Exercice

Informatique tronc commun 2ère année

Pour chaque question, donner la complexité de l'algorithme fourni.

1) Proposer une fonction `est_present(L,x)` qui renvoie un booléen en fonction de la présence de x dans la liste L.

2) Proposer une fonction `compte(L,x)` qui renvoie un entier donnant le nombre d'occurrences (apparitions) de x dans L.

3) Écrire une fonction `max(L)` qui renvoie le maximum de la liste L.

4) Écrire une fonction `indice_max(L)` qui renvoie tous les indices des occurrences du maximum dans L. Essayer en un seul parcours de L.

5) Écrire une fonction `binom(n,p)` qui calcule un coefficient binomial à partir de la formule des factorielles.

6) Écrire une deuxième fonction `binom2(n,p)` **récursive** à partir de la formule $\binom{n}{0} = 1, \binom{n}{n} = 1, \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

7) Rappeler l'instruction qui permet d'inverser une liste Python.

8) Écrire une fonction `PGCD(a,b)` en testant tout les nombres de 1 au minimum de a et b. Faire une deuxième fonction `PGCD(a,b)` basé sur l'algorithme d'Euclide.

9) Écrire une fonction `base2to10(L)` qui prend une liste de 0 et 1 et renvoie l'entier en base 10 correspondant (on supposera que le bit de poids fort est à la fin de la liste (I.E. : [0,1,0,1] correspond à 10).

10) Écrire une fonction `base10to2(x)` qui prend un entier et fait l'opération contraire à la question précédente.

11) Écrire une fonction `maxdeux(L)` qui renvoie le deuxième maximum de la liste L supposé de taille supérieur ou égal à 2 et sans doublons.

12) Écrire une fonction `dicho(f,a,b,eps)` qui prend en entrée une fonction f supposée continue, deux flottants a, b tel que $f(a)$ et $f(b)$ sont de signes différents et qui renvoie un flottant x tel qu'il existe $z \in [x - eps, x + eps]$ et $f(z) = 0$.

13) Écrire une fonction `dicho_liste(L,x)` qui renvoie un booléen si x est dans L. La complexité doit être en O(log(len(L))).

14) Écrire une fonction `suite_succes(L)` qui prend en entrée une liste de booléens et renvoit les longueurs des suites de `True`. Ex : `suite_succes([True,True,False,True,True,True,True,False,False,True])` renvoie `[2,4,1]`.

15) Une application de $[|1,n|]$ dans $[|1,p|]$ peut être représentée par une liste de taille n d'entiers entre $[|1,p|]$. Par exemple, la liste `[1,5,4]` représente la fonction f qui va de $[|1,3|]$ dans $[|1,p|]$ avec $p \geq 5$ et telle que $f(1) = 1, f(2) = 5, f(3) = 4$.
Écrire des fonctions `injective(L,p)`, `surjective(L,p)` et `bijective(L,p)`

16) Proposer une fonction `plus_proche(L)` qui prend en entrée une liste d'entiers de taille au moins 2 et renvoie un couple d'éléments de L les plus proches entre eux. Ex : `plus_proche([1,5,13,18,7]` renvoie (5,7).

17) Écrire une fonction `occurrences_lettre(s)` qui prend en entrée une chaîne de caractères s et renvoie un dictionnaire dont les clés sont les caractères de s, et les valeurs leur nombres d'apparitions dans s.

Faire une fonction `occurrences_mot(s)` qui renvoie un dictionnaire dont les clés sont les mots de s, et les valeurs sont le nombre d'occurences de ces mots. On supposera que les mots sont seulement séparés par des caractères ' ', et on ignorera tous ce qui concerne la ponctuation.

1 Lissage

On souhaite réaliser un algorithme pour générer aléatoirement du relief dans le but de créer une carte d'un monde quelconque. Un monde sera stocké sous forme d'une liste de listes de flottants.

Pour générer du relief, on tire aléatoirement des entiers représentant l'altitude d'une coordonnée. On dispose pour cela de la fonction random() supposée importée du module random. Cette fonction tire un flottant aléatoire entre 0 et 1.

On supposera que notre liste de listes carrée représente une matrice carrée.

18) Écrire une fonction `generer_terrain(N:int,min:float,max:float)` qui génère aléatoirement un terrain de taille N, donc l'altitude minimum est `min` et l'altitude maximum est `max`. La fonction renvoie donc une liste de liste de flottants.

Le monde généré ne sera peut-être pas très vraisemblable. On souhaite donc lisser ce monde. Pour cela, on va lisser le relief en utilisant cette méthode :

— On décide d'une largeur k

— Pour chaque coordonnée x,y; la nouvelle valeur de l'altitude x,y après lissage sera la moyenne des points x',y' telle que
$x' = x + kx, y' = y + ky$ et $|kx| + |ky| \leq k$.
On dira de tel point x', y' qu'ils font partie du cercle de centre x, y et de rayon k.

19) Écrire au préalable une fonction `lissage1(M)` qui effectue un lissage pour un paramètre k valant 1.

20) Écrire une fonction `voisins(N:int, x : int,y : int,k : int)` qui renvoie la liste de tous les couples (x',y') faisant partie du cercle de centre x, y et de rayon k d'un monde le largeur N.