Numéro d'anonymat :

Algorithmique (LU3IN003).
L3 Informatique.

Examen du 3 janvier 2023. Durée : 2 heures

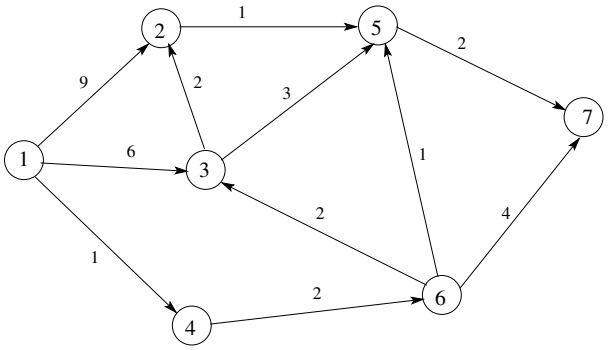
Documents non autorisés. Seule une feuille A4 portant sur les cours et les TD est autorisée.
Téléphones portables éteints et rangés dans vos sacs.

Le barème est indicatif et est susceptible d'être modifié.

Exercice 1 (3 points)

Répondez au questionnaire à choix multiples suivant, en cochant la case correspondant à la bonne réponse pour chaque question. Chaque bonne réponse apportera 0.5 point. Chaque mauvaise réponse coûte une pénalité de 0.25 point, une absence de réponse n'étant pas comptabilisée comme une mauvaise réponse.

Questions	**Réponses**
1. Quelle est la complexité d'un algorithme du type "diviser pour régner" qui divise un problème de taille n en quatre sous-problèmes de taille $n/2$ et dont l'opération de fusion est en $O(n)$?	☐ $\Theta(\log n)$ ☐ $\Theta(n)$ ☐ $\Theta(n \log n)$ ☐ $\Theta(n^2)$ ☐ $\Theta(n^3)$
2. Quel algorithme peut-on utiliser pour détecter la présence d'un circuit absorbant dans un graphe ?	☐ parcours en prodondeur ☐ algorithme de Bellman-Ford ☐ algorithme de Dijsktra ☐ algorithme de Bellman
3. Quel algorithme n'est pas de type "diviser pour régner" ?	☐ l'algorithme de tri fusion ☐ recherche dichotomique ☐ algorithme de Prim ☐ algorithme de Karatsuba
4. Exécuter l'algorithme de Dijkstra sur le graphe de la question 4b, afin de trouver une arborescence des plus courts chemins de racine 1. Dans quel ordre les sommets sont-ils examinés ?	☐ (1,4,3,2,6,5,7) ☐ (1,4,6,3,5,7,2) ☐ (1,4,6,5,3,7,2) ☐ (1,4,6,3,2,5,7) ☐ (1,2,3,4,5,6,7)
	suite sur la page suivante...

Questions	Réponses
4b. Représenter sur le graphe *l'arborescence des plus courts chemins* retournée par l'algorithme de Dijkstra (vous pouvez surligner les arcs sélectionnés) : 	
5. Cocher la permutation des sommets qui peut correspondre à un parcours en profondeur de racine 1 dans le graphe représenté dans la question 4b.	☐ (1,2,3,4,5,6,7) ☐ (1,3,5,2,4,7,6) ☐ (1,2,5,3,4,6,7 ☐ (1,4,6,7,5,3,2) ☐ (1,7,2,6,4,3,5)

Exercice 2 (3 points)

On considérera dans cet exercice un graphe $G = (S, A)$ orienté et valué. On notera $c(e)$ le coût de l'arc $e \in A$. On supposera que $s \in S$ est une racine et qu'il n'y a pas de circuit absorbant dans G. Soit $\mathcal{A}$ une arborescence des plus courts chemins d'origine s. On notera $d(x)$ le coût du chemin de s à x dans $\mathcal{A}$. Le but de cet exercice est de voir dans quelle mesure des perturbations sur un graphe changent une arborescence des chemins de coût minimum.

Question 1 (1/3) — Supposons que l'on multiplie par 10 le coût de chaque arc. Soit G' le graphe obtenu (i.e. on a $G' = (S, A)$ et pour tout arc $e \in A$, le coût de e dans G' est $10 \times c(e)$). L'arborescence $\mathcal{A}$ est-elle nécessairement une arborescence des plus courts chemins d'origine s dans G' ? Justifiez votre réponse.

Question 2 (1/3) — Supposons que l'on retire 1 au coût de chaque arc. Soit G'' le graphe obtenu (i.e. on a $G'' = (S, A)$ et pour tout arc $e \in A$, le coût de e dans G'' est $c(e) - 1$). L'arborescence $\mathcal{A}$ est-elle nécessairement une arborescence des plus courts chemins d'origine s dans G'' ? Justifiez votre réponse.

