Synthèse du cours 6 : programmation dynamique (3)

7 novembre 2023

François Laroussinie

1 Un jeu

On considère un jeu à deux joueurs. Au début du jeu, on dispose une suite de n cartes $c_1,\ldots,\ c_n$ sur une table. Chaque carte c_i a une valeur $v_i \in \mathbf{R}$. A chaque tour, le joueur 1 choisit une des deux cartes situées à une extrémité (gauche ou droite) de la rangée. Le joueur 2 fait ensuite de même. Le jeu s'arrête lorsque toutes les cartes ont été prises par les deux joueurs. Le gagnant est celui dont la somme de ses cartes est la plus grande.

A chaque tour, la position du jeu est définie par une paire (une *configuration*) (i,j) où $1 \leq i \leq j \leq n$: la carte la plus à gauche est c_i et celle la plus à droite est c_j. La configuration initiale est $(1,n)$. L'objectif est de calculer la meilleure stratégie possible pour le joueur 1.

On peut construire une table $\mathsf{TV}[-,-]$ de dimension $n \times n$ telle que pour toute configuration (i,j) $\mathsf{TV}[i,j]$ contient la valeur optimale que peut **assurer** le joueur 1 lorsqu'il a le trait (NB : « assurer » signifie ici que c'est une valeur que le joueur 1 peut obtenir même si le joueur 2 joue parfaitement :

$$\mathsf{TV}[i,j] \;=\; \begin{cases} 0 & \text{si } i > j \\ v_i & \text{si j=i} \\ \max(v_i, v_j) & \text{si j=i+1} \\ \max(v_i + \min(\mathsf{TV}[i+2,j], \mathsf{TV}[i+1,j-1]), & \\ \qquad v_j + \min(\mathsf{TV}[i,j-2], \mathsf{TV}[i+1,j-1])) & \text{sinon} \end{cases}$$

Il reste à énumérer correctement des paires i,j afin de remplir le tableau $\mathsf{TV}[i,j]\ldots$ A faire en exercice !

Une fois calculé $\mathsf{TV}[1,n]$, on peut répondre à la question « est-ce que le joueur 1 dispose d'une stratégie gagnante ou non ? » en comparant $\mathsf{TV}[1,n]$ et $(\sum_{i=1..n} c_i) - \mathsf{TV}[1,n]$.

2 ABR optimaux

On trouvera plus d'explications dans le livre *Algorithms*, de R. Sedgewick.

L'idée est de construire un ABR (Arbre Binaire de Recherche) **optimal** à partir d'un ensemble d'éléments dont on connait la fréquence (de recherche). Par optimal, on entend donc un ABR A tel que la somme $\sum_{x:\text{élt}}((\mathsf{prof}_A(x)+1)\cdot \mathsf{freq}(x))$ soit minimale (où $\mathsf{prof}_A(x)$ désigne la profondeur du noeud de A correspondant à l'élément x et $\mathsf{freq}(x)$ la fréquence de x). Notons que $(\mathsf{prof}_A(x)+1)$ correspond au coût (en nombre de comparaisons) de la recherche de l'élément x dans l'ABR.

On rappelle que dans un ABR, on stocke un élément dans chaque noeud de l'arbre et que l'on assure que si x' est dans le sous-arbre gauche de x alors $x' \leq x$, et si x' est dans le sous-arbre droit de x alors $x' > x$.

Le problème est donc le suivant :

Input : n valeurs $x_1, \cdots, x_n$ ordonnées ($x_1 \leq \cdots \leq x_n$) et n fréquences $f_1, \cdots, f_n$.
Output : Un ABR A contenant les valeurs x_i minimisant la somme $\sum_{x:\text{élt}}((\mathsf{prof}_A(x)+1)\cdot\mathsf{freq}(x))$.

Dans la suite, on appelle le coût de l'arbre A (noté $C(A)$) la somme $\sum_{x:\text{élt}}((\mathsf{prof}_A(x)+1)\cdot\mathsf{freq}(x))$. Pour calculer le coût minimal d'un ABR contenant les n valeurs x_is, on va calculer le coût minimal $d_{i,j}$ (avec $i \leq j$) d'un ABR contenant les éléments $x_i, x_{i+1}, \ldots, x_j$. Ensuite nous verrons comment en déduire un ABR ayant un coût égal à $d_{1,n}$.

La valeur de $d_{i,j}$ va dépendre du choix de l'élément positionné à la racine de l'ABR. Supposons que ce soit l'élément k, alors le meilleur ABR A de ce type sera obtenu en prenant un ABR A_1 optimal pour les valeurs $x_i, \ldots, x_{k-1}$ en sous-arbre gauche et un ABR optimal A_2 pour $x_{k+1} \ldots x_j$ en sous-arbre droit. Le coût de A peut alors s'exprimer ainsi :

$$C(A) = \Big(\sum_{i\leq e\leq k-1}((\mathsf{prof}_{A_1}(x_e)+2)\cdot\mathsf{freq}(x_e))\Big)+\mathsf{freq}(x_k)+\Big(\sum_{k+1\leq e\leq j}((\mathsf{prof}_{A_2}(x_e)+2)\cdot\mathsf{freq}(x_e))\Big)$$

$$\begin{aligned} C(A) &= \Big(\sum_{i\leq e\leq k-1}((\mathsf{prof}_{A_1}(x_e)+1)\cdot\mathsf{freq}(x_e))\Big)+\Big(\sum_{k+1\leq e\leq j}((\mathsf{prof}_{A_2}(x_e)+1)\cdot\mathsf{freq}(x_e))\Big)+ \\ &\quad +\Big(\Big(\sum_{i\leq e\leq j}\mathsf{freq}(x_e)\Big)\Big) \\ &= d_{i,k-1}+d_{k+1,j}+\Big(\sum_{i\leq e\leq j}\mathsf{freq}(x_e)\Big) \end{aligned}$$

On en déduit : $d_{i,j} = \sum_{i\leq e\leq j}\mathsf{freq}(x_e) + \min_{i\leq k\leq j}(d_{i,k-1}+d_{k+1,j})$ avec comme convention $d_{i,i} = \mathsf{freq}(x_i)$ et $d_{i,j} = 0$ si $i > j$. Ensuite il suffit d'énumérer correctement les indices... Voir l'algorithme ci dessous :

Procédure `ABRopt` (X, F)
begin
  `D` : matrice d'entiers de taille $|X| \times |X|$, initialisée avec ∞
  pour $i = 0 \ldots |X|$ **faire** `D`$[i,i] = F[i]$
  pour $d = 1 \ldots |X|-1$ **faire**
    pour $p = 0 \ldots |X|-d-1$ **faire**
      $i := p, \quad j := d+p$
      pour $k = i \ldots j$ **faire**
        `aux` $:= 0$
        si $i \leq k-1$ **alors** `aux`$+=$ `D`$[i][k-1]$
        si $k+1 \leq j$ **alors** `aux`$+=$ `D`$[k+1][j]$
        `D`$[i][j] := \min($`D`$[i][j], aux)$
      `D`$[i][j] :=$ `D`$[i][j] + \sum_{i\leq u\leq j} F[u]$
  return `D`$[0, |X|-1]$