Arbres binaires de recherche optimaux

Pierron Théo

13 avril 2014

On dispose de $k_1 < \ldots < k_n$ clés qu'on veut placer dans un ABR. Soit X la variable aléatoire représentant quelle clé on va chercher. On connaît les probabilités suivantes :

$$p_i := \mathbb{P}(X = k_i) \text{ et } q_i := \mathbb{P}(X \in]k_i, k_{i+1}[)$$

avec par convention $k_0 = -\infty$ et $k_{n+1} = \infty$. On représente la recherche d'une clé absente par la recherche d'une clé factice notée d_i qui devra nécessairement être une feuille. On a donc $q_i = \mathbb{P}(X = d_i)$ et

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

Le coût de la recherche d'une clé k dans T est 1 plus la profondeur de k dans T. Les arbres qui minimisent le coût moyen de recherche, à savoir

$$\sum_{i=1}^{n} p_i(1 + \text{Prof}(k_i)) + \sum_{i=0}^{n} q_i(1 + \text{Prof}(d_i))$$

sont appelés ABRO.

Si T est un ABRO de racine k_r, nécessairement les sous-arbres gauche et droit T_1 et T_2 doivent contenir les clés $k_1, \ldots, k_{r-1}, d_0, \ldots, d_{r-1}$ et $k_{r+1}, \ldots, k_n, d_r, \ldots, d_n$ et minimiser le coût induit, c'est-à-dire minimiser le coût d'une recherche avec les probabilités p'_i et q'_i renormalisées, ce qui est équivalent à minimiser

$$\sum_{i=1}^{r-1} p_i(1 + \text{Prof}_{T_1}(k_i)) + \sum_{i=0}^{r-1} q_i(1 + \text{Prof}_{T_1}(d_i))$$

En effet, si T_1 et T_2 n'étaient pas optimaux, on pourrait les remplacer par des arbres de moindre coût, et l'arbre T' obtenu aurait un coût inférieur à celui de T. Ceci contredit l'optimalité de T. Donc tout sous-arbre d'un ABRO doit contenir une plage contiguë de clés $k_i, \ldots, k_j$ et minimiser

$$\sum_{l=i}^{j} p_l(1 + \text{Prof}(k_l)) + \sum_{l=i-1}^{j} q_l(1 + \text{Prof}(d_l))$$

On note $c_{i,j}$ le coût d'un ABRO $T_{i,j}$ contenant $k_i, \ldots, k_j$.

Alors, si $j = i - 1$, $c_{i,j} := q_{i-1}$ car l'ABRO ne contient que d_i.

Sinon, en notant k_r sa racine, son coût est

$$\begin{aligned}
&\sum_{l=i}^{j} p_l(1+\mathrm{Prof}_{T_{i,j}}(k_l)) + \sum_{l=i-1}^{j} q_l(1+\mathrm{Prof}_{T_{i,j}}(d_l)) \\
&= \sum_{l=i}^{r-1} p_l(1+\mathrm{Prof}_{T_{i,j}}(k_l)) + p_r + \sum_{l=r+1}^{j} p_l(1+\mathrm{Prof}_{T_{i,j}}(k_l)) \\
&+ \sum_{l=i}^{r-1} q_l(1+\mathrm{Prof}_{T_{i,j}}(d_l)) + \sum_{l=r}^{j} q_l(1+\mathrm{Prof}_{T_{i,j}}(d_l)) \\
&= \sum_{l=i}^{r-1} p_l(2+\mathrm{Prof}_{T_{i,r-1}}(k_l)) + p_r + \sum_{l=r+1}^{j} p_l(2+\mathrm{Prof}_{T_{i,r-1}}(k_l)) \\
&+ \sum_{l=i}^{r-1} q_l(2+\mathrm{Prof}_{T_{r+1,j}}(d_l)) + \sum_{l=r}^{j} q_l(2+\mathrm{Prof}_{T_{r+1,j}}(d_l)) \\
&= \underbrace{\sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l}_{=w_{i,j}} + c_{i,r-1} + c_{r+1,j}
\end{aligned}$$

On a donc la relation de récurrence suivante :

$$c_{i,j} = \begin{cases} q_{i-1} & \text{si } i = j+1 \\ \min_{i \leqslant r \leqslant j}(c_{i,r-1} + c_{r+1,j}) + w_{i,j} & \text{sinon} \end{cases}$$

Qui conduit naturellement à l'algorithme de programmation dynamique :

Algorithme 1: ABRO(p,q)

Entrées : Les tableaux de probabilités p_i et q_i
Sorties : Le tableau des coûts c et un tableau rac tel que $rac[i,j]$ est la racine de $T_{i,j}$

```
1  pour i = 1...n+1 faire
2      c[i, i-1] ← q_{i-1}
3      w[i, i-1] ← q_{i-1}
4  pour d = 1...n faire
5      pour i = 1...n-d+1 faire
6          j ← i + l - 1
7          c[i, j] ← ∞
8          w[i, j] ← w[i, j-1] + p_j + q_j
9          pour r = i...j faire
10             t ← c[i, r-1] + c[r+1, j] + w[i, j] si t < c[i, j] alors
11                 c[i, j] ← t
12                 rac[i, j] ← r
13 retourner c, rac
```

Le coût recherché est $c[1,n]$. On peut alors reconstruire l'ABRO grâce au tableau r. On a un algorithme en $O(n^3)$.