Structures de données et algorithmes
Répétition 7: Résolution de problèmes

Jean-Michel Begon

28 avril 2017

100-chaîne

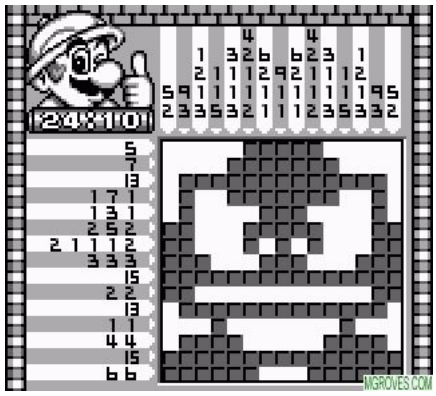
Soit la chaîne "123456789". Proposez un algorithme qui énumère les différentes manières d'insérer des "+" et des "-" entre les *nombres* de manière à obtenir un total de 100.

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

Quelle est la complexité de votre solution ?

Logimage/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 ?

Les 12 pièces

Soit le problème suivant :

> *"Vous disposez de 12 pièces dont une est fausse. Celle-ci possède un poids moindre. Comment déterminer la fausse pièce le plus rapidement possible à l'aide d'une balance à plateaux ?"*

Dans le cadre général où on dispose de N pièces :

(a) Proposez une approche par force brute pour résoudre ce problème.

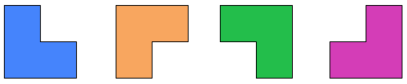
(b) Proposez une approche *divide-and-conquer* pour résoudre ce problème. Peut-on donner une borne sur le nombre de pesées ?

Distance minimum

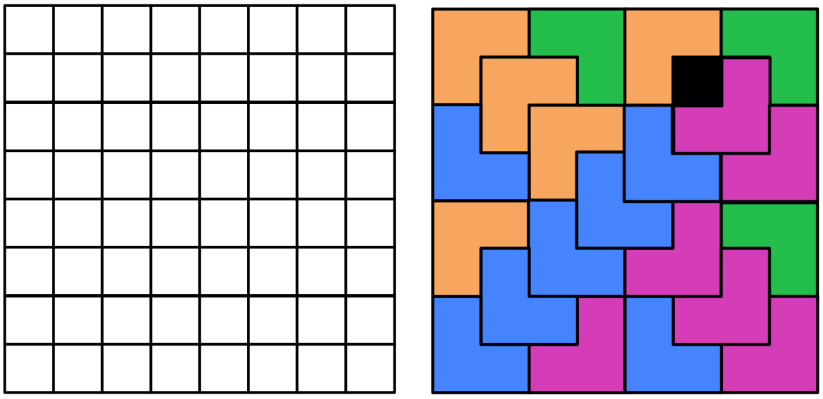
Soit un ensemble $P = \{p_1, p_2, ..., p_n\}$ de points ($n = 2^k$, $k > 0$) répartis dans un plan et tels que $p_i = (x_i, y_i)$. Proposez un algorithme diviser-pour-régner qui détermine la distance (euclidienne) entre les deux points les plus proches.

Carré de triminos

On souhaite recouvrir un carré de taille NxN ($N = 2^m, m > 0$) à l'aide des quatre triminos, à l'exception d'une case vide dont les coordonnées sont données.



Les triminos de base



Carré 8x8 vide et recouvert

Proposez un algorithme pour résoudre ce problème.

Multiplication matricielle

Proposez un algorithme diviser-pour-régner afin de multiplier deux matrices $n \times n$ ($n = 2^k, k \geq 0$). Quelle est la complexité de la solution ?

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 ≤ 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.

(a) Dessinez le graphe des sous-problèmes pour une taille de $N = 4$.

(b) Utilisez la mémoïsation pour faire baisser la complexité de l'algorithme (donnez le pseudo-code d'une version ascendante et d'une version descendante).

(c) Quelles sont ces complexités ?

Bonus

On considère une application biomédicale dont le but est de segmenter les cellules (c'est-à-dire de retrouver le polygone qui correspond à chaque cellule) sur des images de grandes dimensions (de l'ordre du gigapixel). Malheureusement, on est contraint de découper les images en tuiles rectangulaires (de mêmes dimensions) afin de permettre des temps de traitement raisonnables. [1]

La segmentation traite donc chacune des tuiles séparément et identifie dans chacune d'elles les polygones des cellules.

Pour garantir un bon fonctionnement de l'application, il faut cependant éviter qu'une cellule à cheval sur plusieurs tuiles soit considérée comme plusieurs cellules. Pour ce faire, on dispose d'un prédicat `touch` qui permet de vérifier si deux polygones se touchent ainsi qu'une fonction `merge` qui permet de fusionner ceux-ci.

Une approche *brute-force* en $\Theta(n^2)$ pour ce problème serait de passer chaque polygone avec tous les autres à la fonction `touch` et de fusionner ceux qui se touchent. Proposez une solution plus efficace.

1. Le même procédé est utilisé dans Google Maps, par exemple.