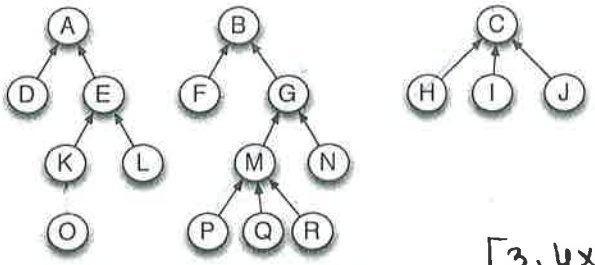
AL5 – Algorithmique
Examen

Jeudi 7 janvier 2016 15h30–18h30

Documentation autorisée : une feuille A4 recto verso manuscrite. Toute autre documentation et appareil sont interdits. **Les téléphones portables doivent être éteints et déposés dans votre sac et vous devez poser vos sacs et vos vestes à l'avant de la salle avant de commencer.**

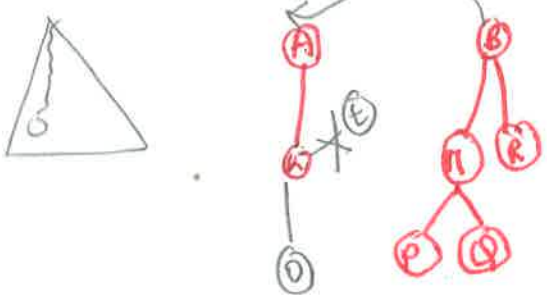
Indications : Le barème est donné à titre indicatif seulement. Justifiez vos réponses et expliquez vos algorithmes. Les algorithmes doivent être aussi efficaces que possible et présentés de façon lisible et claire. Cela veut dire que les noms de variables doivent être bien choisis pour représenter ce qu'elles sont censées contenir et que les principales étapes doivent être accompagnées d'explications. Une partie de la note portera sur la clarté de présentation de vos réponses (mais pas leur longueur).

Exercice 1 : (4 points)



La figure ci-dessus représente la structure de données obtenue en appliquant successivement UNION à toutes les paires de sommets formant des arêtes dans un graphe non orienté G.

1. Quelles sont les composantes connexes de G ?
2. Que peut-on dire sur le nombre d'arêtes dans le graphe G ? Donnez une borne inférieure et une borne supérieure, et justfiez brièvement votre réponse.
3. Donnez le résultat de l'application de UNION(K,R), sans compression de chemin.
4. En repartant de la structure de données dans la figure, donnez le résultat de l'application de UNION(K,R) avec compression de chemin.



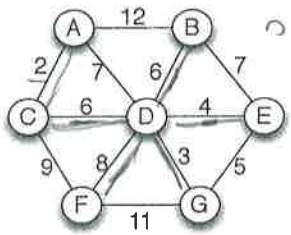
Exercice 2 : (6 points)

Les arbres (a, b) sont des arbres k-aires qui ont les propriétés suivantes :
- La racine a au plus b fils ;
- Chaque nœud interne a entre a et b fils (a et b inclus) ;
- Toutes les feuilles ont la même profondeur.

Les arbres sont représentés comme en cours, c'est-à-dire qu'un arbre est identifié par sa racine et chaque nœud `u` possède un attribut `u.fils` qui est une liste de ses fils. Vous pouvez utiliser des fonctions auxiliaires standard sur les listes comme `l.length()` pour obtenir la longueur d'une liste `l`, mais n'oubliez pas de tenir compte de la complexité de ces fonctions dans le calcul de la complexité de votre algorithme. Si nécessaire vous pouvez ajouter des attributs aux nœuds.

1. Donnez un algorithme qui prend en entrée un arbre k-aire et deux entiers a et b, et retourne VRAI si l'abre est un arbre (a, b) et FAUX sinon. Pour avoir tous les points il faut que votre algorithme soit le plus efficace possible, et pour justifier qu'il est correct (question suivante) vous aurez intérêt à ce qu'il soit le plus simple possible.
2. Sans donner une preuve formelle, donner un argument pour justifier que votre algorithme est correct. Si votre algorithme est basé sur un algorithme de parcours vu en cours, vous pouvez utiliser les propriétés du parcours que nous avons démontrées pour appuyer vos arguments.
3. Donnez la complexité de votre algorithme en pire cas en fonction du nombre de nœuds.

