Programmation avancée

Répétition 8: Résolution de problèmes

Jean-Michel Begon

16 décembre 2016

Les portes russes

Dans la série "Zelda", Link est souvent confronté au jeu suivant : il doit choisir entre K portes. Derrière chaque porte se trouve un coffre contenant un nombre fixé de rubis, ainsi qu'un nouvel ensemble de K portes menant dans K nouvelles pièces. Le processus est répété N fois avant que Link ressorte avec son trésor. Est-ce que la programmation dynamique peut nous aider à résoudre ce problème si on connaît à l'avance le nombre de rubis par coffre ?

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.

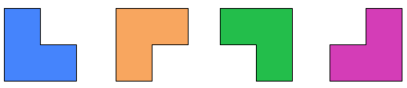
(b) Proposez une approche *divide-and-conquer* pour résoudre ce problème. Peut-on donner une borne sur le nombre de pesées ?

Distance minimum

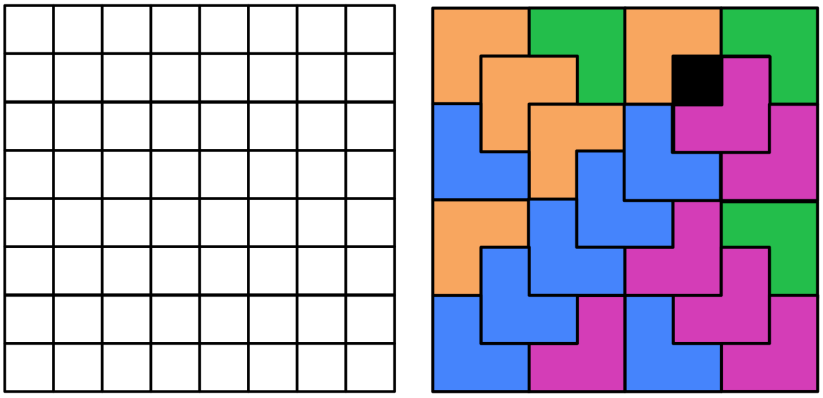
Soit un ensemble $P = \{p_1, p_2, ..., p_n\}$ de points ($n = 2^k$, $k > 0$) répartis dans un plan et tels que $p_i = (x_i, y_i)$. Proposez un algorithme diviser-pour-régner qui détermine la distance (euclidienne) entre les deux points les plus proches.

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 ?

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.