Nu-Huyen-Trang PHAM
Van-Triet NGUYEN
Université de Technologie de Compiègne
7 juin 2019

Rapport TP4 - NF16

Arbre binaire de recherche

A. Partie A

1. **Fonction creerSommet():**

   La fonction ne contient que les instructions simples donc la complexité est O(1)

2. **Fonction insererSommet():**

   On peut voir qu'il y a un boucle while dans cette fonction. Pour insérer un nouveau sommet, il faut parcourir m éléments de l'arbre. Donc la complexité est O(m) avec $m<=2^h$, h l'hauteur de l'arbre car on n'est pas sure que l'arbre est équilibre.

3. **Fonction afficher():**

   Cette fonction a 2 appels de récursivité avec la complexité est O(gauche) et O(droite) avec gauche le nombre d'éléments de sous arbre gauche et droite le nombre d'éléments de sous arbre droite. Donc la complexité de cette fonction est O(n) avec n le nombre d'éléments de l'arbre binaire de recherche (n = gauche + droite).

4. **Fonction rechercheSommet():**

   Dans cette fonction, on parcourt l'arbre jusqu'à ce qu'on trouve l'élément de paramètre. Donc la complexité est O(m) avec m<=n, n le nombre d'éléments de l'arbre.

5. **Fonction tailleArbre():**

   Même raisonnement et même complexité que la fonction afficher().

6. **Fonction deleteAbr():**

   Même raisonnement et même complexité que la fonction afficher().

B. Partie B

7. **Fonction creerSommetCompact():**

   La fonction ne contient que les instructions simples donc la complexité est O(1)

8. **Fonction insererElement():**

   On peut voir qu'il y a un boucle while dans cette fonction. Pour insérer un nouveau sommet, il faut parcourir m éléments de l'arbre. Donc la complexité est O(m) avec $m<=2^h$, h l'hauteur de l'arbre car on n'est pas sure que l'arbre est équilibre.

9. **Fonction afficherArbreCompact():**

   Cette fonction a 2 appels de récursivité avec la complexité est O(gauche) et O(droite) avec gauche le nombre d'éléments de sous arbre gauche et droite le nombre d'éléments de sous arbre droite. Donc la complexité de cette fonction est O(n) avec n le nombre d'éléments de l'arbre binaire de recherche (n = gauche + droite).

10. **Fonction rechercheElement():**

    Dans cette fonction, on parcourt l'arbre jusqu'à ce qu'on trouve l'élément de paramètre. Donc la complexité est O(m) avec m<=n, n le nombre d'éléments de l'arbre.

11. **Fonction tailleArbreCompact():**

    Même raisonnement et même complexité que la fonction afficherArbreCompact().

12. **Fonction deleteAbrCompact():**

    Même raisonnement et même complexité que la fonction afficherArbreCompact().

C. Fonctions supplémentaires

Nous avons créé 2 fonctions min() et max() pour retourner la valeur min, max pour fusionner l'intervalle si l'élément ajouté ne satisfait pas la condition.