Révisions

Piocher parmi les exercices suivants ceux qui vous inspirent. Les exercices sont classés par difficulté plus ou moins croissante.

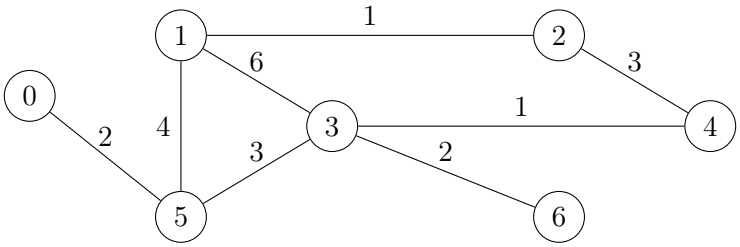
♦ Exercice 1 :

1°) Écrire une fonction somcube, d'argument n, renvoyant la somme des cubes des chiffres du nombre entier n.

2°) Trouver tous les nombres entiers inférieurs à 1000 égaux à la somme des cubes de leurs chiffres.

3°) En modifiant les instructions de la fonction somcube, écrire une fonction somcube2 qui convertit l'entier n en une chaîne de caractères permettant ainsi la récupération de ses chiffres sous forme de caractères. Cette nouvelle fonction renvoie toujours la somme des cubes des chiffres de l'entier n.

♦ Exercice 2 :



1°) Construire la matrice $(M_{i,j})_{0\leqslant i,j\leqslant 6}$, matrice de distances du graphe G, définie par : pour tous les indices i, j, $M_{i,j}$ représente la distance entre les sommets i et j, ou encore la longueur de l'arête reliant les sommets i et j. On convient que, lorsque les sommets ne sont pas reliés, cette distance vaut -1. La distance du sommet i à lui-même est, bien sûr, égale à 0.

2°) Écrire une fonction `voisins`, d'argument i, renvoyant la liste des voisins du sommet i.

3°) Écrire une fonction longueur, d'argument une liste L de sommets de G, renvoyant la longueur du trajet décrit par cette liste L, c'est-à-dire la somme des longueurs des arêtes empruntées. Si le trajet n'est pas possible, la fonction renverra -1.

♦ Exercice 3 :

1°) On considère le code Python de la fonction d suivante :

```
def d ( n ) :
        L =[1]
        for nombre in range (2 , n +1) :
                if n % nombre == 0 :
                        L . append ( nombre )
        return L
```

Que fait la fonction d ?

2°) Un diviseur non-trivial d'un entier n est un diviseur de n différent de 1 et de n. Écrire une fonction `dnt`, d'argument n, renvoyant la liste des diviseurs non-triviaux de l'entier n.

3°) Écrire une fonction `sommeCarresDNT`, d'argument n, renvoyant la somme des carrés des diviseurs non- triviaux de l'entier n.

4°) Écrire la suite des instructions permettant d'afficher tous les nombres entiers inférieurs à 1000 et égaux à la somme des carrés de leurs diviseurs non-triviaux. Que peut-on conjecturer ?

♦ Exercice 4 :

Soit une liste t de longueur n dont les termes valent 0 ou 1. Le but de cet exercice est de trouver le nombre maximal de 0 contigus dans t (c'est-à-dire figurant dans des cases consécutives). Par exemple, le nombre maximal de zéros contigus de la liste $t1$ suivante vaut 4 :

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$t[i]$	0	1	1	1	0	0	0	1	0	1	1	0	0	0	0

1°) Écrire une fonction `nombreZeros(t,i)`, prenant en paramètres une liste t, de longueur n et un indice i compris entre 0 et $n-1$, et renvoyant :

$$\begin{cases} 0 \text{ si } t[i]=1 \\ \text{le nombre de zéros consécutifs dans t à partir de t[i] inclus, si t[i]} = 0 \end{cases}$$

Par exemple, les appels nombreZeros(t1,4), nombreZeros(t1,1) et nombreZeros(t1,8) renvoient respectivement les valeurs 3, 0 et 1.

2°) Comment obtenir le nombre maximal de zéros contigus d'une liste t connaissant la liste des nombreZeros(t,i) pour $0 \leqslant i \leqslant n-1$?
En déduire une fonction nombreZerosMax(t), de paramètre t, renvoyant le nombre maximal de 0 contigus d'une liste t non vide. On utilisera la fonction nombreZeros.

3°) Quelle est la complexité de la fonction nombreZerosMax construite à la question précédente ?

4°) Trouver un algorithme plus performant.

♦ Exercice 5 :

Soient n un entier naturel strictement positif et p un réel compris entre 0 et 1. On considère X et Y deux variables aléatoires à valeurs dans $\mathbb{N}$ sur un espace probabilisé donné. X suit une loi de Poisson de paramètre $\lambda = np$ et Y suit une loi binomiale de paramètres (n,p).

1°) Définir une fonction `Px`, d'arguments k, n et p, renvoyant la valeur de $\mathbf{P}(X=k)$.
Déterminer, pour $n=30$ et $p=0.1$, la liste des valeurs de $\mathbf{P}(X=k)$ pour $k \in \mathbb{N}, 0 \leqslant k \leqslant 30$.

2°) Définir une fonction `Py`, d'arguments k, n et p, renvoyant la valeur de $\mathbf{P}(Y=k)$.
Déterminer, pour $n=30$ et $p=0.1$, la liste des valeurs de $\mathbf{P}(Y=k)$ pour $k \in \mathbb{N}, 0 \leqslant k \leqslant 30$.

3°) Soit $k \in \mathbb{N}$. On rappelle que, sous certaines conditions sur n et p, la probabilité $\mathbf{P}(Y=k)$ peut être approchée par $\mathbf{P}(X=k)$. Déterminer une fonction `Ecart` d'arguments n et p, renvoyant le plus grand des nombres $|\mathbf{P}(Y=k) - \mathbf{P}(X=k)|$, pour $0 \leqslant k \leqslant n$.

4°) Soit e un réel strictement positif. Déterminer une fonction N, d'arguments e et p, renvoyant le plus petit entier n tel que `Ecart(n, p)` soit inférieur ou égal à e.

5°) Faire l'application numérique dans les quatre cas suivants :
- p = 0.075 avec e = 0.008 et e = 0.005.
- p = 0.1 avec e = 0.008 et e = 0.005.

Interpréter le dernier résultat.

♦ Exercice 6 :

1°) Écrire un programme python qui tire au hasard une combinaison de Master Mind (cf. Wikipedia)

2°) Ecrire une fonction `note` qui prend en argument une combinaison secrète et une combinaison proposée et qui renvoie le nombre de "couleurs" bien placées et mal placées ;

3°) Ecrire une fonctionpartie qui choisit une combinaison au hasard et qui l'a fait deviner à l'utilisateur en comptant le nombre d'essais pour y arriver ;

4°) Réfléchir à différentes statégies pour faire jouer l'ordinateur contre lui-même.