Exercice_ecrit_1

October 3, 2019

0.1 Exercise 1: Comprendre les algorithms

En prenant en compte les deux algorithms suivants:

```
In [ ]: def algo1(L, n):
            res = True
            for i in range(2, n):
                if L(i-1) > L(i):
                    res = False
            return res
```

```
In [ ]: import math

        def algo2(L, n):

            if n == 1:
                return 0
            d = math.abs(L(2) - L(1))
            for i in range(3, n):
                next_d = math.abs(L(i) - L(i-1))
                if next_d > d:
                    d = next_d
```

1) Quel est la sortie de chacun de ces algorithmes si les donnees en entree valent L= (7,14,22,13,17) et n = 5 ?

2) De manière générale quel est la srtie de ces deux algorithms ?

0.2 Exercise 2: Corriger des algorithms

On demande à deux étudiants d'écrire un algorithm qui : calcule la somme des n premiers nombres entiers faisant partie de la liste des multiples de 5 ou de celle des multiples de 3 (ou non-exclusif). Voici les algorithms écrits par Bob et Sarah:

```
In [ ]: def bob_algo(n):
            s = 0
            j = 1
            for i in range(n):
                while j%5 != 0 and j%3 != 0:
```

```
                #Rappel j+= 1 est equivalent à j = j + 1
                j += 1
            s += j
        return s
```

```
In [ ]: def sarah_algo(n):
        s = 0
        i = 1
        j = 1

        while i <= n:
            j += 1

            if j%5 == 0:
                s += j
                i += 1
            if j%3 == 0:
                s += j
                i += 1
        return s
```

1) Chacun des deux algorithms ont un problèmes, trouvez lesquelles et donner un example dans lequel ils ne fonctionnent pas.

2) Corriger ces deux algorithms pour qu'ils fonctionnent.

0.3 Exercise 3: Ecrire un algorithme

Ecrire un algorithm qui prend en entrée une liste L de nombres entiers, de taille n et x un nombre entier positif; et qui retourne True s'il existe i et j appartenant à [1, n] tels que |L(i) - L(j)| > x et False dans le cas contaire.

0.4 Exercise 4: Ecrire une version récursive

Voici un algorithme qui calcule la suite de Syracuse (https://fr.wikipedia.org/wiki/Conjecture_de_Syracuse)

```
In [ ]: def syracuse(x):
        L = [x]
        while x != 1:
            if x%2 == 0:
                x /= 2
            else:
                x = 3*x + 1
            L.append(x)
        return L
```

Ecriver une version récursive de cette algorithm.

0.5 Exercise 5: Que fait cette algorithme ?

Condidère l'algorithme suivant qui prend en entrée une liste L de taille n et un nombre entier x:

In [ ]:
```
import math

def bizarre(L, n, x):
    a = 1
    b = n
    while b >= a
        j = math.ceil((a + b)/2)
        if  x == L(j):
            return True
        else:
            if L(j) < x:
                a = j + 1
            else:
                b = j - 1
    return False
```

1) Si L= (2,2,2,2,2,2,2,2), n = 8 et x = 1, quelle est la sortie de l'algorithme et combien de fois la boucle While est-elle executée ?

2) En générale quelle est la sortie de cette algorithme ?

In [ ]: