Sorbonne Université

2I006 : Algorithmique appliquée et structures de données

Projet :

Sorting Robot

Kamil Rzeszutko
Angelo Ortiz

Licence d'Informatique
Année 2017/2018

Table des matières

1 Introduction

Ici, l'intro

2 Exercice 1

1. Soient les deux cases (i,j) et (k,l) dans une grille à m lignes et n colonnes. Soit la fonction $\texttt{dist}((i,j),(k,l)) = |k-i| + |l-j|$. Soit la propriété suivante $P(r), r \geq 0$:
   — Le chemin VH qui consiste à se déplacer de $|k-i|$ cases verticalement vers (k,j), puis de $|l-j|$ cases horizontalement vers (k,l), et
   — le chemin HV qui consiste à se déplacer de $|l-j|$ cases horizontalement vers (i,l), puis de $|k-i|$ cases verticalement vers (k,l),
   sont des plus courts chemins, où $r = \texttt{dist}((i,j),(k,l)) \geq 0$.

   Montrons cette propriété par récurrence faible sur $r \geq 0$.

   Base : Pour $r = 0$, $(k,l) = (i,j)$. On se déplace de 0 case verticalement et horizontalement. Ainsi, on reste dans la même case. Donc, la propriété est vérifiée pour $r = 0$.

   Induction : Supposons que la propriété soit vérifiée pour un $0 \leq r \leq m+n-3$ fixé. Montrons que la propriété est aussi vérifiée pour $r+1$.

   Soient $p = |k-i|, q = |j-l| \in \mathbb{N}$, tels que $r+1 = p+q$. Puisque $r+1 \geq 1$, au moins l'une des variables est non nulle. Supposons sans perte de généralité que $p > 0$, i.e. $p-1 \geq 0$.

   Trois cas sont possibles :

   (a) $i = 0$:
   Tout d'abord, $p = |k-0| = k$. Puis, pour aller de la case $(0,j)$ vers la case (k,l) avec un premier déplacement vertical d'une case, il faut passer par la case $(1,j)$. On a que $|k-1| = p-1$ et, en conséquence, $\texttt{dist}((1,j),(k,l)) = (p-1)+q = r$. Par hypothèse de récurrence, $P(r)$ est vérifiée. De ces deux faits, le chemin se déplaçant de $|k|$ cases verticalement vers (k,j), puis de $|l-j|$ cases horizontalement vers (k,l) est un plus court chemin.

   (b) $i = m-1$:
   Tout d'abord, $p = |k-(m-1)| = m-k-1$. Puis, pour aller de la case $(m-1,j)$ vers la case (k,l) avec un premier déplacement vertical d'une case, il faut passer par la case $(m-2,j)$. On a que $|k-(m-2)| = (m-2)-k = p-1$ et, en conséquence, $\texttt{dist}((m-2,j),(k,l)) = (p-1)+q = r$. Par hypothèse de récurrence, $P(r)$ est vérifiée. De ces deux faits, le chemin se déplaçant de $|k-(m-1)|$ cases verticalement vers (k,j), puis de $|l-j|$ cases horizontalement vers (k,l) est un plus court chemin.

   (c) $0 < i < m-1$:
   Pour aller de la case (i,j) vers la case (k,l) avec un premier déplacement vertical d'une case, il faut passer par la case (h,j), où $h \in \{i-1, i+1\}$, telle que $0 \leq h < m$ et $|k-h| = p-1$. On a alors que $\texttt{dist}((h,j),(k,l)) = (p-1)+q = r$. Par hypothèse de récurrence, $P(r)$ est vérifiée. Donc, en particulier, le chemin se déplaçant de $|k-h|$ cases verticalement vers (k,j), puis de $|l-j|$ cases horizontalement vers (k,l) est un plus court chemin.

   On fait de même pour $q > 0$ avec un traitement non plus sur les lignes, mais sur les colonnes de la grille, et on obtient que le chemin se déplaçant de $|l-j|$ cases horizontalement vers (i,l), puis de $|k-i|$ cases verticalement vers (k,l) est un plus court chemin

   Conclusion :

$$\left.\begin{array}{l} P(0) \text{ vraie} \\ \forall r \in \{0,\ldots,m+n-3\}, [P(r) \implies P(r+1)] \end{array}\right\} \begin{array}{l} \forall r \in \{0,\ldots,m+n-2\}, \\ \text{les chemins } VH \text{ et } HV \\ \text{sont des plus courts chemins.} \end{array}$$

3. — La fonction `estCaseNoire` nous permet de savoir si une case est noire, i.e. si elle porte une pièce de même couleur.

— La fonction `estPieceNoire` nous permet de savoir si une pièce est non noire, i.e. si sa couleur est différente de -1.
— La fonction `robotPortePiece` nous permet de savoir si le robot porte une pièce, i.e. si la couleur de la "pièce" du robot est différente de -1.
— La fonction `couleurPieceRobot` nous permet de savoir dans le cas où le robot porte une pièce, sa couleur.

