UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

DEVOIR 3

PAR

JÉRÉMY BOUCHARD (BOUJ08019605)

JEAN-PHILIPPE SAVARD (SAVJ04079609)

ALEXIS VALOTAIRE (VALA09129509)

DEVOIR PRÉSENTÉ À

M. FRANÇOIS LEMIEUX

DANS LE CADRE DU COURS D'ALGORITHMIQUE (8INF433)

Note : Le code source LaTeX est disponible à l'URL suivant :
`https://github.com/AlexisCode101/algo-devoir-3`

Question 1

Nous avons vu en classe un algorithme vorace pour résoudre le problème du retour de la monnaie en un nombre minimum de pièces. Nous avons observé que cet algorithme fonctionne correctement avec de pièces de monnaie canadiennes mais celui-ci ne trouve pas toujours la bonne réponse si, par exemple, des pièces de 12 cents sont ajoutées (e.g. il y a une erreur pour retourner 15 cents). Le problème général du retour de la monnaie peut être résolu exactement en utilisant la programmation dynamique.

Soit n le nombre de pièces distinctes et soit $T[1 \dots n]$ un tableau donnant la valeur de ces pièces (il n'y a aucun intérêt à trier ce tableau). Supposons une quantité illimitée de chaque type de pièces. Soit L, une limite sur le montant à obtenir.

a) Pour $1 \leq i \leq n$ et $1 \leq j \leq L$, soit $c_{i,j}$ le nombre minimum de pièces pour obtenir le montant j si on se limite aux pièces de type $T[1], T[2], \dots, T[i]$. Si ce montant ne peut être obtenu alors $c_{i,j} = +\infty$. Donnez une équation de récurrence pour $c_{i,j}$ incluant les conditions initiales.

$$c_{i,j} \begin{cases} 1 + c_{i,j}(L - T[i]) & \text{si } L \geq 0 \\ +\infty & \text{ailleurs} \end{cases}$$

b) Donnez un algorithme de programmation dynamique pour calculer tous les $c_{n,j}$ où $1 \leq j \leq L$. Votre algorithme ne doit utiliser qu'un seul tableau (à une dimension) de longueur L.

Algorithme 1 : Trouver $c_{n,j}$

```
Données : T[1...n], entier n, entier L
Résultat : Entier c[L]
si L = 0 alors
    retourner (INT_MAX)
fin
pour j ← 1 à L faire
    c[j] ← INT_MAX
fin
pour j ← 1 à L faire
    pour i ← 1 à n faire
        si T[i] ≤ j alors
            si (j − T[i]) = 0 alors
                temp ← 0
            fin
            sinon
                temp ← c[j − T[i]]
            fin
            si temp ≠ INT_MAX & (temp + 1) < c[j] alors
                c[j] ← temp + 1
            fin
        fin
    fin
fin
```

c) Analysez le temps d'exécution de votre algorithme en fonction de n et de L.

Table 1: Analyse du temps d'exécution de l'algorithme

Ligne	Temps/Exécution	Nombre d'exécutions	Ordre
1	c_1	1	$O(1)$
2	c_2	1	$O(1)$
3	c_3	1	$O(1)$
4	c_4	$[L]+1$	$O(L)$
5	c_5	$[L]$	$O(L)$
6	c_6	1	$O(1)$
7	c_7	$[L]+1$	$O(L)$
8	c_8	$[L]\big([n]+1\big)$	$O(Ln)$
9	c_9	$[L][n]$	$O(Ln)$
10	c_10	$[L][n]$	$O(Ln)$
11	c_11	$[L][n]$	$O(Ln)$
12	c_{12}	$[L][n]$	$O(Ln)$
13	c_{13}	$[L][n]$	$O(Ln)$
14	c_{14}	$[L][n]$	$O(Ln)$
15	c_{15}	$[L][n]$	$O(Ln)$
16	c_{16}	$[L][n]$	$O(Ln)$
17	c_{17}	$[L][n]$	$O(Ln)$
18	c_{18}	$[L][n]$	$O(Ln)$
19	c_{19}	$[L][n]$	$O(Ln)$
20	c_{20}	$[L][n]$	$O(Ln)$
21	c_{21}	$[L]$	$O(L)$

À l'aide du tableau ci-dessus, nous pouvons conclure que le temps d'exécution de notre algorithme est de l'ordre de $O(Ln)$.