Question 3 (1/3) — Soit a et b deux sommets de S tels que $(a, b) \notin A$. Soit G''' le graphe G dans lequel on ajoute cet arc : $G''' = (S, A \cup \{(a, b)\})$. Pour tout arc $e \in A$, le coût de e dans G''' est $c(e)$, et le coût de l'arc (a, b) est de 5. À quelle condition $\mathcal{A}$ est-elle une arborescence des plus courts chemins d'origine s dans G''' ? Justifiez votre réponse.

Exercice 3 (7 points)

Une commune possède une parcelle de forêt rectangulaire de $n \times m$ kilomètres (ce qui représente un rectangle de n kilomètres sur m kilomètres). La commune souhaite vendre cette parcelle et en tirer le plus d'argent possible. Pour cela, elle peut diviser son terrain en plusieurs terrains plus petits à l'aide de coupes horizontales et verticales.

Définition 1. *On appelle* coupe horizontale *une séparation d'un terrain T de taille $i \times j$ en deux terrains T_1 de taille $i_1 \times j$ et T_2 de taille $i_2 \times j$ avec $i_1 + i_2 = i$ et i_1 et i_2 des entiers strictement positifs. Une telle coupe sera notée $Coupe\_Horizontale(T, i_1, T_1, T_2)$.*

Définition 2. *On appelle* coupe verticale *une séparation d'un terrain T de taille $i \times j$ en deux terrains T_1 et T_2 de taille $i \times j_1$ et $i \times j_2$ avec $j_1 + j_2 = j$ et j_1 et j_2 des entiers strictement positifs. Une telle coupe sera notée $Coupe\_Verticale(T, j_1, T_1, T_2)$.*

La commune peut effectuer plusieurs coupes à la suite. On appelle *plan de coupe* une liste P de coupes successives et réalisables. Cette liste peut être vide. Certains plans de coupe différents peuvent aboutir à un même ensemble de terrains. On illustre ces définitions par l'exemple suivant.

Exemple 1. *Si la commune dispose d'un terrain T de 4×4 kilomètres carré, elle peut le découper en deux dans le sens vertical à un kilomètre du bord du terrain. Elle obtient alors un terrain T_1 de taille 4×1 et un autre T_2 de taille 4×3. Elle peut ensuite découper T_2 au milieu dans le sens horizontal : elle aura alors le terrain T_1 de taille 4×1 et deux terrains T_3 et T_4 de taille 2×3. Ce plan de coupe sera noté $P = (Coupe\_Verticale(T, 1, T_1, T_2), Coupe\_Horizontale(T_2, 2, T_3, T_4))$.*

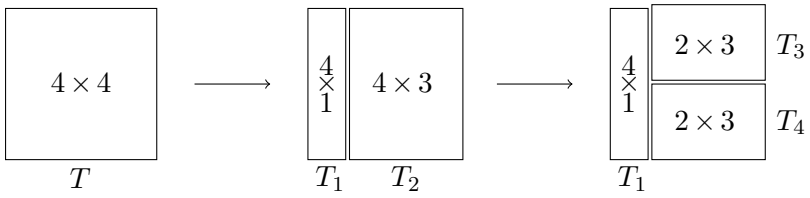


FIGURE 1 – Plan de coupe P

Question 1 (0.5/7) — Indiquer les 5 plans de coupe possibles si la commune dispose d'un terrain T de taille 3×1.

Pour chaque terrain T_k de $i \times j$ kilomètres carré, la commune peut vendre T_k au prix $M(i,j)$ où M est une matrice de prix. On définit une fonction *eval* qui prend en paramètres les dimensions d'un terrain T et un plan de coupe P sur T et retourne l'argent obtenu en vendant les terrains obtenus à partir de T en suivant le plan de coupe P. On note ainsi $eval(i,j,P)$ la somme obtenu en vendant les terrains issus des coupes P à partir d'un terrain de taille $i \times j$ (cette fonction fait la somme des prix des terrains obtenus à l'issue du plan P).

Exemple 1. ***(suite)*** *On considère deux plans de coupe :*

— $P_1 = ()$

— $P_2 = (Coupe\_Verticale(T, 1, T_1, T_2), Coupe\_Horizontale(T_2, 2, T_3, T_4))$.

P_1 est un plan sans coupe, et P_2 est le plan de coupe obtenu sur la figure 1.
On considère la matrice de prix suivante :

M	$i=1$	$i=2$	$i=3$	$i=4$
$j=1$	1	2	4	5
$j=2$	2	3	8	9
$j=3$	4	8	10	11
$j=4$	5	9	11	13

TABLE 2 – Matrice de prix M

A l'issue de P_1, la commune a un terrain de dimensions 4×4 : on a donc $eval(4,4,P_1) = M(4,4) = 13$. A l'issue de P_2, la commune a un terrain de taille 4×1 et deux terrains de taille 2×3, on a alors $eval(4,4,P_2) = M(4,1) + M(2,3) + M(2,3) = 5 + 8 + 8 = 21$.

Le but de cet exercice est de déterminer la somme d'argent maximale que la commune peut obtenir à partir en faisant des coupes de son terrain de taille $n \times m$ (on ne cherche pour l'instant pas à savoir quelles coupes il faut faire pour obtenir cette somme maximale). On va pour cela concevoir un algorithme de programmation dynamique.

