Arbres binaires de recherche optimaux

Manon Ruffini

Étape 1

Présentation du problème

On a une séquence $K = (k_1, \dots, k_n)$ de n clefs triées. Pour chaque clef k_i, on a la probabilité p_i qu'une recherche concerne k_i. On se donne aussi $n+1$ clefs factices $d_0, \dots, d_n$ qui représentent les valeurs qui ne sont pas dans K (d_0 représente les valeurs $< k_0$, d_n représente les valeurs $> k_n$, et $\forall i \in [|1, n-1|]$, d_i représente les valeurs strictement comprises entre k_i et k_{i+1}). Pour chaque clef factice d_i, on a la probabilité q_i qu'une recherche concerne d_i.
On veut construire un arbre binaire de recherche T pour les clefs de K. Chaque clef k_i sera un nœud interne et chaque clef factice d_i sera une feuille de l'arbre.
Comme une recherche soit réussit, soit échoue, on a :

$$\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$$

Pour l'arbre binaire de recherche T, on peut calculer le coût espéré d'une recherche [1] :

$$\begin{aligned}\mathbb{E}[\text{coût de recherche dans } T] &= \sum_{i=1}^{n} (\text{Prof}_T(k_i) + 1) \cdot p_i + \sum_{i=0}^{n} (\text{Prof}_T(d_i) + 1) \cdot q_i \\ &= 1 + \sum_{i=1}^{n} \text{Prof}_T(k_i) \cdot p_i + \sum_{i=0}^{n} \text{Prof}_T(d_i) \cdot q_i\end{aligned}$$

Pour un ensemble donné de probabilités , on veut calculer l'arbre de recherche comme décrit précédemment pour lequel le coût moyen d'une recherche est optimal. On appellera un tel arbre : arbre de recherche optimal (ABRO).
La vérification exhaustive de toutes les possibilités ne fournit pas un algorithme efficace (il y aurait un nombre exponentiel de possibilités).

On va résoudre ce problème grâce à un algorithme de **programmation dynamique**.

Étape 2

Structure d'un arbre de recherche optimal.

On remarque qu'un sous-arbre d'un arbre binaire de recherche doit contenir des clefs appartenant à une plage contigüe $k_i, \dots, k_j$ [2], avec $1 \leqslant j \leqslant n$. En outre, un sous-arbre qui contient les noeuds $k_i, \dots, k_j$ doit aussi contenir les feuille $d_{i-1}, \dots, d_j$.
De plus, si un ABRO T a un sous-arbre T', T' est forcément optimal [3].
Si $T_{i,j}$ est un ABRO contenant les clefs $k_i \dots k_j$, et si k_r est sa racine, on remarque que son fils gauche contient les clefs $k_i \dots k_{r-1}$ et son fils droit $k_{r+1} \dots k_j$. On remarque également que si $r = i$, le fils gauche (clefs $k_i \dots k_{i-1}$) correspond à la feuille d_{i-1} et que si $r = j$, le fils droit (clefs $k_{j+1} \dots k_j$) correspond à la feuille d_j.

1. La profondeur de la racine de l'arbre est 0, et une recherche de la clef racine se fait en temps 1. Une recherche d'une clef de profondeur 1 a un coût de 2, etc.
2. Induction : si un arbre contient une place contigüe, ses fils droit et gauche aussi (ABR)
3. Sinon on remplace T' par un ABRO qui contient les mêmes clefs ; ce qui contredit l'optimalité de T.

Étape 3

Du coup, on peut décrire les sous-problèmes et la relation de récurrence.

Pour $i \geqslant 1$ et $i-1 \leqslant j \leqslant n$, soit $e[i,j]$ le coût espéré de recherche dans un ABRO contenant les clefs $k_i, \ldots, k_j$.

— Pour $j = i-1$: $e[i, i-1] = q_{i-1}$

— Pour $j \leqslant i$:

D'abord, on remarque que quand on considère notre sous-arbre comme sous-arbre d'un noeud, on augmente la profondeur de chaque noeud de 1, et donc le coût de recherche moyen de :

$$w(i,j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l$$

Ainsi, on obtient la relation suivante :

$$e[i,j] = \min_{i \leqslant r \leqslant j} \{p_r + (e[i, r-1] + w(i, r-1)) + (e[r+1, j] + w(r+1, j))\}$$

Or, comme

$$w(i,j) = w(i, r-1) + p_r + w(r+1, j)$$

on obtient :

$$e[i,j] = \begin{cases} q_{i-1} & \text{si } j = i-1 \\ \min\limits_{i \leqslant r \leqslant j} \{e[i, r-1] + e[r+1, j] + w(i,j)\} & \text{si } i \leqslant j \end{cases}$$

On doit également considérer la quantité $e[n+1, n]$, qui correspond au coût moyen d'une recherche dans le sous-arbre contenant uniquement la feuille d_n.

Pour gérer la structure des sous-arbres de recherche optimaux, on définit, pour $1 \leqslant i \leqslant j \leqslant n$, $racine[i,j]$ qui correspond à l'indice r pour lequel k_r est la racine d'un ABRO contenant les clefs $k_i \ldots k_j$.

Enfin, on remarque que :

$$w(i, i-1) = q_{i-1} \text{ et } \forall j \geqslant i, w(i,j) = w(i, j-1) + p_j + q_j$$

Étape 4

Algorithme et complexité

Algorithme 1 : ABRO

Entrées : Les probabilités $p_1 \dots p_n$ et $q_0 \dots q_n$, et la taille n
Résultat : Les tableaux e et $racine$
Soient $e[1 \dots n+1, \dots n]$, $w[1 \dots n, 0 \dots n]$ et $racine[1 \dots n, 1 \dots n]$ des tableaux;
pour $1 \leqslant i \leqslant n+1$ **faire**
    $e[i, i-1] \leftarrow q_{i-1}$;
    $w[i, i-1] \leftarrow q_{i-1}$;
fin
pour $1 \leqslant l \leqslant n$; `/* l = le nombre de clefs dans le sous-arbre */`
faire
    pour $1 \leqslant i \leqslant n-l+1$ **faire**
        $j = i + l - 1$;
        $e[i, j] \leftarrow \infty$;
        $w[i, j] \leftarrow w[i, j-1] + p_j + q_j$;
        pour $i \leqslant r \leqslant j$ **faire**
            $t = e[i, r-1] + e[r+1, j] + w[i, j]$;
            si $t < e[i, j]$ **alors**
                $e[i, j] \leftarrow t$;
                $racine[i, j] \leftarrow r$
        fin
    fin
fin
retourner *e et racine*

Il y a trois boucles imbriquées. On a donc une complexité en $O(n^3)$.

Références

[1] Thomas H. Cormen, *Algorithmique*. Dunod, 3è édition, 2010.