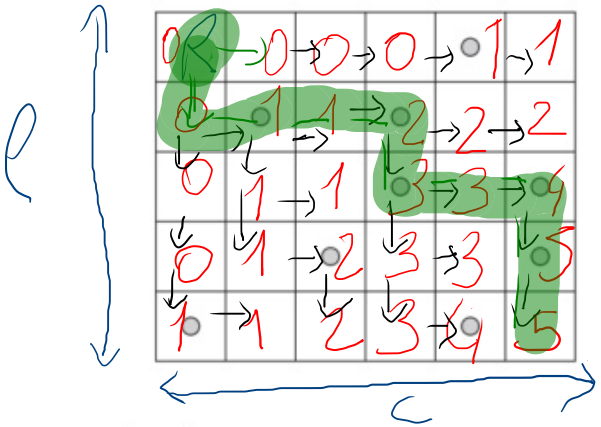
Exercice 7 – Robot collecteur de pièces

Des pièces sont placées sur les cases d'un échiquier $\ell \times c$ (ℓ lignes et c colonnes). Il y a au plus une pièce par case. Un robot, positionné sur la case en haut à gauche de l'échiquier, va collecter le plus de pièces possibles et les placer sur la case en bas à droite. A chaque pas, le robot peut se déplacer d'une case vers la droite ou vers le bas. Quand le robot passe sur une case avec une pièce, il prend la pièce. Le but de cet exercice est de concevoir un algorithme pour collecter un nombre maximum de pièces.



Q 7.1 Dans un premier temps, on s'intéresse uniquement au nombre maximum de pièces qu'on peut collecter sans chercher à identifier la trajectoire correspondante.

a. Quelle est la sous-structure optimale dont on a besoin pour résoudre ce problème ?

b. Caractériser (par une équation) cette sous-structure optimale.

c. En déduire le nombre maximum de pièces qu'on peut collecter.

a) $F(i,j)$: nb max de pièces qu'on peut collecter jusqu'à la case (i,j)

b) $F(i,j) = \max\{F(i,j-1), F(i-1,j)\} + c_{ij}$ avec $c_{ij} = \begin{cases} 1 \text{ si pièce en } (i,j) \\ 0 \text{ sinon} \end{cases}$

$F(1,j) = \sum_{k=1}^{j} c_{1k}$ et $F(i,1) = \sum_{k=1}^{i} c_{k1}$

c) $F(\ell, c)$

d. Ecrire un algorithme de programmation dynamique pour calculer ce nombre.

e. Quelle est la complexité de cet algorithme ?

d) $F(1,1) \leftarrow c_{11}$

pour i allant de 2 à ℓ faire

  $F(i,1) \leftarrow F(i-1,1) + c_{i1}$

pour j allant de 2 à c faire

  $F(1,j) \leftarrow F(1,j-1) + c_{1j}$

pour i allant de 2 à ℓ faire

  pour j allant de 2 à c faire

    si $F(i-1,j) \geq F(i,j-1)$ alors ($F(i,j) \leftarrow F(i-1,j) + c_{ij}$; $T(i,j) \leftarrow (i-1,j)$)

    sinon ($F(i,j) \leftarrow F(i,j-1) + c_{ij}$; $T(i,j) \leftarrow (i,j-1)$)

