TP4 et TP5 : ABR optimaux

Denis Trystram
ENSIMAG 2A Alternants 2020

Préliminaires

Le sujet proposé ici traite d'une amélioration des *arbres binaires de recherche.*

L'idée de base proposée ici est que si l'on connait combien de fois les objets sont accédés, on peut construire un arbre *optimal*, c'est-à-dire, **un arbre où le coût moyen de recherche d'un élément est minimum**. Même si l'on dispose seulement d'une estimation des coûts, les consultations peuvent être considérablement réduites en moyenne.

Par exemple, si l'on considère un ABR sur un dictionnaire de mots en français utilisé comme correcteur orthographique, on peut organiser l'arbre en tenant compte de la fréquence d'apparition des mots dans un texte, en plaçant les mots usuels comme **le** ou **un** près de la racine, et des mots plus rares comme **procrastination** ou **colophon** près des feuilles.

Exemple

A titre d'exemple, on considère 6 mots (indexés par i) dont les probabilités d'apparition dans un texte (notées p_i) sont respectivement :

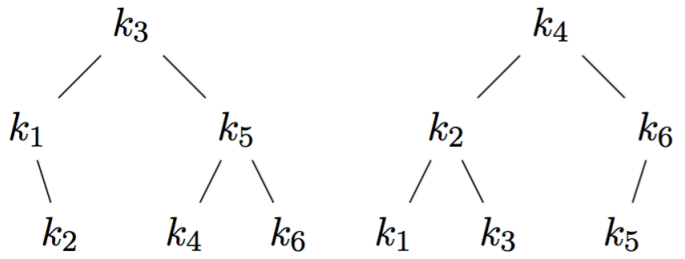
$p_1 = 0.1, p_2 = 0.15, p_3 = 0.45, p_4 = 0.05, p_5 = 0.2$ et $p_6 = 0.05$.

Les n mots sont associés à des clés distinctes k_i triées (de sorte que $k_1 < k_2 < ... < k_n$). Si l'on reprend l'exemple précédent avec $n = 6$, les deux ABR de la figure ci-dessous sont possibles.

On suppose que le coût de la recherche est déterminé par le nombre de comparaisons effectuées. Par exemple pour l'arbre T1, le coût pour rechercher le mot dont la clé est k_3 est 1, le coût pour k_4 est égal à 3, etc.. Sans informations supplémentaires, on ne peut pas dire lequel de ces deux arbres est le *meilleur* pour cette séquence de clés car tout dépend de la consultation... Nous allons donc utiliser les probabilités p_i. Le coût de la recherche de la clé k_i dans T est noté $c(k_i, T)$ et on définit ainsi le coût moyen de recherche dans l'arbre T de la façon suivante :

$c(T) = \Sigma_{i=1}^{n} p_i.c(k_i, T)$

L'objectif est de **déterminer le meilleur ABR** (celui qui minimise le coût moyen de la recherche), puis le **construire**. Nous le noterons *ABRopt.*



(a) Tree T_1 (b) Tree T_2

Le travail demandé est comporte deux parties.

1. Analyse théorique du problème, avec rédaction d'un petit rapport intermédiaire partiel (remis le vendredi soir 16 oct avant minuit). Les questions sont destinées à vous aider à rentrer dans le problème pas à pas...

2. Implémentation et test(s), rendus mardi 20 minuit.

Question 1. Analyse préliminaire.
Calculez les coûts des arbres T1 et T2, quel est le meilleur ?

Question 2.
Dans toute la suite, nous allons nous restreindre aux arbres dont tous les sous-arbres sont constitués de clés consécutives (ce qui est le cas dans les deux exemples précédents...).

En d'autres termes, si T' est un sous-arbre de T, il contient des clés $k_i, k_{i+1}, ..., k_j$ pour $1 \leq i \leq j \leq n$.

Montrez que si T est un ABRopt, alors, tout sous-arbre T' est un ABRopt pour les clés $k_i, ..., k_j$.

On note $\omega_{i,j} = \Sigma_{m=i}^{j} p_m$ et $c_{i,j}$ le coût d'un ABRopt restreint aux clés $k_i, ..., k_j$.

Question 3.
On considère un ABRopt de racine r (pour $i \leq r \leq j$), montrez que :

$$c_{i,j} = c_{i,r-1} + c_{r+1,j} + \omega_{i,j}$$

Question 4.
Donnez une relation de récurrence pour calculer $c_{i,j}$.

Question 5.

Ecrivez un algorithme ComputeW pour calculer **toutes** les valeurs $\omega_{i,j}$ et les stocker dans un tableau.

Evaluez son coût.

Question 6.

En supposant connu l'ABR, écrire l'algorithme récursif pour calculer le coût $c_{1,n}$ et calculer la complexité.