

# Structures de données et algorithmes

## Répétition 7: Résolution de problèmes

Jean-Michel BEGON

27 avril 2018

### 1 100-chaîne

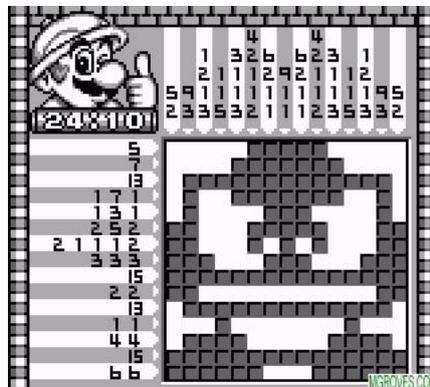
Soit la chaîne “123456789”. Proposez un algorithme qui énumère les différentes manières d’insérer des “+” et des “-” entre les *nombres* de manière à obtenir un total de 100.

Par exemple,  $123 + 45 - 67 + 8 - 9 = 100$ .

Quelle est la complexité de votre solution ?

### 2 Logimage (aussi appelé picross)

Proposez un algorithme par recherche exhaustive pour résoudre un logimage :



Cette solution est-elle envisageable en pratique ? Par exemple pour une grille  $15 \times 15$  ?

### 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. Pour ce faire :

- (a) Discutez des propriétés de sous-structure optimale et de chevauchement des sous-problèmes.
  - (b) Formulez récursivement  $G(i, j)$  ( $1 \leq i, 1 \leq j$ ), 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. Analysez 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 ?

					Hauteur/ligne	
	0	1	7	3	2 ✖	1
	0	6	1	3	1	2
	0	4	4	3	5	3
	0	2	9	3	7	4
Largeur/colonne	1	2	3	4	5	

## 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 contiguë).

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  $A_1 \times A_2 \times \dots \times A_N$ .

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 :

```
AGGCTATCACCTGACCTCCAGGCCGATGCC  
TAGCTATCACGACCGCGTTCGATTTGCCCGAC
```

Leur alignement consiste à appairer 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.