Question 2

Donnez toutes les étapes de l'algorithme de Floyd pour calculer le plus court chemin entre toutes les paires de noeuds du graphe de la figure 1. Donnez la matrice obtenue à chaque itération.

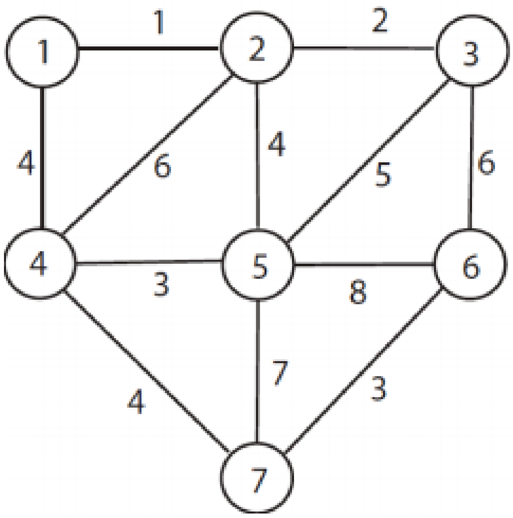

Figure 1: Graphe

Question 3

L'algorithme de Floyd que nous avons vu en classe donne la longueur du plus court chemin entre toutes les paires de noeuds mais ne permet pas de retrouver les plus courts chemins. Montrer comment modifier l'algorithme afin qu'il mémorise l'information permettant de retrouver efficacement (temps $O(n)$) le plus court chemin entre deux noeuds quelconque du graphe.

Question 4

Supposez que vous ayez un damier $n \times n$ et un jeton. Vous devez déplacer le jeton depuis le bord inferieur du damier vers le bord supérieur, en respectant la règle suivante. À chaque étape, vous pouvez placer le jeton sur l'un des trois carrés suivants

- le carré qui est juste au dessus,
- le carré qui est une position plus haut et une position plus à gauche (à condition que le jeton ne soit pas déjà dans la colonne la plus à gauche),
- le carré qui est une position plus haut et une position plus à droite (à condition que le jeton ne soit pas déjà dans la colonne la plus à droite).

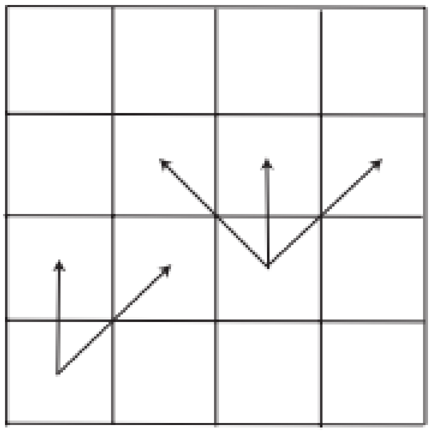

Figure 2: Graphe

La figure 2 illustre les déplacements valides à partir de deux positions dans un damier de dimension 4×4.

Chaque fois que vous vous déplacez du carré $x = (i, j)$ au carré $y \in \{(i-1, j-1), (i-1, j), (i-1, j+1)\}$, vous recevez $p(x, y)$ dollars. On vous donne $p(x, y)$ pour toutes les paires (x, y) correspondant à un déplacement licite de x à y. $p(x, y)$ **n'est pas forcément positif**.

Donner un algorithme efficace de programmation dynamique pour déterminer le montant maximum que vous pouvez empocher. Votre algorithme peut partir de n'importe quel carré du bord inférieur et arriver sur n'importe quel carré du bord supérieur pour maximiser le montant collecté au cours du trajet.

Expliquez comment vous pouvez optimiser l'espace mémoire et évaluez le temps d'exécution et l'espace mémoire de votre algorithme.