtenue par la dernière exécution de la procédure de marquage, sa valeur et redessiner le graphe avec le flot maximum (en précisant sur un arc retour sa valeur).

Solution Q2

Les sommets marqués lors de la dernière exécution la procédure de marquage sont l'ensemble $Y = \{s, A, B\}$. La valeur de cette coupe est $c(AC)+c(BC)+c(BD) = 4+1+1 = 6$, ce qui correspond bien à la valeur du flot maximum trouvé. Le fot maximum est donné par la Figure 3 ci-dessous.

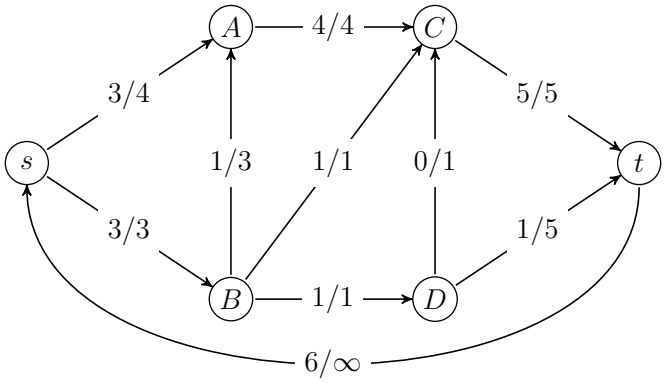

FIGURE 3 – Flot maximum

3 Plus courts chemins

Un étudiant du Master Miage de l'Université de Bordeaux, désirant faire un séjour linguistique, décide de se rendre en Suède. Après avoir fait le tour de quelques compagnies, il a recensé plusieurs connexions aériennes lui permettant d'aller de Bordeaux à Stockholm. Il les a représentées à l'aide du graphe suivant (Figure 4) :

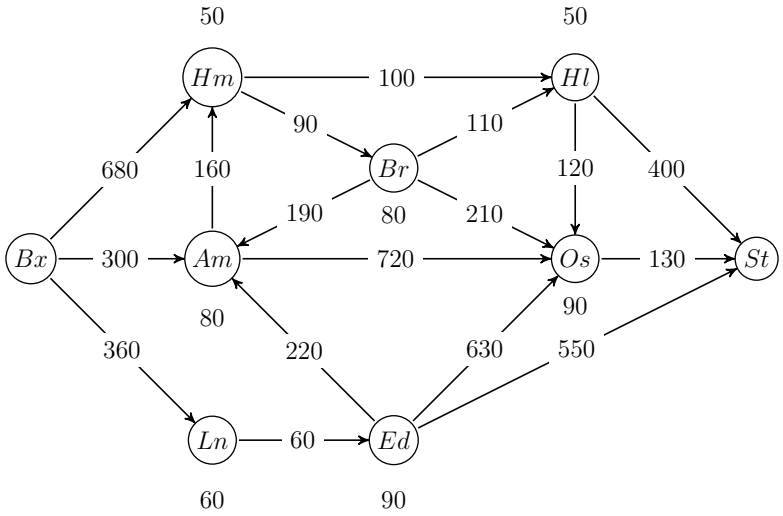

FIGURE 4 – Voyage

où Bx = Bordeaux, Hm = Hambourg, Am = Amsterdam, Ln = Londres, Ed = Edimbourg, Br = Berlin, Hl = Helsinki, Os = Oslo, St = Stockholm.

Cependant, les horaires des vols sont tels que l'étudiant est obligé de passer la nuit dans chacune des villes où il fait escale.

Les valeurs sur les arcs correspondent aux prix en euros pour les vols, et les valeurs à coté des sommets représentent le prix en euros à payer pour passer la nuit dans un hôtel de la ville correspondante.

3.1) On désire trouver une solution optimale pour ce problème. Quelles modifications faut-il appliquer au graphe pour pouvoir utiliser l'algorithme de Dijkstra ?

3.2) Calculer la solution optimale. On donnera en particulier la suite des valeurs prises par la variable `pivot` et pour chacune de ces valeurs, les modifications des valeurs du tableau `d` associées.

