Structures de données et algorithmes

Répétition 8: Résolution de problèmes

Jean-Michel Begon

24 avril 2020

1 Programmation dynamique

1.1 B. Boigelot, 2009

On considère un terrain de jeu rectangulaire possédant $n \times m$ cases organisées en n lignes et m colonnes. Des sommes d'argent sont initialement placées sur ces cases. Un joueur traverse le terrain à partir du coin supérieur gauche jusqu'au coin inférieur droit. A chaque étape de ce trajet, le joueur collecte la somme d'argent placée sur la case où il se trouve, et a ensuite le choix de se déplacer d'une case soit vers la droite, soit vers le bas (les déplacements vers la gauche, vers le haut ou en diagonale sont interdits).

1. Proposez un algorithme permettant de calculer le gain maximum qu'un joueur peut atteindre en démarrant dans le coin supérieur gauche et en arrivant dans le coin inférieur droit. Pour ce faire :
   (a) Formulez récursivement $G(i, j)$ ($1 \leq i \leq n$, $1 \leq j \leq m$), le gain maximum obtenu en atteignant la case i, j à partir du coin supérieur gauche. Précisez le ou les éventuels cas de base.
   (b) Donnez le pseudo-code d'une fonction *efficace* permettant de calculer $G(n, m)$.
   (c) Adaptez votre solution pour renvoyer le parcours optimal.
2. Analyser la complexité en temps en espace de votre solution.
3. Proposez une approximation gloutonne et montrez à l'aide d'un contre exemple qu'elle ne peut pas être optimale.

1.2 Couture minimale

Mister B. dispose d'une pièce pavée de $w \times h$ dalles, comme celle de la figure 1. A l'exception de la première colonne, chaque dalle est occupée par une pile de dossiers, si bien que la porte de sortie n'est plus accessible. Peu enclin aux travaux domestiques, Mister B. aimerait récupérer l'usage de sa porte tout en minimisant son effort. Il souhaite donc enlever une seule pile par jour et avancer d'une rangée de case à chaque fois (il ne peut enlever que la pile de devant ou celles en diagonales d'une case déjà libérée). Partant du postulat que l'effort est proportionnel à la taille de la pile (et que celle-ci est connue), comment Mister B. peut-il minimiser son effort total ?

1. Formulez récursivement $C(r, c)$, le coût optimal pour atteindre la case (r, c).
2. Déduisez en le pseudo-code d'une fonction *efficace* permettant de résoudre le problème de la couture minimale. La fonction prend en entrée B, un tableau 2D représentant la pièce, ainsi que la ligne où se trouve la porte de sortie et renvoie un tableau contenant la ligne optimale pour chacune des colonnes ainsi que le coût de cette solution.
3. Quelle est la complexité de cette solution ?
4. Proposez une approximation gloutonne et montrez à l'aide d'un contre exemple qu'elle ne peut pas être optimale.

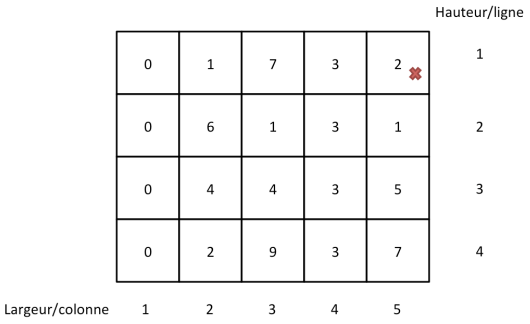

FIGURE 1 – Exemple de pièce de Mister B.

1.3 Plus longue sous-séquence palindromique (adapté de CLRS, 15-2)

On souhaite déterminer le plus grand palindrome qui existe au sein d'un mot. Par exemple, le mot `characters` contient un palindrome de taille 5 : `carac` (il ne s'agit pas d'une sous-séquence contigüe).

1. Quelle serait la complexité d'une recherche exhaustive de la solution ?
2. Proposez une solution par programmation dynamique pour ce problème :
   (a) Soit S, une chaîne de caractères. Formulez récursivement $L(i, j)$ la taille du plus grand palindrome qu'on puisse faire avec la chaîne $S[i\,..\,j]$:
   (b) Déduisez-en le pseudo-code d'un algorithme *efficace* pour déterminer le palindrome.
   (c) Quelle est la complexité de cette solution ?

1.4 Multiplications matricielles

Le coût de multiplication de deux matrices A et B de tailles respectives $m \times p$ et $p \times q$ est $\Theta(mpq)$. Soit le produit de N matrices $A_1 \times A_2 \times ... \times A_N$.

1. Illustrez par un exemple que l'ordre des multiplications a un impact sur le coût global.
2. Quelle serait la complexité d'une approche exhaustive pour déterminer l'ordre optimal ?
3. Proposez une solution par programmation dynamique pour résoudre ce problème.
   (a) Formulez récursivement $C(i, j)$, le coût optimal pour multiplier les matrices $A_i \times \cdots \times A_j$.
   (b) Déduisez-en le pseudo-code d'une fonction *efficace* qui détermine l'ordre optimal de multiplications. L'algorithme prend en entrée M, le tableau de longueur $n+1$ des tailles des matrices ($M[i] = m_i$).
   (c) Quelle est la complexité de cette solution en fonction n, le nombre de matrices ?

1.5 Le problème de la partition

On souhaite déterminer s'il est possible de partitionner un tableau d'entiers positifs A en deux sous-tableaux de somme identique.

1. Quelle serait la complexité d'une approche exhaustive résoudre ce problème ?
2. Proposez un algorithme par programmation dynamique pour ce problème. Quelle est sa complexité ?