Programmation avancée

Répétition 7: Résolution de problèmes

Jean-Michel Begon

8 décembre 2017

1 100-chaîne

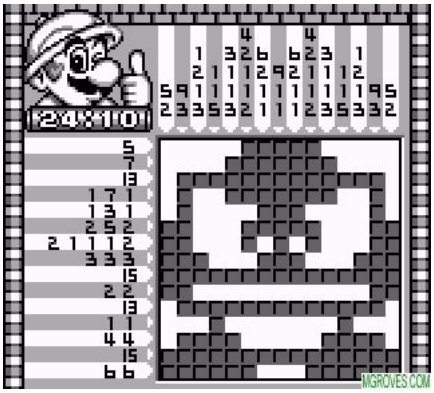
Soit la chaîne "123456789". Proposez un algorithme qui énumère les différentes manières d'insérer des "+" et des "-" entre les *nombres* de manière à obtenir un total de 100.

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 15×15 ?

3 Nombres de Catalan

Dans une répétition précédente, nous avions solutionné l'exercice :

> *Soit un ensemble de N valeurs entières distinctes. Ecrivez une fonction calculant le nombre d'arbres binaires de recherche distincts qu'il est possible de construire à partir de ces N valeurs ($N \geq 1$).*

par le pseudo-code suivant :

NbTree(N)
1 **if** $N \leq 1$
2 **return** 1
3 $Nb = 0$
4 **for** $i = 1$ **to** N
5 $Nb = Nb +$ NbTree$(i-1) *$ NbTree$(N-i)$
6 **return** Nb

dont la complexité est importante à cause des appels à des sous-problèmes déjà résolus.

1. Dessinez le graphe des sous-problèmes pour une taille de $N = 4$.
2. Utilisez la mémoïsation pour faire baisser la complexité de l'algorithme (donnez le pseudo-code d'une version ascendante et d'une version descendante).
3. Quelles sont ces complexités ?

4 Merge Sort

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

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

Pourquoi la mémoïsation ne permet-elle pas d'améliorer ce problème ?

5 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.
   (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. Dessinez le graphe des appels récursifs pour un problème de petite taille.

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.