TD Algo2 – session 3 – Arbres binaires de recherche

11 avril 2025

Objectifs d'apprentissage :
- manipuler la structure d'arbre binaire de recherche et étudier les implications de la propriété d'arbre de recherche ;
- concevoir des algorithmes relatifs aux arbres binaires de recherche ;
- mettre en pratique des techniques de preuve pour la validité ou la complexité d'algorithmes.

Exercice 1 : Hauteurs différentes

Pour les valeurs $(1, 4, 5, 10, 16, 17, 21)$, construire des arbres binaires de recherche de hauteur 2, 3, 4, 5 et 6.

Exercice 2 : Énumération

Combien y a-t-il d'arbres binaires de recherche dont les valeurs sont $(3, 5, 8, 12)$?

Exercice 3 : Contre-exemple

Le professeur Szmoldu pense avoir découvert une remarquable propriété des arbres binaires de recherche. Supposer que la recherche d'une clé k dans un arbre binaire de recherche se termine sur une feuille. On considère trois ensembles : A, les clés situées à gauche du chemin de recherche ; B, celles situées sur le chemin de recherche ; et C, les clés situées à droite du chemin de recherche. Le professeur Szmoldu affirme que, étant donnés trois $a \in A, b \in B, c \in C$ quelconques, ils doivent satisfaire à $a \leq b \leq c$. Donner un contre-exemple qui est le plus petit possible.

Exercice 4 : Récursivité

Donner une version récursive de la procédure Arbre-Insérer.

Exercice 5 : Valeur inférieure

Proposer le pseudo-code d'un algorithme qui trouve le nœud qui possède la valeur la plus grande mais qui soit inférieure ou égale à celle qui est donnée à l'algorithme.

Exercice 6 : Rang

On souhaite un algorithme qui calcule le rang d'une valeur, c'est-à-dire, le nombre de nœuds dont les valeurs sont inférieures ou égales à celle qui est donnée à l'algorithme. Comment faudrait-il procéder pour avoir un algorithme en $O(h)$? Proposer en pseudo-code cet algorithme.

Exercice 7 : Arbre-Supprimer

Comment pourrait-on modifier l'algorithme de suppression pour alterner alternant la recherche/élévation du successeur avec la recherche/élévation d'un prédécesseur. Fournir le pseudo-code.

Exercice 8 : Prédécesseur

Fournir le pseudo-code permettant de rechercher le prédécesseur du nœud x.

Exercice 9 : Équilibre

On considère tous les nombres compris entre 1 et 1000. Donner deux ordres d'insertion de ces nombres dans un arbre binaire de recherche :
- l'un qui va donner un arbre complètement déséquilibré, c'est-à-dire de hauteur maximale ;
- l'autre qui va donner un arbre équilibré, c'est-à-dire le moins haut possible.

Écrire un algorithme qui les génère.

Exercice 10 : Ancêtre

Étant donné un arbre, comment vérifier si v est un ancêtre de w ?

Exercice 11 : Tri arborescent

On peut trier un ensemble donné de n nombres en commençant par construire un arbre binaire de recherche contenant ces nombres (en répétant Arbre-Insérer pour insérer les nombres un à un), puis en imprimant les nombres via un parcours infixe de l'arbre. Analyser les temps d'exécution de cet algorithme de tri, dans le pire et dans le meilleur des cas ?