Solution Q1

Il existe plusieurs possibilités :

- on peut remplacer chaque sommet s ayant un coût d'hôtel par deux sommets s_1 et s_2 reliés par un arc avec comme poids le prix de l'hôtel. Les arcs entrants sur le sommet original s seront entrants sur le sommet s_1 et les arcs sortants du sommet s seront sortants du sommet s_2.
  Le graphe obtenu est décrit dans la Figure 5 ci-dessous.
- Une autre solution est, pour chaque sommet s représentant une ville avec un coût d'hôtel, de rajouter le coût de l'hôtel au poids de chaque arc sortant de s.
  Le graphe obtenu est décrit dans la Figure 6 ci-dessous.
- On peut aussi pour chaque sommet s rajouter le coût de l'hôtel au poids de chaque arc entrant de s.

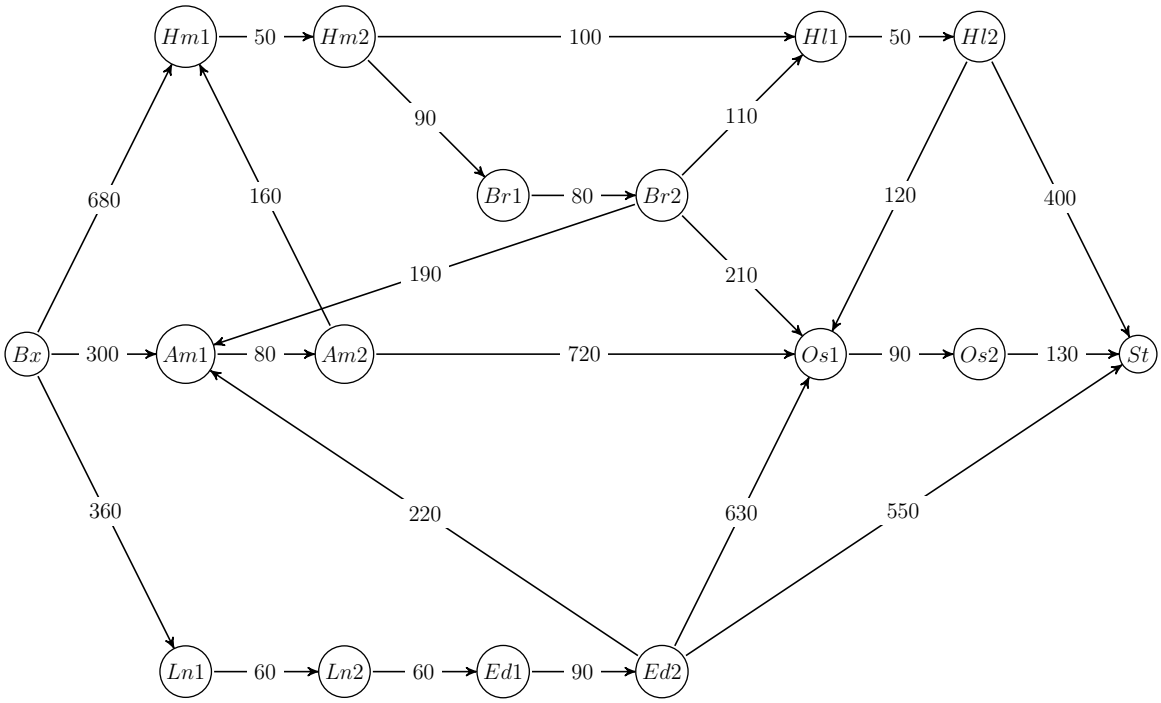

FIGURE 5 – Voyage modifié - version 1

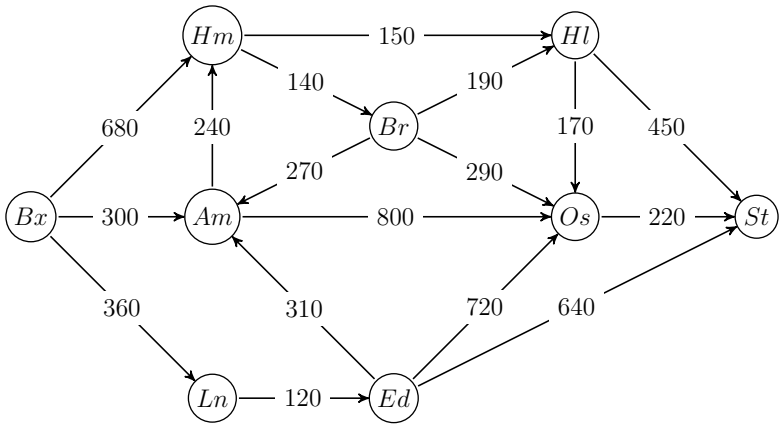

FIGURE 6 – Voyage modifié - version 2

Solution Q2

On va utiliser Dijkstra sur une des deux modifications proposées. Le tableau ci-dessous utilise la 2ème version.

Pivot	*Bx*	*Hm*	*Am*	*Ln*	*Br*	*Ed*	*Hl*	*Os*	*St*
	0	∞	∞	∞	∞	∞	∞	∞	∞
Bx/0	*X*	680	300	360					
Am/300	*X*	540	*X*					1100	
Ln/360	*X*		*X*	*X*		480			
Ed/480	*X*		*X*	*X*		*X*			1120
Hm/540	*X*	*X*	*X*	*X*	680	*X*	690		
Br/680	*X*	*X*	*X*	*X*	*X*	*X*		970	
