TD n°3

Arbre Binaire de Recherche (suite et fin ?)

Exercice 1 [**Suppression dans un ABR**]

Nous avons observe lors du TD precedent, qu'il est possible de supprimer un n ud d'un arbre binaire de recherche. Le but de cet exercice est de concevoir et de comparer deux algorithmes de suppression.

Soit x le noeud que l'on cherche a supprimer. Il n'est pas complique de supprimer x s'il ne possede ni ls gauche, ni ls droit | et si x ne possede qu'un seul ls, il su t de le remplacer par son ls. Les deux algorithmes etudies dans cet exercice di erent dans la maniere dont il supprime un n ud possedant deux ls.

Le premier algorithme cherche a deplacer entierement le sous-arbre gauche de x dans le sous-arbre droit de x (ou inversement).

{ A quel(s) n ud(s) du sous-arbre droit de x est-il possible de rattacher le sous-arbre gauche de x ?
{ Dans le pire des cas quel est le rapport entre la hauteur de l'arbre avant et apres suppresssion ?
{ Ecrivez cet algorithme et calculez sa complexite en moyenne et dans le pire des cas.

Le second algorithme cherche a ne pas desequilibrer inutilement l'arbre apres suppression. Si l'on nomme T l'arbre original et T' celui obtebu apres suppresion, il est possible d'obtenir la propriete suivante :

$$hauteur(T) - 1 \leqslant hauteur(T') \leqslant hauteur(T)$$

{ Ecrivez un algorithme de suppression utilisant la fonction `Node Successeur(x : Node)` (ou la fonction `Node Predecesseur(x : Node)`).
{ Montrez que votre algorithme veri e la propriete ci-dessus.
{ Calculez sa complexite en moyenne et dans le pire des cas.

Exercice 2 [**Petite pause**]

Ecrivez un algorithme, qui donne le nombre de n ud d'un arbre T. Quelle est sa complexite, dans le pire des cas et en moyenne ?

Exercice 3 [**Insertion à la racine**]

Dans un arbre binaire de recherche, avec la methode d'insertion classique, toutes les nouvelles valeurs sont placees aux feuilles de l'arbre. Si l'on souhaite acceder a un n ud inserer recemment dans l'arbre, il faudra parcourir toute la hauteur de l'arbre. Dans certaines applications, on souhaite acceder plus frequemment aux derniers elements inseres. Il s'agit du principe LRU[1]. Dans ce cas particulier, il peut-être interessant d'inserer a la racine. En procedant de la sorte, les valeurs auxquelles on souhaite acceder le plus souvent ont plus de change d'être a une hauteur faible.

En considerant l'exemple de la gure 1 :

{ tracez le chemin qui va de la racine au n ud ou classiquement il faudrait inserer le n ud 33.
{ si on enleve les arêtes de ce chemin, quelles sont les caracteristiques des sous arbres restants ?
{ que peux-t'on dire de l'ordre dans lequel l'on rencontre sur ce chemin les n uds dont l'etiquette est plus petite que 33 ? Et de l'ordre de ceux dont l'etiquette est plus grande que 33 ?

[1] pour Least Recently Used !

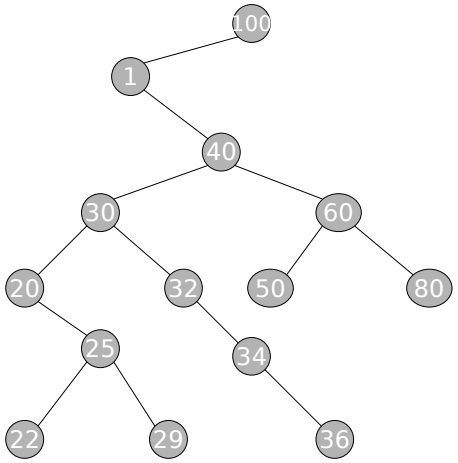

FIG. 1 { un arbre binaire de recherche

A partir des reponses aux questions precedentes, proposez un algorithme d'insertion a la racine dans un arbre binaire de recherche.

Indication : Il est possible d'utiliser deux listes de sous arbres, une pour les \plus grands" et une autre pour les \plus petits".

Evaluez la complexite de cet algorithme.

Exercice 4 [**Petite pause bis**]

Quelle pourrait être la complexite d'une fonction qui teste si un arbre binaire etiquette par des entiers est un arbre binaire de recherche ?

Exercice 5 [**Accès au k-ième élément**]

Etant donne un arbre binaire de recherche on souhaite connaître la valeur du $k^{\text{ième}}$-element.

1. Combien de temps cela prend-il de trouver le $k^{\text{ième}}$-element dans un tableau de n entiers tries ?
2. Proposez un methode simple qui utilise un parcours et une structure auxilliaire pour trouver le $k^{\text{-ième}}$ de l'arbre. Quelle est la complexite de cette methode ?
3. Proposez un methode qui utilise n'utilise pas de structure auxilliaire pour trouver le $k^{\text{-ième}}$ de l'arbre. Quelle est la complexite de cette methode ? (*Indication : Vous pourrez-vous aider de la fonction de nit a l'exercice 2*)
4. En acceptant de modi er legerement la structure des arbres binaires, pouvez-vous proposer une methode avec une meilleur complexite ? Vous veri rez que votre modi cation ne modi e pas la complexite des fonctions d'insertion et de suppression d'un n ud dans un ABR.