DM d'algorithmique

October 16, 2018

1 Partie sur la Programmation Dynamique

1.1 Expliquez l'algorithme formellement en pseudo-code et prouver l'optimalité de la solution produite.

1.1.1 Pseudo code

```
function optimum_tree (alpha, beta):

    let R # the roots
    let P # the weighted paths
    let W # the summed weights
    let n = size(alpha)

    # Initialise values
    for i in 0..n:
        W[i, i] = P[i, i] = alpha[i] # summed paths and weights initialisation
        R[i, i] = i

    # Pre compute W
    for i in 0..n:
        for j in i+1..n:
            W[i, j] = W[i, j-1] + beta[j] + alpha[j]

    # Build R and P
    for subtreeLen in 2..n-1:
        # We build all the possible optimal trees of length subtreeLen
        for i in 0..n-subtreeLen:
            j = i + subtreeLen
```

```
            # find the best root for the subtree i..j
            # according to the article, we don't have to check all
            # the values from i to j but only a subset:
            for k in R[i, j-1]..R[i+1, j]:
                currentWeight = P[i, k+1] + P[k+1, j] + W[i, j]
                # Then update the current values of R and P if we have a better value
                update_min_weight_and_min_root(R, k, P, currentWeight)

    return R
```

1.1.2 Preuve d'optimalité

L'algorithme consiste en la construction de l'arbre en respectant la contrainte :

$$P_{ij} = \min_{R_{i,j-1} < k \leq R_{i+1,j}} (P_{i,k-1} + P_{k,j}) + W_{ij} \tag{1}$$

Où P, R et W représentent :

- P_{mn} le poids du sous-arbre optimal consituté des noeuds $A_m ... A_n$
- R_{mn} l'indice de sa racine
- W_{mn} la somme $\alpha_m, \beta_{m+1}, \alpha_{m+1}, ..., \alpha_n$

Montrons la construction d'un arbre respectant pour tout intervalle $A_i, A_{i+1} ... Aj$ la contrainte (1) produit un arbre optimal.

Tout d'abord : Si un arbre de noeuds $A_i ... A_j$ de racine k avec $i < k \leq j$ est optimal, alors des sous-arbres droit et gauche de cet arbre, constitués respectivement des noeuds $A_i ... A_{k-1}$ et $A_{k+1} ... A_j$ sont optimaux. En effet dans le cas contraire, si un des sous-arbres est sous-optimal, en choisissant à sa place un sous-arbre optimal, on améliore le poids de l'arbre $A_i ... A_j$ qui ne serait donc pas optimal.

On en déduit que pour construire un arbre optimal, il suffit de choisir sa racine telle que le poids de l'arbre complet constitué du sous-arbre optimal à gauche de la racine, et du sous-arbre optimal à droite, soit minimal, autrement dit, trouver R_{ij} tel que :

$$P_{i,R_{ij}-1} + P_{R_{ij},j} = \min_{i<k\leq j} (P_{i,k-1} + P_{k,j}) \tag{2}$$

D'autre part, d'après le corollaire de l'article, on sait qu'il existe une telle racine R_{ij} respectant la condition $R_{i,j-1} \leq R_{ij} \leq R_{i+1,j}$. On en déduit qu'il

suffit de chercher la racine seulement dans cet intervalle, à savoir trouver R_{ij} tel que :

$$P_{i,R_{ij}-1} + P_{R_{ij},j} = \min_{R_{i,j-1} \leq k \leq R_{i+1,j}} (P_{i,k-1} + P_{k,j}) \tag{3}$$

On en déduit que la procédure induite par (1) permet bien de construire un arbre optimal.

2 Partie sur les Flots

2.1 Montrez comment ce problème peut être réduit à un problème de flot.

2.1.1 Partie 1: Simplification et reformulation du problème.

Soit A une matrice à arrondir de dimensions $m \times n$. On peut poser A' et A'' deux matrices telles que :

$$A = A' + A''$$

Avec A' et A'' correspondant respectivement aux parties entières et parties fractionnaires de A, c'est-à-dire que $\forall (i,j) \in [\![1,m]\!] \times [\![1,n]\!]$:

$$a'_{i,j} = \lfloor a_{i,j} \rfloor$$

et

$$a''_{i,j} = \{a_{i,j}\}$$

Pour aboutir à B , une matrice arrondie de A, on cherche $B = B' + B''$ avec $B' = A'$ et B'' à déterminer. Pour cela, il suffit de déterminer si on prend la partie entière par défaut ou par excès de la partie fractionnaire, *ie* soit 0, soit 1.

En outre, on peut déjà remarquer que, pour $(i,j) \in [\![1,m]\!] \times [\![1,n]\!]$:

$$\lfloor a_{i,j} \rfloor = a_{i,j} \Longrightarrow \{a_{i,j}\} = 0 \ ie \ b''_{i,j} = 0$$

Donc si $\lfloor a_{i,j} \rfloor = a_{i,j}$, il n'y a pas besoin de determiner les valeurs de b'' pour ces couples (i,j).

Pour déterminer le reste de la matrice B'', on utilise la contrainte sur les sommes des lignes et des colonnes. En effet, en calculant la somme des lignes et des colonnes sur A', on peut connaitre la somme des lignes et des colonnes sur A'', et donc de B''.

