Algorithmique (AL5) — Examen — 7 janvier 2022

Justifier proprement vos réponses ; vous ne recevrez pas tous les points pour une réponse correcte sans justification. On peut énoncer des résultats du cours sans les démontrer. Les documents ne sont pas autorisés à l'exception d'une feuille A4 recto-verso. Les appareils électroniques sont interdits. Le document fait 3 pages et contient 7 exercices. Le barème (sur 20 points) est inscrit à titre indicatif et est susceptible de changements.

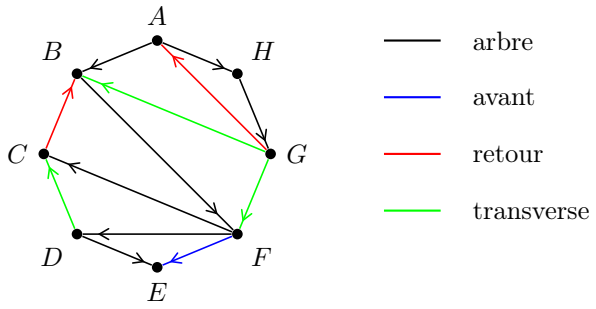
Exercice 1. Parcours en profondeur (3 points)

Effectuer un parcours en profondeur (DFS) sur le graphe ci-dessous, partant du sommet A. Chaque fois que vous avez le choix entre plusieurs sommets, choisir celui qui est le premier par ordre alphabétique.

Question 1. Pour chaque sommet u, indiquer l'intervalle $[\mathrm{pre}(u), \mathrm{post}(u)]$.

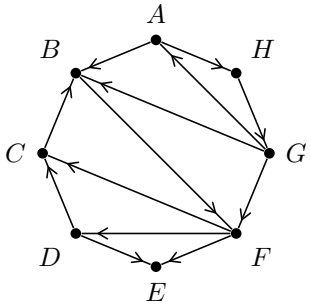
sommet	intervalle
A	$[1, 16]$
B	$[2, 11]$
C	$[4, 5]$
D	$[6, 9]$
E	$[7, 8]$
F	$[3, 10]$
G	$[13, 14]$
H	$[12, 15]$

Question 2. Indiquer le type de chaque arc (arc de l'arbre, avant, retour ou transverse).



Question 3. Avons-nous un DAG (graphe acyclique orienté) ?

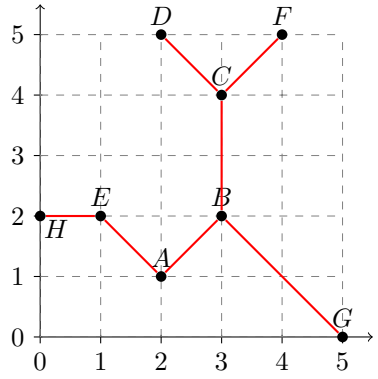
Non, le graphe n'est pas un DAG car il possède des arcs retour.



Exercice 2. Arbre couvrant de poids minimum (2 points)

Soit K le graphe complet dont les 8 sommets correspondent aux 8 points dans le diagramme ci-dessous, tel que le poids de chaque arête est la distance euclidienne (longueur du segment) entre les deux points. Par exemple, la distance entre A et B est $\sqrt{2} \approx 1{,}41$, la distance entre B et C est 2, et la distance entre D et E est $\sqrt{10} \approx 3{,}16$.

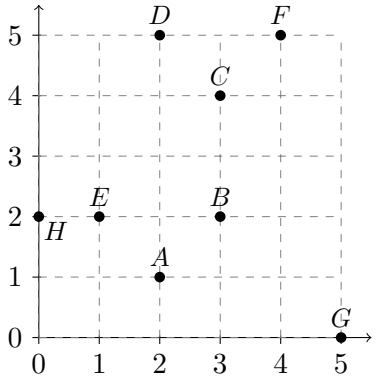
Question 1. Trouver un arbre couvrant de poids minimum de K et indiquer le nom de l'algorithme utilisé.



L'algorithme utilisé : l'algorithme de Kruskal (on peut aussi utiliser l'algorithme de Prim)

Question 2. Donner l'ordre dans lequel les arêtes sont choisies par l'algorithme.