On appelle *eval_opt* la fonction prenant en paramètres les dimensions d'un terrain T et retournant la somme d'argent maximale que l'on peut obtenir en suivant un plan de coupe pour le terrain T. Ainsi, $eval\_opt(i,j)$ sera la somme maximale que l'on peut obtenir en coupant un terrain de taille $i \times j$ via une succession de coupes.

Question 2 (0.5/7) — Soit T un terrain de taille $i \times j$ et soit $C = Coupe\_Horizontale(T, i_1, T_1, T_2)$ une coupe de ce terrain. Exprimer, en fonction des dimensions de T_1 et T_2, la plus grande valeur que l'on peut obtenir à partir du terrain T avec un plan de coupes P commençant par la coupe C.

Question 3 (1.5/7) —

(a) Si la commune possède un terrain de taille $i \times j$, par quelles coupes peut-elle commencer son plan (en supposant que la commune décide de couper son terrain) ?

(b) En déduire une relation de récurrence permettant d'exprimer $eval\_opt(i, j)$.

Question 4 (0.5/7) — Pour quelles valeurs de i et j la fonction $eval\_opt(i, j)$ donne-t-elle le revenu maximal que la commune peut obtenir ?

Question 5 (2/7) — Écrire un algorithme de programmation dynamique pour calculer ce revenu maximal.

Question 6 (1/7) — Quelle est la complexité de cet algorithme en fonction de n et m ? Justifiez votre réponse.

Question 7 (1/7) — Décrire en quelques phrases comment modifier cet algorithme pour qu'il retourne également un plan de coupe menant à un revenu maximal. La complexité de l'algorithme est-elle modifiée ?

Exercice 4 (9 points)

On considère l'algorithme suivant, appelé CoupeCycles, que l'on applique sur un graphe connexe $G = (S, A)$:

Entrées : $G = (S, A)$: un graphe connexe non orienté pondéré.
Trier les arêtes de A par coûts décroissants : $c(e_1) \geq \cdots \geq c(e_m)$.
$R = \emptyset$.
pour *chacune des arêtes e_i, examinées par coûts décroissants,* **faire**
  si *le graphe est toujours connexe après suppression de l'arête e_i* **alors**
    Supprimer l'arête e_i : $A = A \setminus \{e_i\}$.
    $R = R \cup \{e_i\}$.
  fin
fin
Retourner (S, A).

Algorithme 1 : Algorithme CoupeCycles.

Question 1 (0.5/9) — Exécuter cet algorithme sur le graphe $G_1 = (S_1, A_1)$ de la figure 2. Vous indiquerez l'ordre dans lequel les arêtes sont examinées et vous surlignerez les arêtes qui restent dans A à l'issue de l'exécution de l'algorithme.

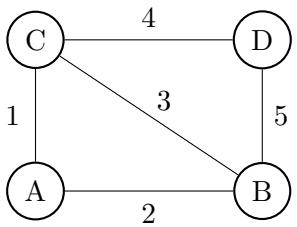


Figure 2 – Graphe G_1.

Les questions suivantes concernent le cas général, i.e. **un graphe G connexe quelconque.**

Question 2 (1.5/9) — Comment peut-on vérifier qu'un graphe G est toujours connexe après suppression d'une de ses arêtes $e_i = \{a, b\}$? Vous indiquez en une phrase ou deux comment il est possible de faire cette vérification en utilisant un algorithme vu en cours. Vous préciserez également la complexité de cette vérification en fonction du nombre de sommets et/ou d'arêtes du graphe.

Question 3 (1/9) — Montrer que l'algorithme CoupeCycles retourne un arbre couvrant de G.

Question 4 (3.5/9) — Montrer qu'à l'issue de chaque itération de l'algorithme, il existe un arbre couvrant de coût minimum qui ne contient pas les arêtes de R. Vous pouvez pour cela faire une preuve par induction, en montrant que cette propriété est vérifiée avant la première itération, et que si elle est vérifiée à la fin de l'itération $(i-1)$ elle est également vérifiée à la fin de l'itération i.

Indication : on notera R_k les arêtes de l'ensemble R à la fin de l'itération k. On pourra raisonner en considérant, à la fin de la i-ème itération, un arbre couvrant de coût minimum $\mathcal{A}$ ne contenant pas les arêtes de R_{i-1}. En partant de cet arbre, on montrera qu'il existe un arbre couvrant de coût minimum ne contenant pas les arêtes de R_i.

Question 5 (1/9) — Montrer que l'arbre retourné par l'algorithme CoupeCycles est un arbre couvrant de coût minimum de G.

Question 6 (1.5/9) — Quelle est la complexité de cet algorithme ? Cette complexité est-elle meilleure que la complexité des deux autres algorithmes retournant des arbres couvrants de coût minimum que vous avez vus en cours ? Vous indiquerez le nom de ces algorithmes et justifierez votre réponse.