université de BORDEAUX

Année 2019-2020 — Session de décembre 2020
Étape : INF501-MI501 — UE : 4TIN501U
Épreuve de : Algorithmique des graphes
Date : 15 décembre 2020 – 9h — **Durée :** 1h30
Épreuve de : O. Baudon — **Nb. pages : 9**

Aucun document autorisé.
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.

1 Plus court chemin

1.1) Rappeler à quelles conditions il est possible d'utiliser l'algorithme de Dijkstra pour calculer les plus courts chemins dans un graphe.

1.2) En utilisant l'algorithme de Dijkstra rappelé à la fin du document (Algorithme 1), trouver les plus courts chemins de s aux autres sommets du graphe G de la Figure 1. A chaque étape, on donnera le pivot utilisé et les modifications apportées aux distances à partir de s. A la fin de l'exécution, donner l'arborescence des plus courts chemins d'origine s.

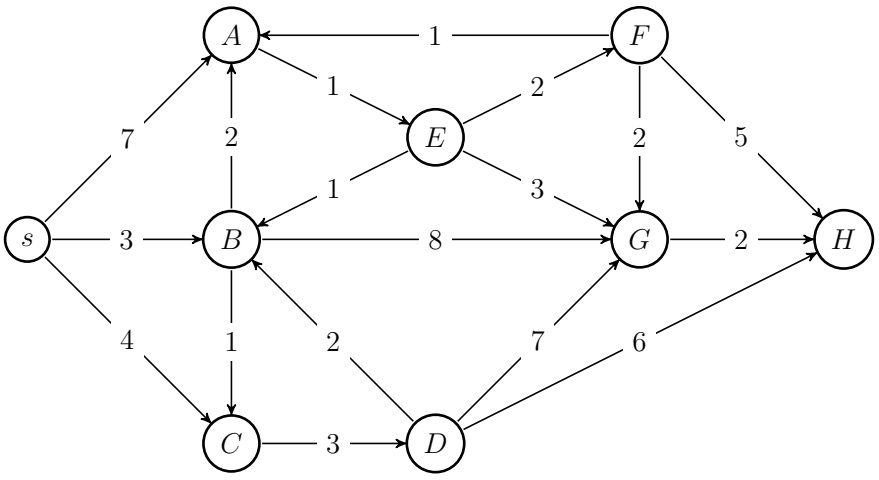


Figure 1 – Graphe G

2 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.

2.1) 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 $\forall e' \in C_{uv}, h(e) \leq h(e')$.

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

2.3) Que faut-il modifier à l'algorithme de Prim, rappelé à la fin du document (Algorithme 2), pour calculer un arbre de poids maximum ?

2.4) Donner l'ensemble des routes à emprunter dans le réseau routier de la Figure 2 pour maximiser la hauteur des camions utilisables pour tout transport entre deux villes.

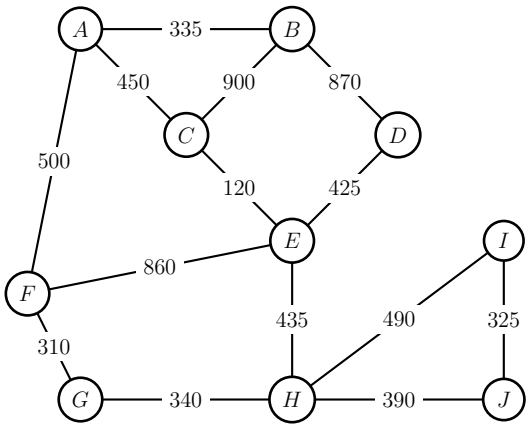


FIGURE 2 – Réseau routier

3 Flot Maximum

3.1) En utilisant l'algorithme de Ford-Fulkerson rappelé à la fin du document (Algorithme 3), trouver le flot maximum entre les sommets s et t du réseau de la Figure 3. 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 8 et une capacité de 13. *Vous devez améliorer ce flot existant, sans repartir d'un flot nul.*

On donnera pour chaque exécution de la procédure de marquage le chemin augmentant obtenu et l'augmentation correspondante.

3.2) Donner la coupe minimum correspondante et redessiner le graphe avec le flot maximum.

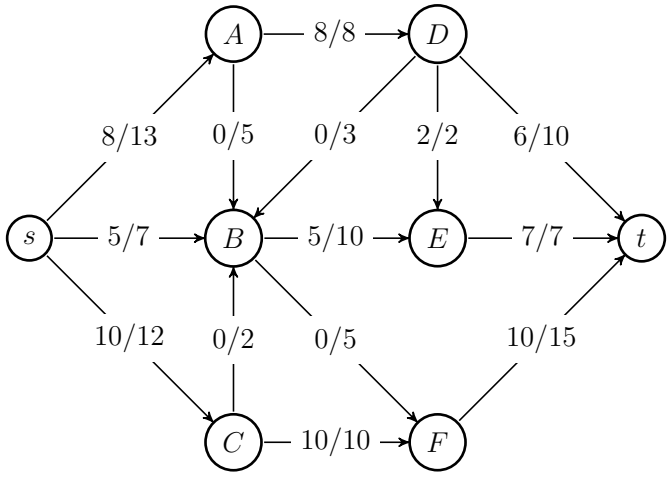


FIGURE 3 – Réseau

4 Interrupteurs

Soit G un graphe simple non orienté à $n > 1$ sommets. Soit s un sommet de G, appelé *sommet initial*. Sur chaque sommet se trouve un interrupteur. Dans ce problème, on se déplace sur le graphe en changeant la position des interrupteurs. Si on se place au sommet s_1 et qu'on franchit l'arête $\{s_1, s_2\}$, l'interrupteur de s_2 change de valeur. Plus formellement, on modélise la situation comme suit.

Une *configuration* est la donnée d'un couple (t, in) où t est un sommet (le sommet courant qui représente où on est dans le graphe) et in un tableau représentant l'état des interrupteurs (0 si l'interrupteur est éteint, 1 sinon). On peut passer de la configuration

(s_1, in_1) à (s_2, in_2) si et seulement si $\{s_1, s_2\} \in E(G)$ et pour tout sommet t de G, on a :

$$\text{in}_2[t] = \begin{cases} 1 - \text{in}_1[t] & \text{si } t = s_2 \\ \text{in}_1[t] & \text{sinon} \end{cases}$$

Dans ce cas, on notera $(s_1, \text{in}_1) \to (s_2, \text{in}_2)$.

Une configuration (t, in') est *accessible* depuis (s, in) si on peut trouver un entier $k \geqslant 1$ et des configurations telles que :

$$(s, \text{in}) = (s_0, \text{in}_0) \to (s_1, \text{in}_1) \to \cdots \to (s_k, \text{in}_k) = (t, \text{in}')$$

Une configuration $\mathcal{C}$ est *résoluble* s'il existe un sommet t tel que $(t, [1; 1; \cdots; 1])$ est accessible depuis $\mathcal{C}$. Le but de cet exercice est de résoudre le problème suivant : étant donné un graphe G, on veut pouvoir déterminer si une configuration $\mathcal{C}$ est résoluble. Autrement dit, on veut trouver un chemin dans G qui commence en un sommet fixé et qui allume tous les interrupteurs.

Par exemple, si on considère le graphe $P3$ si dessous, la configuration $(0, [1\,;\,1\,;\,0])$ est résoluble car :

$$(0, [1; 1; 0]) \to (1, [1; 0; 0]) \to (2, [1; 0; 1]) \to (1, [1; 1; 1])$$

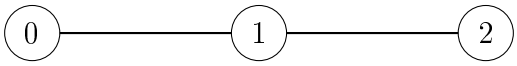
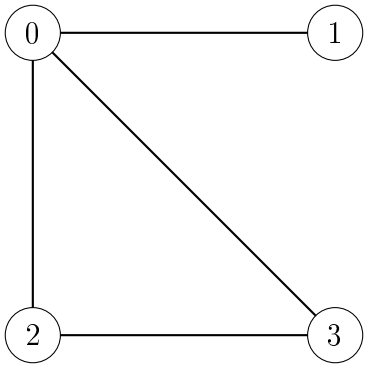


FIGURE 4 – Graphe $P3$

1. Soit $\mathcal{C}$ la configuration $(0, [1; 0; 1; 0])$.
   (a) Donner les configurations (s, in) accessibles depuis $\mathcal{C}$, c'est à dire telles que $\mathcal{C} \to (s, \text{in})$ dans le graphe suivant :



   (b) $\mathcal{C}$ est-elle résoluble ?

2. Dans cette question, on s'intéresse aux configurations résolubles d'un arbre. On va montrer par récurrence sur $n \geqslant 2$ la propriété P_n suivante : "Pour tout arbre A à n sommets et toute configuration (s, in), (s, in) est résoluble".

   (a) Montrer que la proposition P_2 est vraie, c'est à dire que toute configuration dans un arbre à deux sommets est résoluble.

