DS n° I Informatique

lycée A. Brizeux
M.Roger 2020-2021

Durée : 1h. Les calculatrices sont **interdites**

Exercice 1 *Puissances rapides*

On considère les fonctions récursives `c1(n)`, `c2(n)`, `c3(n)` dépendant d'un entier $n \geq 0$ ci-après.

```
def c1(n):
    if n>0:
        return(c1(n-1)**2)
```

```
def c2(n):
    if n==0:
        return(2)
    else:
        return(c2(n-1)*c2(n-1))
```

```
def c3(n):
    if n==0:
        return(2)
    else:
        return(c3(n-1)**2)
```

1. Quelles est (sont) les fonctions récursives ?
2. Quelles est (sont) les fonction(s) qui ne termine(nt) pas ?
3. Parmi la (les) fonction(s) c_i qui terminent, compléter le tableau

n	0	1	2	3	4
$c_i(n)$	...	...	...	...	...

4. en déduire les fonctions qui renvoient le calcul de $2^{(2^n)}$
5. Pour chaque fonction c_i qui calcule $2^{(2^n)}$, exprimer en fonction de n la complexité $C_i(n)$ en nombre de multiplications.

Exercice 2 *Liste des carrés d'entiers naturels*

Un étudiant souhaite programmer une fonction `listecarres(N)` en Python qui construit et renvoie une liste L contenant les carrés des entiers de 0 à N.

Il propose le code suivant :

```
def listecarres(N):
    L=[]
    for i in range(N):
        L.append(i**2)
        return(L)
```

Son algorithme est-il correct, et si non, quelle(s) est(sont) la(les) erreur(s) commise(s) ?

Exercice 3 *Fonction mystere*

```
def mystere(n):
    if n == 1:
        return 1
    else:
        return 1/n**2 + mystere(n-1)
```

1. Que renvoie `mystere(2)` ?
2. Que calcule `mystere(n)`, pour $n \geq 1$ entier ?

Exercice 4 *Algorithmes de contrôle*

Une liste X est trié par ordre croissant si $X[i] \leq X[i+1]$ pour tout $i \in [\![0, n-1]\!]$, où n est le nombre d'éléments de X.

On va étudier plusieurs algorithmes permettant de tester si une liste est triée.

1. (a) On considère l'algorithme suivant :

```
def testi(X):
    n=len(X)
    res=True
    for i in range(n-1):
        if X[i]>X[i+1]:
            res=False
    return(res)
```

Justifier que cet algorithme est correct, c'est à dire qu'il renvoie `True` si la liste X est triée par ordre croissant, `False` sinon.

(b) Quel est sa complexité A_N en nombre de comparaisons pour une liste de longueur N ?

2. (a) Si X est une liste de longueur $N \geq 2$, que représentent les commandes Python `X[0:N//2]` et `X[N//2:]` ?

(b) Elaborer un algorithme récursif `testr(X)` permettant de vérifier qu'un tableau X est trié ou non, pour lequel, si la liste X a une longueur N supérieure ou égale à 2, on appelle récursivement `testr(X[0:N//2])` et `testr(X[N//2:])`.

(c) Estimer sa complexité B_N, en nombre de comparaisons, pour une liste de longueur $N = 2^p$, avec $p \in \mathbb{N}^*$.

3. Commentaire ?