TELECOM Nancy

MODULE CS54
Examen du 14 décembre 2022 (2h)
Module

UNIVERSITÉ DE LORRAINE

N'essayez pas nécessairement de tout faire. Lisez bien l'intégralité du sujet avant de choisir vos exercices. La notation tiendra compte de la validité des réponses, mais aussi de la présentation et de la clarté de la rédaction.

Documents interdits.

★ Questions de cours. (2 pts)

▷ **Question 1:** À quelles classes de complexité (en notation Θ) appartiennent les algorithmes 1 et 2 suivants ? (2 pts)

algorithme 1

```
1 def veryLogique(a:int,b:int)->int:
2     res = 0
3     for i in range(1,a+1):
4         for j in range(i,a):
5             res =res+1
6         for k in range(i,b):
7             res =res+1
8     return res
```

algorithme 2

```
1  def NoComplexe(l:list[int],v:int):
2      res = 0
3      for i in range (1,v):
4          if (i in l):
5              res= res+1
6      t = res
7      c = 2
8      while t !=0:
9          c = c*c
10     return c
```

★ Exercice 1: Les Guirlandes de Noël (8 pts).

La ville de Nancy a déployé une immense guirlande de Noël dans toute la ville. Constituée de plusieurs millions de leds séparées les unes des autres de quelques centimètres, cette guirlande peut être pilotée à distance (chaque led peut-être allumée ou éteinte séparément) par un programme informatique. Dans ce programme, l'état d'une guirlande est représenté par un tableau contenant pour chaque led i la valeur 0 si la led est éteinte, 1 si la led est activée. Lorsque la guirlande est en fonctionnement, elle affiche des séquences de lumière. Une séquence est définie comme une suite contigüe de leds allumées. Deux séquences sont séparées par au moins une led éteinte. Par exemple, la configuration ci-dessous : `[0,1,1,1,0,0,1,0,0,0,1,1,0,0,1,1,1,0,0,0,0]` comprend 4 séquences (`[1,1,1]`,`[1]`,`[1,1]`,`[1,1,1]`)

En 2022, la ville décide d'aller encore plus loin et d'enchanter un peu plus les nancéennes et les nancéens. Pour cela elle décide que cette année, une couleur différente sera associée à chaque séquence de led allumées (au sein d'une séquence, toutes les leds affichent la même couleur).

▷ **Question 1:** (6 pts)

On vous demande d'automatiser l'allocation des couleurs aux séquences. En supposant que vous disposer d'une palette de couleurs dans laquelle chaque couleur est associée à un entier positif supérieur ou égal à 2 et strictement inférieur à 256, écrivez une fonction **récursive** qui renvoie une liste de leds avec pour chaque led, la couleur que celle-ci doit porter. Attention, le maire vient de nous informer en dernière minute que, pour maintenir l'harmonie visuelle de la Place Stanislas, deux séquences consécutives ne peuvent se voir affecter la même couleur.

Par exemple, le résultat de la fonction sur la configuration présentée ci-dessus pourrait être par exemple :

`[0,2,2,2,0,0,3,0,0,0,4,4,0,0,2,2,2,0,0,0,0]`

Commencez par donner et motiver les paramètres essentiels de votre fonction avant d'en donner l'algorithme.

▷ **Question 2:** (2 pts)

En vous appuyant sur les mécanismes de dérécursivation présentés en cours et mis en oeuvre en TD, donnez la version dérécursivée de la fonction que vous venez de construire (NB la fonction dérécursivée ne sera considérée uniquement si la version récursive fonctionne). **Attention, il ne s'agit pas de construire à la main une version itérative de l"algorithme mais bien d'appliquer STRICTEMENT les patrons de traduction vus en cours.**

★ **Exercice 2: LastChristmas** (5 pts) – d'après un exercice de Maylis Delest).

On considère la fonction ci-contre avec la liste `A` de taille 8 contenant la séquence {2,8,7,4,3,6,5,1}.

```
def lastXChristmas(l:list[int], cle:int) -> int:
    d=0
    f=len(l)-1
    while (d<f):
        while (d<len(l) and l[d] <= cle):
            d=d+1
        print(d)
        while (f>=0 and l[f]>cle):
            f=f-1
        if (d<f):
            temp= l[d]
            l[d]= l[f]
            l[f]= temp
            d = d+1
            f = f-1
    return f
```

▷ **Question 1:** (2pt) Représentez graphiquement l'effet de l'appel `lastChristmas(A, 6)` sur la liste A en détaillant les différentes étapes et indiquez la valeur retournée par la fonction.

▷ **Question 2:** (1pt) Expliquez en 3 phrases ce que fait cette fonction

▷ **Question 3:** (1pt) Expliquez le comportement de la fonction sur des cas limites

▷ **Question 4:** (1 pt) Donnez la complexité maximale de cette fonction en nombre de tests, et justifiez votre réponse.

★ **Exercice 3: LastChristmas bis** (5 pts) – d'après The Boss).

```
def JFNQ(n:int) -> int :
    if (n == 0):
      return 1
    elif (n == 1):
      return 1
    elif (n == 2):
      return 2
    elif (n%2 == 1 and n >= 3):
      return JFNQ((n-1)//2) + JFNQ((n-1)//2-1) + 1
    elif (n%2 == 0 and n >= 4):
      return JFNQ(n//2) + JFNQ(n//2+1) + 1
```

▷ **Question 1:** (2 pts) Dessinez l'ordre et l'imbrication des appels à la fonction dans le cas de l'appel `JFNQ(9)`.

▷ **Question 2:** (1 pt) Combien d'appels à la fonction sont effectués pour `JFNQ(9)`. Expliquez votre réponse.

▷ **Question 3:** (1 pt) En observant le résultat sur les 8 premiers entiers (1 à 8), que pouvez vous dire sur ce que calcule `JFNQ`

▷ **Question 4:** (1pt) Quelle est la complexité (en notation Θ) de cette fonction ? Justifiez votre réponse