Programmation avancée

Répétition 6: Algorithme glouton et diviser-pour-régner

Jean-Michel Begon

5 décembre 2019

1 Approximation gloutonne

Proposez une approximation gloutonne, ainsi qu'un contre-exemple démontrant sa sous-optimalité pour le problème de la chasse au trésor (Boigelot).

2 La course de Julien (adapté de CLRS, 16.2-4)

Julien participe à une course à pieds qui consiste à relier une ville A à une ville B. Etant donné la contenance de son bidon, il sait qu'il peut parcourir m mètres sans tomber à cours d'eau. Il a également pu obtenir des organisateurs du marathon la liste des endroits où il pourra remplir son bidon et les distances entre ces endroits.

Julien aimerait minimiser le nombre d'arrêts qu'il devra effectuer lors de sa course pour remplir son bidon. Donnez une méthode efficace pour déterminer quels arrêts il devrait faire. Montrez que votre stratégie donne une solution optimale et discutez sa complexité.

3 L'ordonnancement

L'ordonnancement des tâches est un problème fréquent en informatique : on dispose d'un ensemble $S = \{s_1, s_2, ..., s_n\}$ de n tâches qu'on doit ordonner. A chaque tâche s_i est associé un temps d'exécution t_i. Une fois l'ordre des tâches fixé, on peut également associer à chaque tâche un temps de complétion (ou temps de réponse), c'est-à-dire, le temps qui s'est écoulé jusqu'à la fin de la tâche.

Par exemple, si on dispose des tâches s_1 et s_2 dont les temps d'exécution respectifs sont $t_1 = 3$ et $t_2 = 5$ et si on exécute les tâches dans l'ordre $< s_1, s_2 >$, alors les temps de complétion seront :

- $C_1 = t_1 = 3$
- $C_2 = C_1 + t_2 = 3 + 5 = 8$

En outre, la somme des temps de complétion sera $C_\Sigma = \sum_i C_i = C_1 + C_2 = 11$.

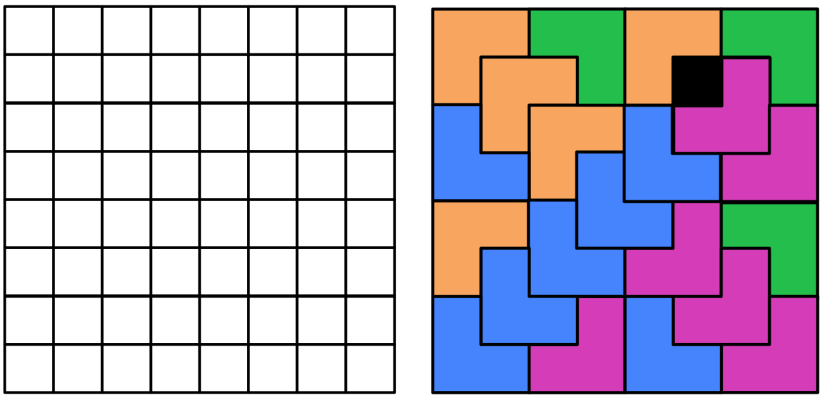
On souhaite disposer d'un algorithme d'ordonnancement des tâches qui minimise la somme des temps de complétion (C_Σ).

1. Quelle serait la complexité d'un algorithme *brute-force* pour résoudre ce problème ?
2. Est-il possible de faire mieux ?(*i.e.* le problème dispose-t-il de la propriété de sous-structure optimale ?)
3. Le cas échéant, proposez un algorithme efficace pour résoudre ce problème.

4 Carré de triminos

On souhaite recouvrir un carré de taille NxN ($N = 2^m, m \geq 0$) à l'aide des quatre types 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.

5 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 $p_i = (x_i, y_i)$. On cherche un algorithme qui détermine la distance (euclidienne) entre les deux points les plus proches.

1. Proposez une approche exhaustive pour solutionner ce problème.
2. Proposez un algorithme diviser-pour-régner pour ce problème.
3. Comparez la complexité des deux algorithmes.

Bonus – La peinture **(adapté de l'IEEExtreme 2016)**

Vous souhaitez mettre en couleur une mosaïque. Comme vous êtes méthodiques, vous avez déjà acheté tous les pigments nécessaire et vous avez décidé de peindre la toile de haut en bas et droite à gauche. Vous connaissez donc exactement la suite de couleurs à utiliser. Vous disposez de deux pinceaux et souhaitez minimiser le nombre de fois que vous devez nettoyer votre pinceau. Quelle stratégie allez-vous adapter ?