Exercice 3 : (2 points)

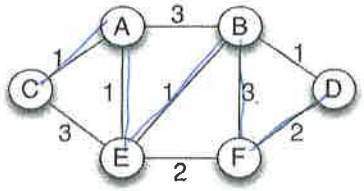


1. Exécutez l'algorithme de Prim sur le graphe ci-dessus, en arrêtant lorsque vous aurez sélectionné 5 arêtes. Donnez le résultat partiel obtenu et précisez l'ordre dans lequel les arêtes ont été considérées lors de l'exécution.
2. Exécutez l'algorithme de Kruskal sur le graphe ci-dessus, en arrêtant lorsque vous aurez sélectionné 5 arêtes. Donnez le résultat partiel obtenu et précisez l'ordre dans lequel les arêtes ont été considérées lors de l'exécution.

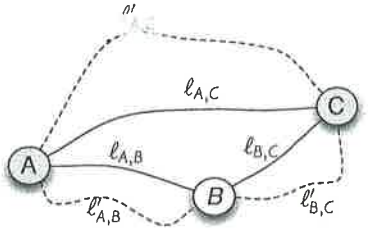
Exercice 4 : (8 points)

Dans cet exercice l'objectif est de donner un algorithme qui retourne la longueur des *deux* plus courts chemins à partir de tous les sommets de départ s jusqu'à tous les sommets d'arrivée t. Par exemple, s'il y a des chemins de longueur 12, 14, 14, 15, 19 et 32 de s à t, alors votre algorithme doit retourner 12 et 14 pour cette paire de sommets, et de même pour toutes les autres paires de sommets source-destination.
Vous devez modifier un des algorithmes de plus courts chemins toutes sources toutes destinations vus en cours.

1. Dans le graphe ci-dessous, donner les 2 plus courts chemins de C à D. (Donnez les chemins ainsi que leurs longueurs.)



2. (Cette question a pour but de vous aider à concevoir un algorithme qui trouve les deux plus courts chemins.) Soit $\mathcal{E}$ un sous-ensemble de sommets et B un sommet n'appartenant pas à $\mathcal{E}$. Supposons que vous ayez trouvé
   - les deux plus courts chemins de A à B dont les sommets intermédiaires sont tous dans l'ensemble $\mathcal{E}$, de longueur $l_{A,B}$ et $l'_{A,B}$ respectivement ($l_{A,B} \leqslant l'_{A,B}$),
   - les deux plus courts chemins de B à C, dont les sommets intermédiaires sont dans $\mathcal{E}$, de longueur $l_{B,C}$ et $l'_{B,C}$ respectivement ($l_{B,C} \leqslant l'_{B,C}$), ainsi que
   - les deux plus courts chemins de A à C, dont les sommets intermédiaires sont dans $\mathcal{E}$, de longueur $l_{A,C}$ et $l'_{A,C}$ respectivement ($l_{A,C} \leqslant l'_{A,C}$).



   Donner une expression qui donne la longueur des deux plus courts chemins de A à C dont les sommets intermédiaires sont dans l'ensemble $\mathcal{E} \cup \{B\}$. Pensez bien à tous les cas de figure.
3. En vous inspirant de la question précédente, nommez l'algorithme vu en cours que vous pouvez utiliser comme point de départ, et expliquez brièvement comment le modifier pour trouver les deux plus courts chemins.
4. Donnez l'algorithme ainsi modifié. Il prend en entrée un graphe G, une fonction de coût w, et retourne deux matrices `D1`, `D2` qui pour chaque paire s, t donnent dans `D1[s][t]` la longueur du plus court chemin de s à t et dans `D2[s][t]` la longueur du deuxième plus court chemin.
5. Donnez la complexité de votre algorithme. Justifiez brièvement votre réponse. n^3
6. Donnez les principales étapes de l'exécution de votre algorithme sur l'exemple de la question 1.