retourner $(F(\ell,c), T)$

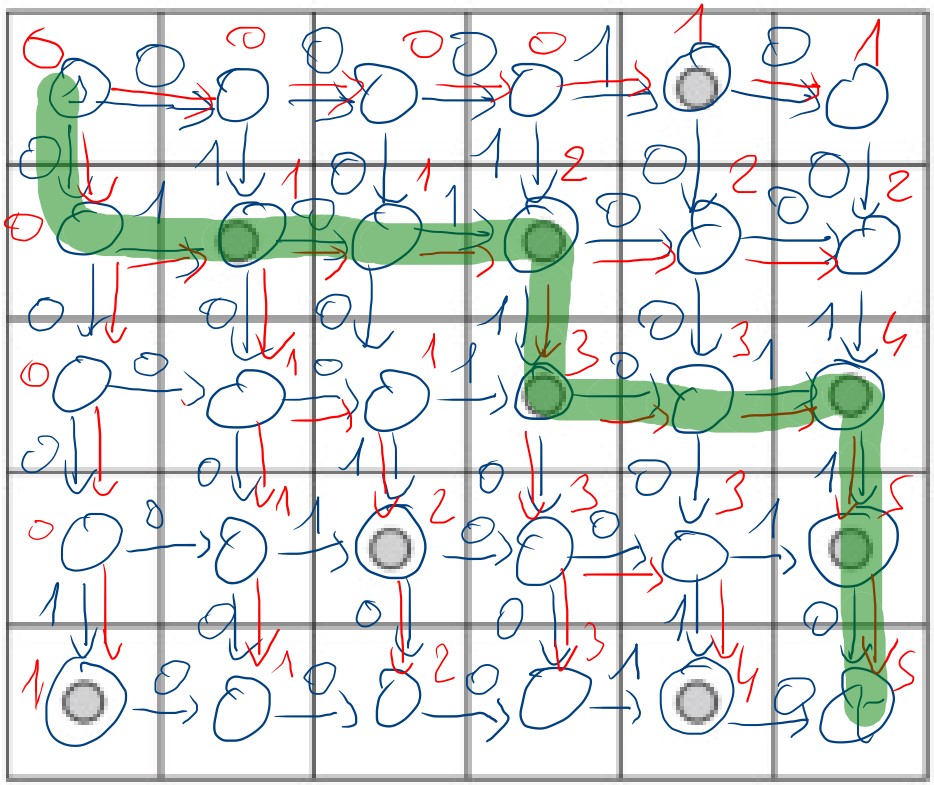
Complexité : $\Theta(\ell c)$

Q 7.2 Modifier l'algorithme précédent afin qu'il renvoie également la trajectoire optimale.

Voir les ajouts en rouge au dessus

Dans les questions qui suivent, on cherche à reformuler et résoudre le problème comme la recherche d'un plus long chemin dans un graphe.

Q 7.3 Pour l'échiquier ci-dessus, tracer un graphe valué avec un sommet source s et un sommet puits t, tel que le problème de collecte de pièces revient à déterminer un plus long chemin de s à t. Plus généralement, indiquer le nombre n de sommets et le nombre m d'arcs en fonction de ℓ et de c.



La valuation d'un arc est 1 s'il y a une pièce au sommet d'arrivée, 0 sinon.

- ℓc sommets
- $2\ell c - c - \ell$ arcs

Il y a un sommet par case de l'échiquier, et deux successeurs par sommets (correspondant aux deux cases où il est possible de se déplacer).

Q 7.4 En s'inspirant d'un algorithme vu en cours dont on rappelera le nom, donner le *principe* (sans entrer dans les détails) d'un algorithme de calcul d'un plus long chemin dans un graphe sans circuit. Appliquer cet algorithme sur le graphe de la question 1. Quel est le plus long chemin ?

Q 7.5 Quelle est la complexité de l'algorithme précédent en fonction de n et de m ? En déduire la complexité (en fonction de ℓ et de c) de l'algorithme qui, partant d'un échiquier sur lequel des pièces sont positionnées, détermine un itinéraire optimal pour collecter un nombre maximum de pièces.

4) Comme le graphe est sans circuit par construction, on peut adapter l'algorithme de Bellman pour calculer un plus long chemin en substituant une opération de max à l'opération de min habituellement utilisée quand on recherche un plus court chemin.
Le plus long chemin est souligné en vert sur la figure précédente.

5) Complexité de Bellman : $\Theta(n+m)$ si le graphe est représenté par des listes de prédécesseurs.
$\rightarrow n + m = 3\ell c - c - \ell \rightarrow \Theta(\ell c)$