

## TP Informatique 8

### Exercice 1

La suite de Fibonacci  $(u_n)_n$  est définie par

$$u_0 = 0, \quad u_1 = 1, \quad \forall n \in \mathbb{N} \quad u_{n+2} = u_{n+1} + u_n$$

1. Écrire une fonction `fib1(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière descendante et sans mémorisation.
2. Écrire une fonction `fib2(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière descendante et avec mémorisation.
3. Écrire une fonction `fib3(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière ascendante.
4. Tester chaque fonction pour  $n \in \{35, 100, 1000\}$  puis commenter les comportements observés.

### Exercice 2

La suite de Catalan  $(u_n)_n$  est définie par

$$u_0 = 1, \quad \forall n \in \mathbb{N}^* \quad u_n = \sum_{k=0}^{n-1} u_k u_{n-1-k}$$

1. Écrire une fonction `cata1(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière descendante et sans mémorisation.
2. Écrire une fonction `cata2(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière descendante et avec mémorisation.
3. Écrire une fonction `cata3(n)` d'argument `n` entier qui renvoie  $u_n$  en réalisant le calcul de manière ascendante.
4. Tester chaque fonction pour  $n \in \{17, 100, 1000\}$  puis commenter les comportements observés.

### Exercice 3

On rappelle que le coefficient binomial  $\binom{n}{k}$  avec  $k$  et  $n$  entiers désigne le nombre de parties à  $k$  éléments dans un ensemble à  $n$  éléments (choix de  $k$  éléments parmi  $n$  sans ordre ni répétition). On a la relation de Pascal

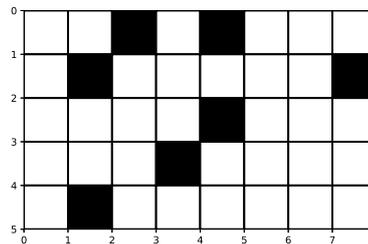
$$\forall (k, n) \in \mathbb{N}^{*2} \quad \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

1. Écrire une fonction `binom1(n, k)` d'arguments `n` et `k` entiers qui renvoie  $\binom{n}{k}$  en réalisant le calcul de manière descendante et sans mémorisation.
2. Écrire une fonction `binom2(n, k)` d'arguments `n` et `k` entiers qui renvoie  $\binom{n}{k}$  en réalisant le calcul de manière descendante avec mémorisation.
3. Écrire une fonction `binom3(n, k)` d'arguments `n` et `k` entiers qui renvoie  $\binom{n}{k}$  en réalisant le calcul de manière ascendante puis modifier la fonction pour qu'elle renvoie la totalité du triangle de Pascal.
4. Tester chaque fonction pour  $n = 10$  et  $k \in \llbracket 0; n \rrbracket$ .

## Exercice 4

Une pièce quadrillée est modélisée par une liste de listes : une case vide est représentée par 0 et un mur est représenté par 2. Par exemple, la liste

```
[[0, 0, 2, 0, 2, 0, 0, 0],  
 [0, 2, 0, 0, 0, 0, 0, 2],  
 [0, 0, 0, 0, 2, 0, 0, 0],  
 [0, 0, 0, 2, 0, 0, 0, 0],  
 [0, 2, 0, 0, 0, 0, 0, 0]]
```



représente la configuration de l'image ci-contre

On se propose de déterminer un plus court chemin depuis la case  $(0, 0)$  en haut à gauche vers la case  $(L-1, C-1)$  en bas à droite avec  $L$  le nombre de lignes de la pièce et  $C$  le nombre de colonnes.

⚠ Les seuls déplacements autorisés sont : vers la droite et vers le bas.

On considère que le coût d'accès à une case vide est égal à 1 et que le coût d'accès à une case murée est égal à  $+\infty$ . Pour  $(i, j) \in \llbracket 0; L-1 \rrbracket \times \llbracket 0; C-1 \rrbracket$ , on note  $c_{i,j}$  le coût d'accès à la case  $(i, j)$  et  $t_{i,j}$  le coût d'un plus court chemin depuis  $(0, 0)$  vers  $(i, j)$ .

Les fonctions sont à compléter dans le fichier `TP08_EX04.py`.

1. Pour  $j \in \llbracket 1; C-1 \rrbracket$ , déterminer  $t_{0,j}$  en fonction de  $t_{0,j-1}$  et  $c_{0,j}$  puis pour  $i \in \llbracket 1; L-1 \rrbracket$ , déterminer  $t_{i,0}$  en fonction de  $t_{i-1,0}$  et  $c_{i,0}$ .
2. Pour  $(i, j) \in \llbracket 1; L-1 \rrbracket \times \llbracket 1; C-1 \rrbracket$ , déterminer  $t_{i,j}$  en fonction de  $t_{i,j-1}$ ,  $t_{i-1,j}$  et  $c_{i,j}$ .
3. Compléter la fonction `ppc_tab(config)` d'argument `config` une liste de listes représentant une pièce, qui renvoie la matrice  $(t_{i,j})_{(i,j) \in \llbracket 0; L-1 \rrbracket \times \llbracket 0; C-1 \rrbracket}$  de manière ascendante.
4. Compléter la fonction `ppc(config)` d'argument `config` une liste de listes représentant une pièce, qui renvoie, s'il existe un plus court chemin depuis  $(0, 0)$  vers  $(L-1, C-1)$ , la liste de listes avec des 1 placés sur les cases du plus court chemin obtenu avec le poids de ce plus court chemin et `False` sinon.
5. Tester le programme pour différentes configurations.