Algorithmique 1

Devoir final

Les notes de cours et de TD sont autorisées. Pour les questions demandant de dérouler un algorithme, vous veillerez à faire apparaître les étapes de calcul permettant de juger de votre bonne compréhension (le résultat final est rarement suffisant). Le barème indiqué n'est pas définitif.

Question 1.[1pt] Présenter les 7 arbres de recherche binaires obtenus durant la séquence d'opérations suivante (en partant d'un arbre vide).

```
INSERE(5); INSERE(6); INSERE(2); INSERE(4); INSERE(1); INSERE(3); SUPPRIME(5);
```

en utilisant les procédures décrite en cours.

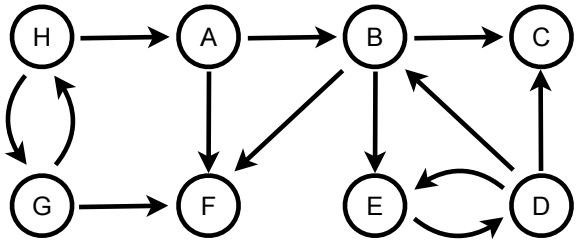
Question 2.[2pt] Même question avec des AVL pour la séquence suivante.

```
INSERE(1); INSERE(2); INSERE(3); INSERE(5); INSERE(4); SUPPRIME(2);
```

Lorsque des rééquilibrages sont nécessaires, vous prendrez soin de faire figurer les arbres (non-AVL) intermédiaires avant rotation.

Question 3.[2pt] Ecrire en pseudo-code une procédure $SUCCESSEUR(A, i)$ qui prend en entrée un arbre AVL A contenant la clé i et qui calcule la plus petite clé de A, plus grande que i, si elle existe.

Question 4.[1pt] On considère le graphe orienté suivant.



Dans les parcours demandés ci-dessous, une itération sur les sommets du graphe devra se faire par ordre alphabétique croissant, idem pour une itération sur les adjacents d'un sommet.

4.1.[1pt] Effectuer un parcours en largeur de ce graphe en faisant apparaître, pour chaque sommet, l'état de la file d'attente au moment où ce sommet est extrait de la file.

4.2.[1pt] Effectuer un parcours en profondeur (procédure récursive) de ce graphe en faisant apparaître les différentes catégories d'arcs (liaison, arrière, avant, transverse). Par convention les adjacents d'un sommet seront parcourus par ordre alphabétique croissant. Mettre en évidence la forêt correspondant à ce parcours.

4.3.[2pt] Appliquer l'algorithme de Kosaraju pour le calcul de composantes fortement connexes sur ce même exemple. Mettre en évidence la forêt correspondant au 2ème parcours effectué pendant cet algorithme.

4.4.[2pt] Appliquer l'algorithme de Tarjan pour le calcul de composantes fortement connexes sur ce même exemple.

Question 5. Soit G un graphe orienté.

5.1.[1pt] Soit C une composante fortement connexe de G et i, j deux sommets de C. Montrer que tous les chemins de i à j passent exclusivement par des sommets de C.

5.2.[2pt] On considère un parcours en profondeur de G. Prouver que toute composante fortement connexe de G est incluse dans un arbre de sa forêt de parcours.

5.3.[1pt] Pouvez-vous raffiner cette propriété ?

Question 6. Soit $G = (S, A)$ un graphe orienté, avec $S = [0, n-1]$, n un entier fixé. On veut calculer la fermeture transitive de G sous la forme d'une matrice A^+ de dimension $n \times n$, à valeur dans les booléens, telle que pour tout sommet x, y, il existe un chemin de x à y dans G si et seulement $A^+[x, y] =$ vraie.

6.1.[1pt] Expliquer pourquoi le nombre d'arcs de liaison (dans un parcours en profondeur quelconque) de G est un $\mathcal{O}(n)$.

6.2.[2pt] Proposer une équation récursive qui exprime, pour chaque sommet i, le contenu de la ligne $A^+[i, \cdot]$ en fonctions des lignes $\{A^+[j, \cdot] \mid (i, j) \in A\}$.

6.2.[3pt] On suppose G **acyclique**. En adaptant un parcours en profondeur récursif, écrire un algorithme de calcul de A^+ avec une complexité temporelle en $\mathcal{O}(n^2 + n \cdot m)$, où m est le nombre d'arcs transverses dans G. Justifier la correction de cet algorithme et sa complexité.