Algorithmique Avancée — Examen Réparti 1

Les seuls documents autorisés sont les polys de cours, ainsi que la copie double personnelle. Le barème donné est indicatif.

Exercice 1 *: QCM [5 points]*

Dans ce QCM, pour chaque question vous devez donner 1 seule réponse et expliquer votre choix par une ligne de texte ou une figure. Le barème est le suivant : **Réponse correcte et correctement argumentée : 1 point. Réponse incorrecte ou correcte mais mal argumentée : 0 point.** *Il n'y a pas de points négatifs.*

Question 1 Dans un tournoi binomial :

A. Le parcours préfixe donne les clés dans l'ordre croissant.

B. Le parcours infixe donne les clés dans l'ordre croissant.

C. Pour toute branche ayant pour feuille μ, pour tout nœud ν de cette branche, le chemin de ν vers μ est étiqueté de façon croissante.

Question 2 Toute file binomiale

A. a une taille qui est une puissance de 2.

B. est une file binomiale relâchée.

C. contient au moins deux tournois binomiaux.

Question 3 Pour deux files binomiales de taille m et n telles que $m \leq n$. La complexité de fusion des deux files est en $O(\log n)$ comparaisons de clés,

A. dans le pire des cas.

B. en complexité amortie.

C. en moyenne.

Question 4 Dans un arbre $2-3-4$, l'algorithme d'insertion avec éclatements à la remontée est tel que

A. la hauteur de l'arbre obtenu est toujours strictement plus faible qu'avec des éclatements à la descente.

B. on doit faire, dans certains cas, plusieurs aller-retours de la feuille vers ses ancêtres pour gérer les éclatements.

C. la hauteur de l'arbre peut être plus faible qu'avec des éclatements à la descente.

Question 5 M. Martin candidat à l'élection présidentielle souhaite pendant sa campagne se rendre dans les 100 plus grandes villes de France. Pouvez vous lui proposer un circuit optimal qui passe une et une seule fois par toutes ces villes ?

A. Oui.

B. Non.

Exercice 2 *: Arbres couvrants [4 points]*

Question 1 L'algorithme de Kruskal retourne des arbres couvrants différents pour un même graphe G, selon l'ordre dans lequel sont examinées les arêtes de même poids dans la liste triée. Montrer que, pour chaque arbre couvrant minimal T de G, il existe un moyen de trier les arêtes de G dans l'algorithme de Kruskal pour que l'algorithme retourne T.

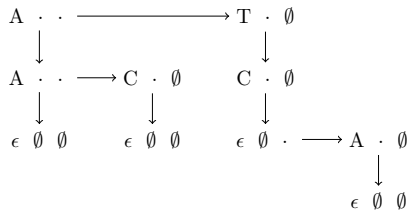
Question 2 Exécuter l'algorithme de Dijkstra (ci-dessous) sur le graphe donné sur la feuille jointe en complétant les lettres et nombres manquant dans les crochets. Le premier sert à l'initialisation, et les autres sont pour chaque tour de boucle `While`. (Ne pas oublier d'inscrire vos nom et prénom sur la feuille).

```
def Dijkstra(G, a):
        """ Graphe * sommet -> (liste * liste)
            Hypotheses : G = (S, A, v) un graphe avec une valuation positive
                         a appartient a S.
        """

        R = {a}
        pere[s] = a   pour chaque sommet s de S
        dist[s] = +infini   pour chaque sommet s de S
        dist[a] = 0

        while (le cocycle de R est non vide):
            Choisir u=(y,z) de poids minimum dans le cocyle de R
            for (tous les voisins t de z tels que t n est pas dans R):
                if (dist[t] > dist[z] + v(z,t)):
                    dist[t] = dist[z] + v(z,t)
                    pere[t] = z
            R = union(R, {z})

        return (pere, dist)
```

Exercice 3 *: Arbre de la Briandais [6 points]*

Les arbres de la Briandais consistent en une représentation compacte des R-tries. Chaque chemin dans l'arbre est un préfixe d'un mot stocké (dans l'arbre). Ainsi, chaque nœud contient une liste chaînée. On a besoin d'un nouveau caractère (ϵ) pour indiquer la fin d'un mot. On suppose que les éléments de la liste frères sont ordonnés selon l'ordre alphabétique. Dans cet exemple sur 5 caractères : $\epsilon < A < C < G < T$. Le pointeur vers `Nil` est représenté par $\emptyset$.