Notation: Pour $(i,j) \in [\![1,m]\!] \times [\![1,n]\!]$, et A une matrice, on note respectivement $S_{R_i}(A)$ et $S_{C_j}(A)$ les sommes sur la ligne i et la colonne j de la matrice A. On remarque que, pour la matrice B'', si $S_{R_i}(B'') = 0$, alors la ligne i n'est constituée que de 0.

On arrive donc finalement à n'avoir qu'une matrice remplie de 0 et de 1 à determiner pour résoudre le problème, en connaissant les sommes sur les lignes et les colonnes. Cette matrice est du type:

$$B'' = \begin{bmatrix} \alpha_{11} & \dots & \alpha_{1n} \\ \vdots & \vdots & \vdots \\ \alpha_{m1} & \dots & \alpha_{mn} \end{bmatrix}$$

où

$$\alpha_{i,j} = \left\{ \begin{array}{c} 0 \\ ou \\ 1 \end{array} \right\}$$

2.1.2 Partie 2 : Modélisation du problème en problème de flots

Pour $(i,j) \in [\![1,m]\!] \times [\![1,n]\!]$, on définit r_i et c_j respectivement la ligne i et la colonne j de la matrice B'' .

On construit un graphe de source s, d'un sommet r_i par ligne vérifiant $S_{R_i}(B'') \neq 0$, d'un sommet c_j par colonne vérifiant $S_{C_j}(B'') \neq 0$, et d'un puits t. Il y a une arrête de capacité 1 entre la source et chacun des sommets r_i.

Pour construire les arrêtes entre les sommets r_i et c_j, on procède par itération sur les valeurs des sommes des lignes et colonnes de B''.

Explications : On définit

$$S_{max} = \max_{i,j}(S_{R_i}(B''), (S_{C_j}(B''))$$

et on considère tous les sommets r_i et c_j tels que

$$\left\{ \begin{array}{c} S_{R_i}(B'') = S_{max} \\ S_{C_j}(B'') = S_{max} \end{array} \right\} \tag{4}$$

Pour chacun des sommets r_i considérés, on place une arrête avec les tous les sommets c_j considérés. Le poids de chacune de ces arrêtes est de :

$$\left\{ \begin{array}{c} 1, \text{ si } \alpha_{i,j} \text{ n'a pas encore été déterminé} \\ 0, \text{ sinon} \end{array} \right\}$$

On peut alors choisir de déterminer la valeur de $\alpha_{i,j}$ pour un des couples (r_i, c_j) relié par une arrête de poids non nul.

On peut itérer le procéder jusqu'à se retrouver avec $S_{max} = 0$. En effet, après chaque itération, les sommes 'restantes' sur les lignes et les colones pour que la somme totale des lignes de A' et A'' soit égale à la somme des lignes de A diminue chacune de 1, vu qu'on rajoute un 1 sur la ligne i et la colonne j en même temps. Il reste maintenant à prouver qu'un solution existe toujours.

2.2 Montrez qu'il existe toujours une solution.

Remarquons dans un premier temps que

$$\text{Il existe une solution} \iff \left(\begin{array}{c} \text{A chaque étape de construction du graphe,} \\ \text{on peut trouver un chemin} \\ \text{de capacité non nulle entre} \\ \text{la source et le puits} \end{array} \right)$$

Soit d_i le nombre d'$\alpha_{i,j}$ non déterminés dans le ligne i. Ce nombre correspond également au nombre d'arrêtes totales provenant d'un sommet r_i lorsqu'il vérifie (4). Si nous pouvons montrer qu'à chaque étape, on a : $\forall i,\ S_{R_i}(B'') \leq d_i$. On pourra toujours trouver un chemin entre la source et le puits. On en vient donc à l'équivalence suivante:

$$\text{Il existe une solution} \iff \left(\begin{array}{c} \text{A chaque itération} \\ \forall i,\ S_{R_i}(B'') \leq d_i \end{array} \right) \tag{5}$$

Montrons qu'à chaque itération, on a toujours $S_{R_i}(B'') \leq d_i$ pour tout i. Pour construire la matrice A'', nous avions pris la partie fractionnaire des entrées de A. Or, on a:

$$\forall i \in [\![1, m]\!], \qquad S_{R_i}(A'') = \sum_j \{a''_{i,j}\} = \sum_{\{a''_{i,j}\} \neq 0} \{a''_{i,j}\}$$

Et

$$\forall (i,j) \in [\![1, m]\!] \times [\![1, n]\!], \qquad \{a''_{i,j}\} \leq 1$$

On obtient donc directement une majoration de la somme :

$$\forall i \in [\![1, m]\!], \qquad S_{R_i}(A'') \leq \Big(\sum_{\{a''_{i,j}\} \neq 0} 1 \Big) = d_i$$

En rappelant que $S_{R_i}(A'') = S_{R_i}(B'')$, on obtient bien

$$S_{R_i}(B'') \leq d_i$$

D'où, grâce à (5), l'existence d'une solution au problème quelque soit la matrice A de départ, pourvu qu'elle corresponde au caractéristiques décrites dans l'énoncé.