Programmation avancée
Répétition 8: Résolution de problèmes

Jean-Michel Begon

7 décembre 2018

1 Merge Sort

Dessinez l'arbre de récursion du `MergeSort` pour le problème suivant :

$$A =< 5, 2, 4, 7, 1, 3, 2, 6 >$$

Pourquoi n'utilise-t-on pas la mémoïsation dans le cadre du `MergeSort` ?

2 Micmaths

Dans une vidéo intitulée "La puissance organisatrice du hasard - Micmaths" (2017), Mickaël Launay exprime qu'une bille qui se déplace au hasard mais strictement vers la droite sur un treillis similaire à celui de la figure 1 va adopter une trajectoire presque droite car il y a plus de qui mènent vers les positions centrales (par exemple la position $(3, 3)$, au départ de la position $(0, 0)$.

1. Soit $T(i, j)$ $(i, j = 1, \ldots, n)$ le nombre de chemins menant à la position (i, j) en partant de la position $(0, 0)$. Formulez récursivement T.
2. A l'aide de la mémoïsaiton, donnez le pseudo-code d'une fonction *efficace* pour calculer $T(i, j)$ $(i, j = 1, \ldots, n)$.

Remarque : si on suppose que tous les chemins sont équiprobables, on s'aperçoit que certains états "macroscopiques" (*i.e.* la position d'arrivée) sont, eux, plus probables. C'est la seconde loi de la thermodynamique.

3 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) Discutez des propriétés de sous-structure optimale et de chevauchement des sous-problèmes.

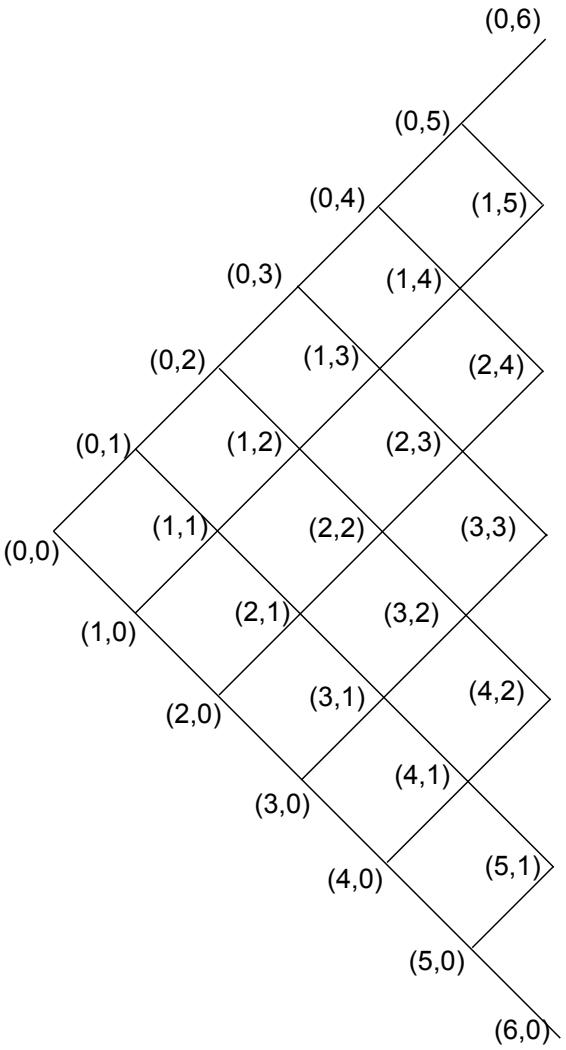


FIGURE 1 – Treillis de Launay

(b) Formulez récursivement $G(i, j)$ ($1 \leq n$, $1 \leq m$), le gain maximum obtenu en atteignant la case i, j à partir du coin supérieur gauche. Précisez le ou les éventuels cas de base.

(c) Donnez le pseudo-code d'une fonction *efficace* permettant de calculer $G(n, m)$.

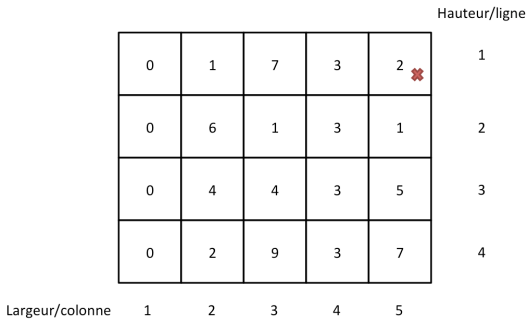
(d) Adaptez votre solution pour renvoyer le parcours optimal.

2. Analyzer la complexité en temps en espace de votre solution.
3. Dessinez le graphe des appels récursifs pour un problème de petite taille.

4 Couture minimale (*Checkerboard*)

Mister B. dispose d'une pièce pavée de $w \times h$ dalles. 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. Proposez une formulation récursive du problème.
2. Implémentez cette solution de manière efficace. 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 ?



5 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. Proposez une solution *brute-force* à ce problème.
2. Proposez une solution par programmation dynamique pour ce problème.
   (a) Formulez récursivement la taille de la plus longue sous séquence palindromique.
   (b) Ecrivez un pseudo-code pour déterminer ce palindrome.
   (c) Quelle est la complexité de cette solution ?

6 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 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. Proposez une solution *brute-force* pour calculer l'ordre optimal des multiplications.
3. Proposez une solution par programmation dynamique pour ce problème.
   (a) Proposez une formulation récursive du nombre minimum d'opérations à effectuer.
   (b) Déduisez-en le pseudo-code d'une fonction efficace qui optimise l'ordre des multiplications.
   (c) Quelle est la complexité de cette solution ?

7 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. Proposez une solution *brute-force* à ce problème. Quelle est sa complexité ?
2. Proposez un algorithme par programmation dynamique pour ce problème. Quelle est sa complexité ?

8 Le vrai parenthésage

Soit une expression booléenne composée des symboles `true`, `false`, `and` et `or`. Proposez un algorithme qui détermine le nombre de façons de parenthéser l'expression de sorte qu'elle soit évaluée à `true`.

Bonus

En bioinformatique, l'alignement de séquences est un outil qui permet de représenter deux ou plusieurs séquences d'ADN de manière à en faire ressortir les régions homologues. L'objectif de l'alignement est de disposer les composants de sorte à identifier les zones de concordance.

Par exemple, étant donné les deux séquences de bases :

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

Leur alignement consiste à apparier une base (un caractère) de l'une soit avec une base de l'autre, soit avec un trou :

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

Proposer un algorithme par programmation dynamique qui permet d'aligner deux séquences de bases en minimisant le nombre de trous. Etudier sa complexité.