Hl/690	*X*	*X*	*X*	*X*	*X*	*X*	*X*	860	
Os/860	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	1080
St/1080	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*

Ce qui nous donne l'arborescence des plus courts chemins de la Figure 7.

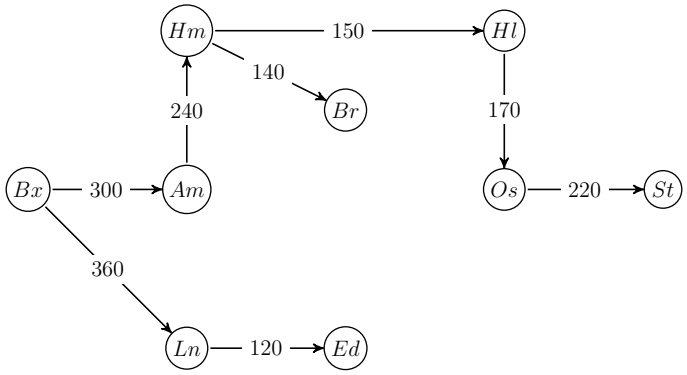

FIGURE 7 – Voyages les moins chers à partir de Bordeaux

Le chemin le moins cher pour relier Bordeaux à Stockholm est donc : Bordeaux, Amsterdam, Hambourg, Helsinki, Oslo, Stockholm pour un coût total de 1080 euros.

4 Arbre de poids maximum

On considère le problème suivant :

un réseau routier reliant un ensemble de villes est représenté par un graphe G simple, non orienté, chaque ville étant représentée par un sommet et deux sommets a et b étant reliés par une arête s'il existe une route directe reliant la ville a à la ville b.

Chaque arête e est munie d'un poids $h(e)$ donnant la hauteur maximum (en cm) d'un véhicule pouvant emprunter cette route. On souhaite calculer les itinéraires entre les villes maximisant la hauteur possible d'un véhicule.

On considère T_{max} l'arbre couvrant de G de poids maximum.

4.1) Montrer que si une arête $e = \{u, v\}$ n'appartient pas à T_{max}, et si C_{uv} désigne la chaîne reliant u et v dans T_{max}, alors $\forall e' \in C_{uv}, h(e) \leq h(e')$.