Ainsi, les mots AA, AC, TCA et TC sont encodés en arbre de la Briandais ainsi :



Question 1 Encoder l'ensemble des mots suivants dans un arbre de la Briandais :
`bon, bonheur, algav, arité, arrêt, trie`.

Question 2 Écrire un algorithme effectuant la suppression d'un mot dans un arbre de la Briandais.
Indiquer une mesure de complexité ayant du sens dans ce contexte, et la complexité de l'algorithme.

Question 3 Écrire un algorithme effectuant la fusion de deux arbres de la Briandais.
Indiquer une mesure de complexité ayant du sens dans ce contexte, et la complexité de l'algorithme.

Exercice 4 *: (Problème) Tournoi auto-adaptatif [10 points]*

Un tournoi auto-adaptatif est un arbre binaire étiqueté, tel que la suite des clés de la racine à n'importe quelle feuille est croissante (strictement). La taille d'un arbre est le nombre de clés qu'il contient. Voilà les notations que nous utiliserons :

- $< >$ est le tournoi vide
- $< a >$ est le tournoi réduit à une feuille dont la clé est a
- $< T1\ a\ T2 >$ est le tournoi dont la racine est la clé a, et les tournois $T1$ et $T2$ sont les fils gauche et droit.

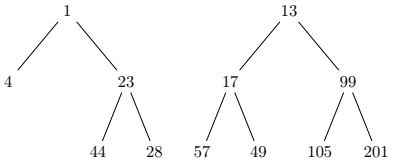
On a des opérations de base sur les tournois A et B (ne contenant aucune clé identique, chacun de taille > 1).

```
Fusion(A, < >) = A          Fusion(< >, A) = A
Fusion(A, B) = Jonction(A, B)   # si la racine de A est plus petite que celle de B
Fusion(A, B) = Jonction(B, A)   # sinon

Jonction(<A1 a A2>, B) = Mix(<A1 a Fusion(A2, B)>)

Mix(<A1 a A2>) = <A2 a A1>   # si |A1|>1 ou |A2|>1
Mix(<A1 a A2>) = <A1 a A2>   # sinon
```

L'algorithme (principal) qui permet de ***fusionner deux tournois*** *A* ***et*** *B est appelé via* `Fusion(A, B)`.

Question 1 Soit A (à gauche) et B (à droite) les deux arbres suivants :

Donner l'arbre résultant de `Fusion(A, B)`.



Question 2 Donner un algorithme d'insertion de la clé x dans un tournoi A, basé sur la fonction `Fusion`.

Question 3 À partir de l'arbre vide, insérer successivement les clés `1,2,3,4`.
À partir de l'arbre vide, insérer successivement les clés `4,3,2,1`.

Question 4 Prouver que la `Fusion` de 2 tournois est un tournoi. Prouver la correction de `Insertion`.

Question 5 Donner un algorithme pour la fonction d'extraction de la clé minimale d'un tournoi A et renvoyant un couple formé de la clé min. et d'un tournoi contenant les autre clés de A. Prouver la correction de `ExtractionMin`.

Le *poids* d'un nœud est la taille du sous-arbre enraciné en ce nœud. Un nœud (autre que la racine) est appelé *gauche* (respectivement *droit*) s'il est le fils gauche de son père (resp. fils droit). On dit qu'un nœud (autre que la racine) est *lourd*, si son poids est strictement supérieur à la moitié du poids de son père. Sinon il est *léger*. La racine n'est ni lourde ni légère. On définit le potentiel $\phi(A)$ pour un arbre A, le nombre de nœuds droits et légers qu'il contient. On pose $\phi(<>) = 0$.

Question 6 Étant donné $A =< A1\ a\ A2 >$, donner la récurrence permettant de calculer $\phi(A)$.

La mesure de complexité correspond au nombre de comparaison de clés ; on note $|A|$ la taille de l'arbre A et $\hat{c}(op)$ le coût amorti pour effectuer l'opération *op*.

Question 7 Soit A un tournoi de taille n. Étant donné un nœud de A, montrer que ses deux fils ne peuvent pas être simultanément lourds.

Question 8 Soit A un tournoi de taille n. Étant donné un chemin de la racine à une feuille de A, montrer que ce chemin contient au plus $\lfloor \log_2 n \rfloor$ nœuds légers.

Question 9 Soit A et B deux tournois de tailles respectives n_1 et n_2, avec $n_1 + n_2 = n$. Montrer que la complexité amortie de `Fusion(A,B)` est $O(\log n)$.

Question 10 Conclure l'exercice concernant la complexité amortie sur les tournois auto-adaptatifs.