Colles programmation dynamique

Exercice 2

On considère un tableau $T = [t_0, ..., t_{n-1}]$ d'entiers. On dit que t_i est un maximum local de T si t_i est supérieur ou égal à son voisin de gauche (s'il existe) et à son voisin de droite (s'il existe). En particulier, t_0 est un maximum local si et seulement si il est supérieur à t_1.

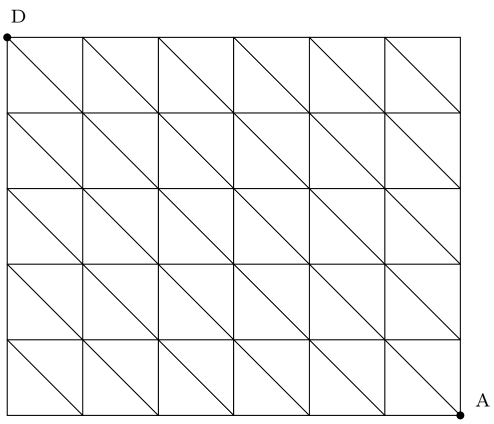
1. Décrire un algorithme naïf permettant de calculer l'un des maxima locaux de T et sa complexité.
2. Montrer que tout tableau non vide admet un maximum local.
3. En déduire un algorithme efficace pour trouver un maximum local de T et sa complexité.

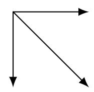
On considère à présent une matrice de taille $n \times n$ et on y étend la définition de maximum local (une valeur est un maximum local si elle est supérieure aux valeurs voisines - qui sont au plus au nombre de 4).

4. Concevoir un algorithme aussi efficace que possible pour calculer l'un des maxima locaux d'une matrice de taille n (en justifiant sa correction et sa complexité).

Exercice 4

On considère une grille de taille $n \times p$ telle que celle dessinée ci-dessous (pour $n = 5$ et $p = 6$). On note $f(n, p)$ le nombre de chemins possibles permettant d'aller du point supérieur gauche au point inférieur droit en se déplaçant dans la grille uniquement selon les directions proposées ci-dessous :





1. Indiquer les valeurs de $f(0, p)$ et $f(n, 0)$.
2. Déterminer une relation de récurrence sur $f(n, p)$ lorsque que $p, n \geq 1$.
3. En déduire un algorithme dynamique utilisant une approche bottom-up permettant de calucler $f(n, p)$ étant donnés n et p. Déterminer sa complexité en temps et en espace.
4. Expliquer comment améliorer la complexité spatiale de cet algorithme.

Exercice 5

On considère des entiers naturels disposés sur un triangle équilatéral de hauteur n. La figure suivante donne un exemple pour un tel triangle lorsque $n = 4$:

3

7 **4**

2 **4** 6

8 5 **9** 3

On appelle chemin une suite de nombres du triangle obtenue en se déplaçant vers les nombres adjacents de la ligne inférieure à partir du sommet et finissant sur un nombre de la base. Un chemin est indiqué en gras sur la figure précédente. On cherche à calculer la valeur maximale d'un chemin reliant le sommet à la base d'un triangle de hauteur n donné. Un triangle sera représenté par une matrice $n \times n$ contenant à la ligne i les coefficients de la i-ème ligne du triangle.

1. Résoudre le problème dans le cas du triangle ci-dessus.

2. Combien y a-t-il de chemins possibles dans le triangle ? En déduire la complexité d'un algorithme naïf (brièvement décrit) résolvant ce problème : est-il envisageable de procéder ainsi ?

3. Proposer un algorithme dynamique permettant de résoudre ce problème. On pourra établir une relation de récurrence sur $f(i,j)$, la valeur maximale d'un chemin passant du sommet à la position (i,j).

4. Etudier la complexité temporelle et spatiale de cet algorithme.

5. Peut on modifier l'algorithme de sorte à calculer un chemin de valeur maximale en plus de ladite valeur sans déteriorer sa complexité ?

Exercice 6

On considère une matrice M à m lignes et n colonnes dont les coefficients sont dans $\{0,1\}$. On cherche à déterminer le nombre maximal de lignes d'un carré inclus dans A dont tous les coefficients valent 1.

1. Résoudre ce problème dans le cas de la matrice suivante :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Notons $f(i,j)$ la taille maximale d'un carrée de 1 dont le coin inférieur droit est à la position (i,j).

2. Etablir une relation de récurrence sur les $f(i,j)$.

3. En déduire un algorithme dynamique permettant de résoudre ce problème.

4. Etudier sa complexité temporelle et spatiale. Peut on améliorer cette dernière ?

5. Expliquer comment calculer la localisation du carré de taille maximale en adaptant l'algorithme précédent.

Exercice 7

On considère un graphe orienté à n sommets $s_1, ..., s_n$. On suppose que tous les sommets sauf s_n ont au moins un arc sortant et que si (s_i, s_j) est une arête alors $i < j$. On suppose en outre que pour tout $i \in [\![1, n]\!]$, il existe un chemin de s_i vers s_n. On cherche à calculer le plus long chemin entre s_1 et s_n et sa longueur.

1. Montrer que l'algorithme glouton ci-dessous ne résout pas le problème :

```
u ← s_1, L ← 0
Tant qu'il existe un arc sortant de u
   Déterminer l'arc (u, s_i) tel que i est minimal
   u ← s_i
   L ← L + 1
Renvoyer L
```

2. Pour tout $i \in [\![1, n]\!]$, on note $l(s_i)$ la longueur d'un plus long chemin de s_1 à s_i. Déterminer une relation de récurrence sur cette quantité.

3. En déduire un algorithme dynamique permettant de calculer la longueur d'un plus long chemin de s_1 à s_n. Etudier sa complexité en temps et en espace.

4. Modifier l'algorithme de sorte à calculer un plus long chemin en plus de sa longueur.

5. Peut-on modifier cet algorithme de sorte à calculer le chemin le plus lourd dans le cas où le graphe est pondéré ?

Exercice 8

Un étudiant a récupéré les grilles de salaires de n emplois : dans la case (i,j) de cette grille se trouve le salaire $s(i,j)$ que l'on gagne si on travaille i heures par semaine dans l'emploi j. L'étudiant ne souhaite pas travailler plus de H heures par semaine et veut maximiser son salaire hebdomadaire. On note $f(h,i)$ le salaire maximal perçu en travaillant h heures réparties sur les emplois de 1 à i.

1. Déterminer $f(h, 1)$ et $f(0, i)$ puis décrire une relation de récurrence vérifiée par $f(h, i)$.

2. En déduire un algorithme dynamique utilisant une approche top-down permettant de calculer le salaire maximal que l'on peut obtenir en travaillant H heures réparties sur les n emplois.

3. Indiquer quelles sont les valeurs de $f(h, i)$ qui seront calculées par cet algorithme, quel est le salaire maximal et quelle répartition permet d'obtenir ce salaire maximal dans le cas où $H = n = 4$ et où la grille des salaires est :

	Emploi 1	Emploi 2	Emploi 3	Emploi 4
0 heures	0	0	0	0
1 heure	26	23	16	19
2 heures	39	36	32	36
3 heures	48	44	48	47
4 heures	54	49	64	56

4. Peut on utiliser une approche bottom-up pour résoudre ce problème ? Si oui, indiquer quels seront les $f(h, i)$ que cet algorithme calculera sur l'exemple précédent. Etudier sa complexité spatiale et temporelle.