   (b) Soit $n > 2$ un entier tel que P_{n-1} est vraie. Soit A un arbre à n sommets et (s, in) une configuration.

   On rappelle que dans tout arbre ayant au moins deux sommets, il existe au moins deux feuilles distinctes. On peut donc toujours supposer qu'il existe une feuille f avec $f \neq s$.

   Montrer qu'il existe une configuration (s, in') accessible depuis (s, in) telle que $\text{in}'[f] = 1$. (On pourra distinguer les cas $\text{in}[f] = 0$ et $\text{in}[f] = 1$ et exhiber un chemin bien choisi.)

   (c) Conclure la récurrence.

3. En déduire une condition nécessaire et suffisante sur un graphe à $n \geqslant 2$ sommets pour que toutes ses configurations soient résolubles.

Algorithmes (Rappel)

Algorithme 1 Dijkstra(G, w, s)

```
1: pour tout v de V(G) faire
2:    d(v) ← ∞
3:    pere(v) ← NIL
4:    couleur(v) ← BLANC
5: fin pour
6: d(s) ← 0
7: F ← FILE_PRIORITE({s}, d)
8: tant que F ≠ ∅ faire
9:    pivot ← EXTRAIRE_MIN(F)
10:   pour tout e = (pivot, v) arc sortant de pivot faire
11:      si couleur(v) = BLANC alors
12:         si d(v) = ∞ alors
13:            INSERER(F, v)
14:         fin si
15:         si d[v] > d[pivot] + w(e) alors
16:            d[v] ← d[pivot] + w(e)
17:            pere[v] ← pivot
18:         fin si
19:      fin si
20:   fin pour
21:   couleur[pivot] ← NOIR
22: fin tant que
```

Algorithme 2 Prim(G, w)

```
1: F ← FILE_PRIORITE(V(G), cle)
2: pour tout v de V(G) faire
3:    cle(v) ← ∞
4: fin pour
5: cle(r) ← 0
6: pere(r) ← NIL
7: tant que F ≠ ∅ faire
8:    u ← EXTRAIRE_MIN(F)
9:    pour tout v ∈ Adj(u) faire
10:      si v ∈ F et w(u, v) < cle(v) alors
11:         pere(v) ← u
12:         cle(v) ← w(u, v)
13:      fin si
14:   fin pour
15: fin tant que
```

Dans l'Algorithme 3 (FlotMax), le paramètre c désigne les capacités du graphe G, s la source et t la destination du flot f calculé. $I(e)$ et $T(e)$ désignent respectivement le sommet initial et le sommet terminal d'un arc e.

Algorithme 3 FlotMax(G, c, s, t)

```
1: pour tout e de E(G) faire
2:    f[e] ← 0
3: fin pour
4: répéter
5:    Marquage(G, c, f, s, t)
6:    si t ∈ Y alors
7:       v ← t
8:       C⁺ ← {(t, s)}
9:       C⁻ ← ∅
10:      tant que v ≠ s faire
11:         e ← A[v]
12:         si v = T[e] alors
13:            C⁺ ← C⁺ ∪ {e}
14:            v ← I[e]
15:         sinon
16:            C⁻ ← C⁻ ∪ {e}
17:            v ← T[e]
18:         fin si
19:      fin tant que
20:   fin si
21:   pour tout e ∈ C⁺ faire
22:      f(e) ← f(e) + δ[t]
23:   fin pour
24:   pour tout e ∈ C⁻ faire
25:      f(e) ← f(e) − δ[t]
26:   fin pour
27: jusqu'à t ∉ Y
```

Algorithme 4 Marquage(G, c, f, s, t)

1: $Y \leftarrow \{s\}$
2: $\delta(s) \leftarrow +\infty$
3: $Max \leftarrow$ faux
4: **tant que** $t \notin Y et Max =$faux **faire**
5: **si** il existe $e = (u, v)$ avec $u \in Y, v \notin Y, f(e) < c(e)$ **alors**
6: $Y \leftarrow Y \cup \{v\}$
7: $A[v] \leftarrow e$
8: $\delta[v] \leftarrow min(\delta[u], c(e) - f(e))$
9: **sinon**
10: **si** il existe $e = (u, v)$ avec $v \in Y, u \notin Y, f(e) > 0$ **alors**
11: $Y \leftarrow Y \cup \{u\}$
12: $A[u] \leftarrow e$
13: $\delta[u] \leftarrow min(\delta[v], f(e))$
14: **sinon**
15: Max $\leftarrow vrai$
16: **fin si**
17: **fin si**
18: **fin tant que**