3 Exercice 2

3. Pour l'analyse de la complexité des quatre implémentations, on commence par les morceaux de code communs aux toutes. Dans le pire cas, il n'y a aucune case noire dans la grille, i.e. il y a n^2 pièces à traiter, ce qui apparaît sous la forme d'une boucle `while` dans le code fourni, en l'occurrence la boucle principale des fonctions. De plus, la recherche d'une pièce non noire aux alentours du robot est faite au tout début de l'algorithme et à chaque fois que l'on ferme un circuit : cela dépend fortement du nombre de cycles dans la grille. Par conséquent, ceci n'est pas pris en compte pour l'analyse de la complexité et celle-ci se réduit à la recherche de pièces. Par ailleurs, on a un paramètre $\alpha = \left\lceil \frac{nm}{c} \right\rceil$, où m, n et c correspondent respectivement au nombre de lignes, de colonnes et de couleurs de la grille. Ici, $\alpha = \left\lceil \frac{n^2}{c} \right\rceil$.

   Les deux premières implémentations de l'algorithme au plus proche ont une complexité en $O(n^4)$. Pour chaque pièce on cherche la case la plus proche dont la couleur coïncide avec la sienne. L'implémentation naïve de l'algorithme fait une recherche sur toute la grille, ce qui est de l'ordre de $\Theta(n^2)$. Quant à la deuxième, la recherche est circulaire et on a donc une complexité en $O(n^2)$.

   Il est impératif de remarquer que la recherche circulaire est plus rapide au fur et à mesure que le paramètre α augmente.

   On a comparé les temps d'exécution des versions naïve et circulaire pour des grilles carrées de taille variable : la première version met 31.48 s pour une grille de longueur 110, tandis que la deuxième met 32.25 s pour une grille de longueur 155.

   Les expérimentations ont été faites avec un nombre de couleurs égal à $\left\lfloor \frac{n}{2} \right\rfloor$, ce qui donne $\alpha \approx 2n$. Comme dit précédemment, ce paramètre est important pour la durée effective de la recherche circulaire. C'est pourquoi on a obtenu des meilleurs résultats pour la deuxième version alors que la complexité pire-cas est la même pour toutes les deux.

4 Exercice 3

5. L'analyse de la complexité de l'implémentation par couleur de l'algorithme au plus proche se fera en deux parties. Dans un premier temps, on détermine une borne supérieure de la complexité. En effet, le robot porte une pièce à presque tout moment du jeu et cherche alors la case de même couleur la plus proche de lui. On a un accès en $\Theta(1)$ à la liste chaînée correspondant à la couleur de la pièce et on est obligé de parcourir toute la liste car il n'y a pas d'ordre défini lors de l'insertion. Soit l la longueur de la liste chaînée. On a donc que la complexité de la recherche de ladite case est en $\Theta(l)$. Or on a $l \leq n^2$. On en déduit que le traitement d'une pièce est de l'ordre de $O(n^2)$, ce qui donne une complexité en $O(n^4)$ pour cette version. Nonobstant, on a négligé un facteur clé de cette version : le paramètre α représentant le nombre maximal de pièces d'une même couleur et ainsi celui des cases de même couleur.

Dans l'étape suivante, on tient compte de ce paramètre dans l'analyse et on remarque qu'il remplace n^2 dans le paragraphe ci-dessus. De ce fait, on obtient que la complexité de la recherche dans la liste chaînée est en $O(\alpha)$, ce qui donne finalement une complexité en $O(\alpha n^2)$ pour la version par couleur de l'algorithme.

N.B. L'insertion de chacune des n^2 cases dans la liste chaînée correspondante est en $\Theta(1)$, ce qui donne une complexité de l'insertion en $\Theta(n^2)$. On remarque ainsi que les complexités données ci-dessus l'emportent à l'égard de celle de l'initialisation des listes.

5 Exercice 4

7. La dernière version de cette algorithme répose sur l'utilisation des arbres AVL. On a rangé les cases dans une matrice d'arbres : chaque ligne correspond à une couleur distincte et chaque colonne correspond à une ligne de la grille du jeu. Etant donné que l'on travaille avec une matrice, l'accès à un certain arbre est en $\Theta(1)$. Cependant, la recherche de la case la plus proche dans une ligne donnée est en $O(h(T))$, où $h(T)$ est la hauteur de l'arbre T correspondant. On sait d'ailleurs que $h(T) \leq \lceil \log_2 n(T) \rceil$, où $n(T)$ est le nombre de nœuds dans l'arbre T. Or on a $n(T) = \min(n, \alpha)$. Ceci nous donne une complexité en $O(\log \min(n, \alpha))$. On en déduit que la recherche sur les n lignes de la grille est en $O(n \log \min(n, \alpha))$. On obtient finalement une complexité en $O(n^3 \log \min(n, \alpha))$ pour cette dernière implémentation de l'algorithme au plus proche.

   N.B. L'insertion de chacune des n^2 cases dans l'arbre AVL correspondant est en $O(\log \min(n, \alpha))$, ce qui donne une complexité de l'insertion en $O(n^2 \log \min(n, \alpha))$. On remarque ainsi que la complexité donnée ci-dessus l'emporte à l'égard de celle de l'initialisation des arbres.

   Comme mentionné ci-dessus, les expérimentations ont été faites avec $\alpha \approx 2n$, ce qui transforme les complexités données précédemment en $O(n^3)$ pour la version par couleur et $O(n^3 \log n)$ pour la version par AVL. C'est pourquoi on a obtenu des meilleurs résultats pour la première de ces deux versions.

Les résultats empiriques ne collent pas à l'analyse de la complexité