Soit $e = uv$ une arête n'appartenant pas à T_{max} et supposons qu'il existe une arête e' dans la chaîne C_{uv} reliant u et v dans T_{max} telle que $h(e) > h(e')$.
On rappelle que dans un arbre T, pour toute paire de sommets u et v, il existe une et une seule chaîne reliant u et v et que l'ajout d'une arête entre u et v créera un et un seul cycle.
Par conséquent, si on ajoute e à T_{max} et que l'on supprime e', on obtient à nouveau un arbre T' qui sera de poids total strictement supérieur à celui de T_{max}, ce qui est contradictoire avec les hypothèses.
Par conséquent, si une arête $e = \{u, v\}$ n'appartient pas à T_{max}, et si C_{uv} désigne la chaîne reliant u et v dans T_{max}, alors $\forall e' \in C_{uv}, h(e) \leq h(e')$.

4.2) Que pouvez-vous en conclure si les poids sont tous différents ?

Si tous les poids sont différents, l'arbre de poids maximum sera unique car on ne pourra échanger dans l'arbre deux arêtes de même poids si l'une est dans l'arbre et l'autre non.

4.3) Que faut-il modifier à l'algorithme de Prim, rappelé à la fin du document (Algorithme 1), pour calculer un arbre de poids maximum ?

Il faut modifier
la ligne 5 en remplaçant $cle(v) \leftarrow \infty$ par $cle(v) \leftarrow -\infty$
la ligne 8 en remplaçant $EXTRAIRE\_MIN$ par $EXTRAIRE\_MAX$ et
la ligne 10 en remplaçant $w(u, v) < cle(v)$ par $w(u, v) > cle(v)$.

4.4) Appliquer l'algorithme de Prim modifié au réseau routier de la Figure 8, en partant du sommet A. Donner l'ensemble des routes à emprunter pour maximiser la hauteur des camions utilisables pour tout transport entre deux villes, et préciser dans quel ordre les sommets ont été rajoutés à l'arbre obtenu.

D'après la question 4.1, la solution sera un arbre de poids maximum. Le résultat est donné par l'arbre de la Figure 9.
L'ordre de rajout des sommets est le suivant : au départ, on a juste le sommet A. Puis on rajoute dans l'ordre : $F, E, C, B, D, H, I, J, G$.

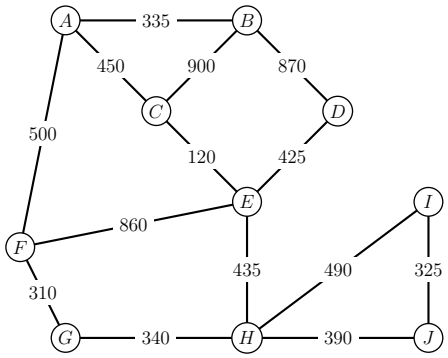

FIGURE 8 – Réseau routier

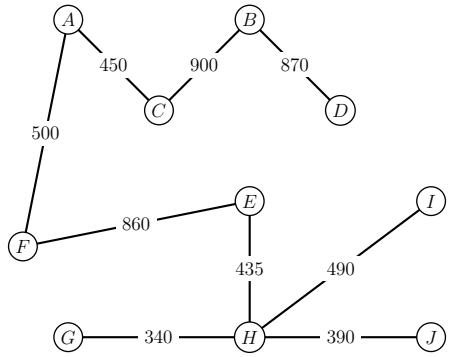

FIGURE 9 – Arbre de poids maximum

Algorithme 1 Prim(G, w)

$F \leftarrow FILE\_PRIORITE(V(G), cle)$
pour tout v de $V(G)$ **faire**
  $cle(v) \leftarrow \infty$
fin pour
$cle(r) \leftarrow 0$
$pere(r) \leftarrow NIL$
tant que $F \neq \emptyset$ **faire**
  $u \leftarrow EXTRAIRE\_MIN(F)$
  pour tout $v \in Adj(u)$ **faire**
    si $v \in F$ et $w(u, v) < cle(v)$ **alors**
      $pere(v) \leftarrow u$
      $cle(v) \leftarrow w(u, v)$
    fin si
  fin pour
fin tant que