EXAMEN 2nde SESSION JUIN 2024

Documents autorisés : une feuille A4 manuscrite recto-verso

Ex 1 (1,5 pt)

Donner sans justifier la classe de complexité temporelle de chacune des fonctions suivantes :

```
def f4(m,n):
    x = 0
    for i in range(m):
        for j in range(n,0,-1):
            x = x+1
    return x
```

```
def f5(n):
    x = 0
    while n>0:
        x = x+1
        n = n//10
    return x
```

```
def f7(n):
    x = 0
    m=1
    while m<=n:
        x = x+1
        m = m*2
    return x
```

Ex 2 (3 pts) : Sort or not sort ?

Si une liste de *n* nombres n'est pas triée, on dispose de deux stratégies pour y trouver un élément :
(a) On commence par trier la liste, puis on utilise une recherche dichotomique sur la liste triée.
(b) On parcourt linéairement tous les éléments de la liste. On s'arrête dès qu'on a trouvé l'élément cherché.
1) Quels sont les avantages et les inconvénients respectifs de ces deux méthodes de recherche, du point de vue de l'efficacité, dans le cas général, mais aussi dans le "pire" et le "meilleur" cas ?
2) Dans quel contexte est-il préférable d'utiliser plutôt la méthode (a) ? Est-elle toujours applicable ?

Ex 3 (3 pts) : Voyage à Syracuse

Lors d'une fête foraine, le manège "Voyage à Syracuse" propose d'effectuer un vol en avion selon un plan de vol défini de la manière suivante :
Le client donne un nombre *n* correspondant à son altitude de départ, puis l'altitude *n* va évoluer de la manière suivante :
- si *n* est pair, l'altitude diminue de moitié (*n* devient *n*/2)
- sinon l'altitude triple puis augmente de 1 (*n* devient 3*n*+1)

Ces changements d'altitude s'opèrent les uns à la suite des autres tant que l'altitude *n* n'a pas atteint la valeur 1 (retour au sol).
La suite d'altitudes obtenue à partir d'un entier *n* est appelée son "vol", et le nombre de termes de la suite avant d'atterrir à 1 est appelé "temps de vol".
Exemple : le vol de 15 est : 15 → 46 → 23 → 70 → 35 → 106 → 53 → 160 → 80 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1. Le temps de vol de 15 est donc 17.
Ecrire une fonction Python **récursive** qui renvoie le temps de vol d'une altitude de départ *n* donnée en paramètre.

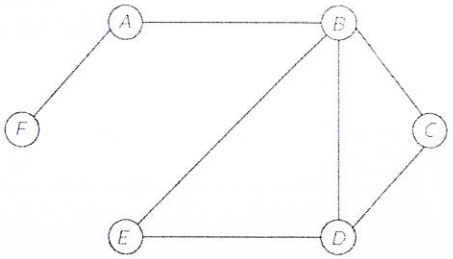
Ex 4 (3 pts) : Parenthésé ou pas ?

Ecrire une fonction Python `est_bien_par(s)` qui indique si une chaîne de caractères `s` est composée uniquement de parenthèses ouvrantes '(' et fermantes ')', et correspond à une séquence "bien parenthésée" (comme vu en CM et en TD-TP). La fonction renverra la valeur `True` ou `False`, sans affichage supplémentaire. Elle pourra si-besoin utiliser une structure de données implémentée en TP.

Ex 5 (3,5 pts) : Parcours eulérien

On s'intéresse ici à la question : peut-on parcourir avec un crayon toutes les arêtes d'un graphe connexe en parcourant chaque arête une seule fois, sans lever le crayon ?

Euler a montré qu'un graphe connexe possède un tel parcours si *tous ses sommets sont de degré pair (un nombre pair d'arêtes part de chaque sommet), sauf éventuellement deux d'entre eux*.



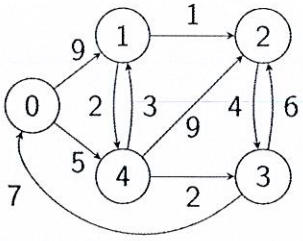
1) Donner la liste d'adjacence (sous forme de dictionnaire) du graphe non-orienté ci-contre.

2) Possède-t-il un tel parcours ? Si oui, donner un exemple d'ordre des sommets rencontrés lors de ce parcours.

2) Donner une fonction Python prenant en paramètre la liste d'adjacence représentant un graphe et indiquant si oui ou non ce graphe possède un tel parcours.

Ex 6 (3 pts) : Graphe orienté pondéré

Effectuer "à la main" le déroulement de l'algorithme de Dijkstra sur le graphe orienté pondéré ci-contre, en donnant les distances de poids minimal de chaque sommet au sommet 0, ainsi que le chemin associé. On utilisera un tableau, comme fait en CM.



Ex 7 (3 pts) : Gloutonnerie

On dispose de n skieurs représentés par leurs tailles t_1 , ... , t_n et de n paires de skis de tailles s_1 , ... , s_n.

1) En une ou deux phrases, donner une stratégie gloutonne qui permet d'associer une paire de skis à chaque skieur de sorte que la différence moyenne entre les tailles des skieurs et les tailles de leurs skis soit minimale. On n'impose pas le fait que cette stratégie soit optimale.

2) Donner une fonction Python implémentant cette stratégie. Elle prendra en paramètre 2 liste d'entiers (tailles des skieurs et tailles des skis) et renverra une liste de couples (taille d'un skieur, taille de ses skis).