$HE, AB, AE, CD, CF, BC, BG$ (Kruskal)



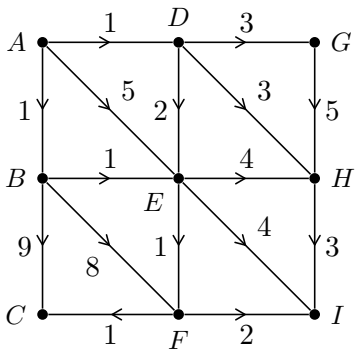
Exercice 3. Plus court chemin (2 points)

Question 1. Utiliser l'algorithme de Dijkstra pour déterminer la distance du sommet A aux autres sommets dans le réseau suivant.

sommet	**distance de A**
A	0
B	1
C	4
D	1
E	2
F	3
G	4
H	4
I	5

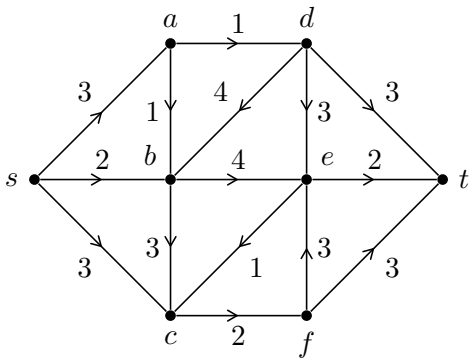
Question 2. Donner l'ordre dans lequel les sommets sont enlevés de la file de priorité.

$A, B, D, E, F, C, G, H, I$



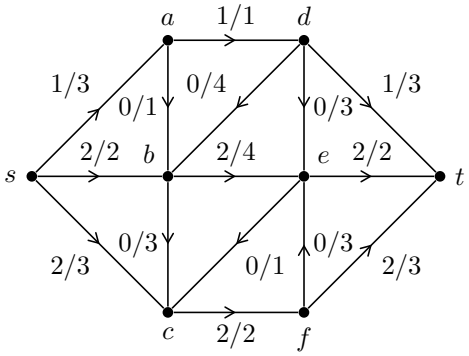
Exercice 4. **Flot max et coupe min (3 points)**

Soit G le réseau ci-dessous :



Question 1. Déterminer la valeur maximum d'un s–t flot dans G et indiquer le nom de l'algorithme utilisé.

La valeur d'un flot maximum : 5. On le trouve avec l'algorithme de Ford–Fulkerson. Un flot de valeur 5 est indiqué ci-dessous :



Question 2. Trouver une s–t coupe dans G dont la capacité est égale à la valeur du flot.

Une $s-t$ coupe de capacité 5 est $(\{s, a, b, c, e\}, \{d, f, t\})$.

Exercice 5. **Vrai ou faux (3 points)**

Répondre vrai ou faux à chacune des questions ci-dessous. Dans cet exercice (et uniquement dans cet exercice), aucune justification n'est demandée.

1. Chaque DAG (graphe orienté acyclique) contient au moins une source (sommet sans aucun arc entrant) et au moins un puits (sommet sans aucun arc sortant).

   Vrai

2. Si un graphe orienté contient au moins une source, alors il contient au moins un puits.

   | Faux

3. Un DAG à n sommets contient n composantes fortement connexes.

   | Vrai

4. Pour trouver un plus long chemin dans un DAG, on peut multiplier tous les poids par -1 et ensuite appliquer l'algorithme de Bellman–Ford.

   | Vrai

5. Un graphe biparti G avec bipartition (A, B) (c'est-à-dire, $V = A \cup B$, $A \cap B = \emptyset$ et toute arête de G a une extrémité dans A et l'autre dans B) contient un couplage parfait ssi $|N(X)| \geq |X|$ pour tout sous-ensemble $X \subseteq A$ (*rappel* : $N(X)$ est l'ensemble des voisins des sommets dans X).

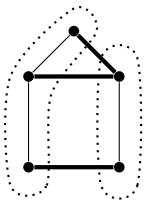
   | Faux (manque la condition $|A| = |B|$)

6. Soient $\nu(G)$ et $\tau(G)$ respectivement la taille maximum d'un couplage et la taille minimum d'un transversal (ou *vertex cover* : un sous-ensemble U des sommets tel que toute arête ait au moins une extrémité dans U). Alors, $\nu(G) \leq \tau(G) \leq 2\nu(G)$.

   | Vrai

Exercice 6. **Max-Cut (3,5 points)**

Étant donné un graphe G et une partition de ses sommets en deux parties, l'ensemble des arêtes avec une extrémité dans chaque partie est appelé une *coupe*. Voici un exemple d'une coupe de taille 3 (arêtes en gras) :



Le problème Max-Cut consiste à trouver, étant donné un graphe $G = (V, E)$, une coupe de taille maximum dans G.

