Fiche d'exercices *III*: Algorithmes de Recherche

Exercice 1 (Recherche dichotomique et p-adique) *Écrire un algorithme itératif pour la recherche dichotomique dans un tableau trié. Généraliser cet algorithme à la recherche p-adique. Étudier la complexité.*

Exercice 2 (Recherche autoadaptative) *Donner un algorithme de recherche autoadaptative pour une répartition uniforme (exemple: nombre de $[n] = \{1, ..., n\}$). Étudier la complexité de votre algorithme sous cette hypothése.*

Exercice 3 (Arbres ordonnés) *Soient les 4 éléments ordonnés suivants: $a < b < c < d$. Donner tous les arbres de recherche strictement binaire ordonné.*

Exercice 4 (Recherche itérative) *Donner un algorithme itératif de recherche dans un arbre binaire ordonné. Étudier sa complexité.*

Exercice 5 (Propriété sur les $BB[\alpha]$) *Démontrer que pour tout $\alpha \in (\frac{1}{3}, \frac{1}{2})$ si un arbre A est $BB[\alpha]$ alors il est $BB[\frac{1}{2}]$ et réciproquement.*

Exercice 6 (Suite de Fibonacci) *La suite des arbres binaires de Fibonacci est définie par: F_0 est l'arbre vide, F_1 est un atome et $F_n =< F_{n-2}, F_{n-1} >$ pour tout $n > 1$.*

- *Donner un algorithme qui teste si un arbre est de Fibonacci ou non.*
- *Donner une fonction calculant F_n.*
- *Montrer que les arbres de Fibonacci sont tous dans $BB[\frac{1}{3}]$.*

Exercice 7 (Poids/facteur d'équilibre) *Soit $A =< A_g, A_d >$ un arbre strictement binaire et ρ son facteur d'équilibre. Prouver que $\rho = \frac{Poids_{(A_g)+1}}{Poids_{(A)+1}}$ où le poids d'un arbre est le nombre de nœuds internes le composant.*

Exercice 8 (Facteur d'équilibre) *Soit $A = c_1[A_1, c_2[A_2, A_3]]$ et $B = Rotation(A) = c_2[c_1[A_1, A_2]A_3]$. Soit $\beta_1 = \rho(A)$ et $\beta_2 = \rho(c2[A_2, A_3])$. Montrer que $\rho(B) = \beta_1 + (1 - \beta_1)\beta_2$ et $\rho(c_1[A_1, A_2]) = \frac{\beta_1}{\beta_1 + (1-\beta_1)\beta_2}$.*

Exercice 9 (Opérations sur les $BB[\alpha]$) *Écrire les algorithmes de recherche, insertion/suppression sur les arbres $BB[\alpha]$.*

Exercice 10 (Arbre 2-3) *Les arbres 2-3 sont définis comme suit: l'arbre vide ou un atome sont des arbres 2-3 et tout arbre possédant les deux propriétés suivantes est un 2-3 arbre:*

1. *Tout nœud interne a 2 ou 3 fils.*
2. *Tout chemin de la racine à une feuille a une longueur fixe.*

On peut représenter une table par des arbres 2-3 comme suit: on place les éléments de la table aux feuilles en les ordonnant de gauche à droite et les clefs sont placés dans les nœuds internes. À un nœud interne, on associe au plus deux clefs: la plus petite clef des descendants du second fils et la plus petite clef des descendants du troisième fils si celui-ci existe.
Montrer que la longueur d'un chemin de la racine à une feuille dans un arbre 2-3 de recherche codant une table est en $\Theta(\log n)$. Donner les algorithmes de recherche, insertion, suppression dans les arbres 2-3.

Exercice 11 (Hachage directe Bell&Kaman) *On rappelle la définition de la suite $\{p_i\}_i$:*

$$p_1 = h_1(e)$$
$$p_i = ((p_{i-1} - h_2(e)) \bmod M) + 1$$

avec M tels que M et $M-2$ soient premiers. On a $p_1 = v(e) \bmod M + 1$ et $h_2(e) = (v(e) \bmod (M-2)) + 1$.

1. *Montrer que pour tout n, on a:*

$$p_n = (h_1(e) - (n-1)h_2(e) + (n-1)) \bmod M + 1$$

2. *En déduire que la suite $\{p_i\}_i$ balaie bien le tableau sans répétition.*

Exercice 12 (Hachage) *Soit h une fonction de hachage dans $[m]$ et $\mathcal{E}$ un ensemble de n éléments. Calculer la probabilité que h soit surjective.*

Exercice 13 (Hachage quadratique) *On utilise souvent la technique de hachage quadratique (hachage avec résolution directe des collisions). On définit la suite des p_i comme suit:*

$$p_i = (h(e) + (i-1)^2) \bmod M$$

où M est un nombre premier et h est une fonction de hachage uniforme dans $\{0, ..., M-1\}$. Montrer que la suite de p_i balaie au moins la moitié du tableau.