Cette *vingt-huitième* colle vous fera travailler sur un problème combinatoire, à résoudre par programmation dynamique.

Cet énoncé est inspiré d'une colle donnée cette année en MP2I au Lycée Fénelon Sainte-Marie.

Problème étudié

Étant donnée une matrice d'entiers $A = (a_{i,j})_{i,j}$, de taille $n \times k$ ($n, k \in \mathbb{N}^*$), on souhaite connaître un chemin, qui n'utilisera que des déplacements $\rightarrow$ (d'une case vers la droite) ou $\downarrow$ (d'une case vers le bas), partant de la case en haut à gauche (d'indice $(i, j) = (0, 0)$) et allant à la case en bas à droite (d'indice $((i, j) = (n - 1, k - 1))$), qui maximise la somme des entiers rencontrées. On appelle cette somme le **poids** du chemin.

Voici un exemple de matrice A, de taille $n = 7$ fois $k = 8$, avec un chemin de poids maximum indiqué **en gras** :

$$A = \begin{pmatrix} \mathbf{2} & \mathbf{39} & \mathbf{12} & \mathbf{49} & \mathbf{47} & 18 & 22 & 19 \\ 37 & 21 & 34 & 26 & \mathbf{10} & 2 & 35 & 39 \\ 31 & 21 & 12 & 26 & \mathbf{34} & \mathbf{27} & 7 & 22 \\ 20 & 46 & 16 & 2 & 11 & \mathbf{40} & \mathbf{36} & \mathbf{13} \\ 18 & 30 & 32 & 37 & 28 & 24 & 9 & \mathbf{6} \end{pmatrix}$$

Le poids maximum est ici égal à $2 + 39 + 12 + 49 + 47 + 10 + 34 + 27 + 40 + 36 + 13 + 6 = 315$.

Exploration exhaustive ?

1. Si on suppose la matrice carrée (c'est-à-dire $k = n$), quelle serait la complexité temporelle d'un algorithme de recherche exhaustive, qui énumère tous les chemins possibles allant de $(0, 0)$ à $(n - 1, n - 1)$?
2. Est-ce que c'est assez efficace pour être implémenté en pratique, sur des grandes matrices ?

Propriété de sous-problèmes optimaux

On ne suppose plus la matrice A carrée pour la suite. On considère une matrice A fixée.

Supposons qu'un chemin C de poids maximum allant de $(0, 0)$ à $(n - 1, k - 1)$ passe par la case (i, j).

3. Montrer que le sous chemin extrait de C de la case $(0, 0)$ à (i, j) est de poids maximum. (c'est une propriété de **sous-optimalité**)

Pour deux indices $0 \leqslant i < n$ et $0 \leqslant j < k$, on note $p_{i,j}$ le poids maximum d'un chemin de $(0,0)$ à (i,j).

4. Justifier (rapidement) pourquoi $p_{i,j}$ est noté **le poids maximum** alors qu'il peut ne pas y avoir unicité d'un chemin de poids maximum (par exemple pour une matrice remplies d'une même valeur).
5. Justifier (rapidement) pourquoi $p_{i,j}$ existe.
6. Donner des cas de bases de $p_{i,j}$ qui soient directement calculables, sans récursion.
7. Donner une formule de récurrence sur $p_{i,j}$, si $i > 0$ et $j > 0$, faisant intervenir des valeurs proches.
8. Comment trouver le poids maximum du chemin total (cf. problème initial), en fonction de ces valeurs de $p_{i,j}$?

Implémentation : en OCaml

On va choisir la simplicité et représenter la matrice A par un `int array array` en OCaml : on accédera donc à sa valeur $a_{i,j}$ par `a.(i).(j)` dans le code.

Résolution récursive naïve

9. Depuis cette formule de récurrence obtenue en 7., écrire une fonction récursive simple `poidsmax_rec : int array array -> int * int -> int` tel que l'appel à `poidsmax_rec matrice_a (i, j)` renvoie le poids maximum d'un chemin allant de $(0,0)$ jusqu'à (i,j) dans la matrice A donnée par le `int array array` appelé `matrice_a`.
10. Quelle est la complexité temporelle de cette fonction `poidsmax_rec` ? Comparer avec la complexité de l'approche par exploration exhaustive de la Q1.
11. Et quelle est sa complexité mémoire ? On fera attention à bien compter la taille de la pile d'appel, et pas seulement la mémoire explicitement allouée et utilisée dans la fonction.

Résolution par programmation dynamique

12. Au brouillon, décrire un algorithme, suivant le paradigme de la programmation dynamique, pour calculer `poidsmax_rec matrice_a (n-1, k-1)` sans appels récursifs.
13. En déduire une fonction `poidsmax_dyn : int array array -> int` qui implémente cet algorithme.
14. Quelle est sa complexité temporelle en fonction des dimensions n et k de la matrice A ?
15. Comparer avec le résultat de la Q10. sur la méthode récursive naïve.
16. Quelle est la complexité spatiale de la fonction `poidsmax_dyn` ? Peut-on faire mieux, quitte à être un peu malin ?

Exemples

17. Écrire une matrice `matrice_a : int array array` qui représente celle de l'exemple donné en première page.
18. En appelant `poidsmax_rec` et `poidsmax_dyn`, vérifier que sur cet exemple le poids du chemin maximum est bien égal à 315.
19. Tester sur au moins un autre exemple non trivial ($n, k \geqslant 3$).

Dernière question

20. Les fonctions précédentes ne font que donner le poids maximum d'un chemin (la valeur $f(x^*)$ dans les notations du cours). Comment ferait-on pour **trouver** un chemin de poids maximum ? Expliquer votre idée, sans l'implémenter.

Extra (révisions concours, cet été, etc) : implémentation en C

21. Faire la même implémentation, mais en langage C.