Considérez l'algorithme suivant.

1. Commencer par une partition quelconque de V en deux parties.
2. Tant qu'il existe un sommet avec strictement plus de voisins dans sa partie que dans l'autre partie, déplacer le sommet vers l'autre partie.
3. Retourner l'ensemble des arêtes avec exactement une extrémité dans chaque partie.

Question 1. Montrer que l'algorithme termine au bout d'au plus $|E|$ itérations et en déduire qu'il s'agit d'un algorithme polynomial (c'est-à-dire, sa complexité est bornée par un polynôme en terme de $|V|$).

> Il suffit d'observer qu'à chaque itération, le nombre d'arêtes dans la coupe augmente d'au moins 1. Comme le nombre d'arêtes dans une coupe ne peut pas dépasser le nombre d'arêtes dans le graphe, on conclut que l'algorithme termine au bout d'au plus $|E|$ itérations.
> À chaque itération nous devons vérifier au pire tous les $|V|$ sommets et parcourir les $|E|$ arêtes, donc le temps total est $O(|E|(|E| + |V|)) = O(|V|^4)$; il s'agit bien d'un algorithme polynomial.

Question 2. Montrer que la coupe obtenue a au moins $|E|/2$ arêtes.

Soit (A, B) la bipartition trouvée par l'algorithme. La coupe trouvée est de taille $|\delta(A)|$. On a

$$\begin{aligned} 2|\delta(A)| &= |\delta(A)| + |\delta(B)| \\ &\geq \tfrac{1}{2} \sum_{v \in A} d(v) + \tfrac{1}{2} \sum_{v \in B} d(v) \\ &= \tfrac{1}{2} \sum_{v \in V} d(v) \\ &= |E|, \end{aligned}$$

donc $|\delta(A)| \leq |E|/2$.

Question 3. En déduire que c'est un algorithme de 2-approximation (c'est-à-dire, une coupe maximale contient au plus deux fois le nombre d'arêtes de la coupe trouvée par l'algorithme).

Le facteur d'approximation d'un algorithme de maximisation $\mathcal{A}$ est $\rho = \min_I \frac{\mathrm{OPT(I)}}{\mathcal{A}(I)}$, où le minimum est prix parmi toutes les instances I. (Remarque : la fraction est parfois inversée dans la définition de ρ, et on utilise max au lieu de min.) Sachant que $\mathrm{OPT(I)} \leq |E|$ et $\mathcal{A}(I) \geq |E|/2$, on conclut que $\rho \leq 2$, on a donc bien un algorithme de 2-approximation.

Exercice 7. Deux ivrognes (3,5 points)

Deux ivrognes ont à se partager équitablement (c'est-à-dire par moitié) 8 litres de vin. Ils disposent de trois cruches : celle qui contient les 8 litres, et deux autres, de contenances respectives 5 litres et 3 litres. Aucune d'entre elles n'étant graduée, ils ont recours à une seule opération : verser le contenu d'une cruche (source) dans une autre (destination), en ne s'arrêtant que lorsque la cruche source soit vide ou que la cruche de destination soit pleine.

Question 1. Modéliser ce problème comme un problème de graphe : donner une définition précise du graphe concerné (quels sont les sommets et les arêtes ou arcs) et énoncer la question spécifique à ce graphe à laquelle il faut répondre.

On peut représenter chaque état par un triple (x, y, z), où x, y et z représentent la quantité de vin dans la cruche à 8, 5 et 3 litres, respectivement. On associe à chaque état un sommet, et on met un arc de (x, y, z) vers (x', y', z') s'il existe une opération qui nous ramène de l'état (x, y, z) vers l'état (x', y', z'). La question est alors : existe-t-il un chemin (orienté) du sommet $(8, 0, 0)$ vers le sommet $(0, 4, 4)$?

Question 2. Quel algorithme peut être appliqué pour résoudre le problème ?

On peut utiliser DFS ou BFS.

Question 3. Trouver la réponse en appliquant l'algorithme.

Voici une solution possible : $(8, 0, 0) \to (3, 5, 0) \to (3, 2, 3) \to (6, 2, 0) \to (6, 0, 2) \to (1, 5, 2) \to (1, 4, 3) \to (0, 4, 4)$.