Devoir surveillé d'informatique

Remarques et consignes importantes

- On pourra toujours librement utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée.
- Veillez à présenter vos idées et vos réponses partielles même si vous ne trouvez pas la solution complète à une question.
- La clarté et la lisibilité de la rédaction et des programmes sont des éléments de notation.

❑ Exercice 1 : *Programmes divers et saut de taille maximale* 🎓 d'après CAPES 2023 (Partie 1)

Notes de programmation : Vous disposez pour répondre aux questions de cet exercice des fonctions Python de manipulation de listes suivantes :

- On peut créer une liste de taille n remplie avec la valeur x avec `li = [x] * n`
- On peut obtenir la taille d'une liste `li` avec `len(li)`.
- Si `li` est une liste de n éléments, on peut accéder au k-ème élément (pour $0 \leq k <$ `len(li)`) avec li[k]. On peut définir sa valeur avec li[k] = x.
- On peut concaténer deux listes `li1` et `li2` en utilisant l'opération `li1 + li2`. On utilisera aussi cette opération dans des expressions mathématiques.
- `li[a:b]` désigne la liste des éléments d'indice compris entre a et $b-1$ dans `li`. On utilisera aussi cette opération dans des expressions mathématiques.

Les autres fonctions sur les listes (`sort`, `index`, `max`, etc.) sont interdites à moins de les réécrire explicitement. L'opérateur `in` d'appartenance à une liste est interdit, mais on peut utiliser ce mot-clé dans les autres contextes (par exemple dans une boucle `for`).

Complexité : Par *complexité* d'un algorithme, on entend le nombre d'opérations élémentaires nécessaires à l'exécution de cet algorithme dans le pire cas. Lorsque cette complexité dépend d'un ou plusieurs paramètres $k_0, \ldots, k_{r-1}$, on dit que la complexité est $\mathcal{O}(f(k_0, \ldots, k_{r-1}))$ s'il existe une constante $C > 0$ telle que, pour toutes les valeurs $k_0, \ldots, k_{r-1}$ suffisamment grandes, ce nombre d'opérations élémentaires est majoré par $C \times f(k_0, \ldots, k_{r-1})$.

■ Partie I : Programmes divers

Q1– Ecrire une fonction `fibonacci` qui prend en argument un entier `n` supérieur ou égal à 2 et renvoie la liste des `n` premiers termes de la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ définie par $F_0 = 0$, $F_1 = 1$ et $\forall n \geq 2, F_n = F_{n-1} + F_{n-2}$ (chaque terme est la somme des deux précédents).

```python
def fibonacci(n):
    termes = [0, 1]
    for i in range(n-2):
        termes.append(termes[-1]+termes[-2])
    return termes
```

Q2– Ecrire une fonction `indice_min` qui prend en argument une liste d'entiers `li` et renvoie l'indice d'un de ses minimums.

```python
def indice_min(entiers):
    mini, indice_min = entiers[0], 0
    for i in range(1, len(entiers)):
        if entiers[i] < mini:
            mini, indice_min = entiers[i], i
    return indice_min
```

Q3– Que renverra `indice_min([1, 0, 2, 0])` avec votre programme ?

> Dans la fonction précédente le minimum (et son indice) ne sont mis à jour qu'en cas d'infériorité strict, donc la fonction affichera l'indice du premier minimum c'est-à-dire 1.

Q4– Ecrire une fonction `lettre_majoritaire` qui prend en argument une chaîne de caractères non vide et renvoie le caractère qui apparait le plus fréquemment. Ainsi, `lettre_majoritaire('abcdedde')` devrait renvoyer `'d'`.
Note : l'utilisation efficace d'un dictionnaire sera valorisée. On pourra alors utiliser l'opérateur `in`

```
def lettre_majoritaire(chaine):
    occurrences = {}
    for c in chaine:
        if c in occurrences:
            occurrences[c] += 1
        else:
            occurrences[c] = 1
    maxl = 0
    for c in occurrences:
        if occurrences[c] > maxl:
            maxl,lm = occurrences[c], c
    return lm
```

■ **Partie II** : Saut de valeur maximale Dans une liste de flottants `li`, on appelle *saut* un couple (i, j) avec $0 \leq i \leq j <$ `len(li)` et la *valeur* d'un saut est la valeur `li[j]-li[i]`. On va ici programmer plusieurs manières de trouver un saut de valeur maximale dans une liste. Par exemple, dans la liste `[2.0, 0.2, 3.0, 5.3, 2.0]`, un tel saut est `(1, 3)` (car `0.2` et `5.3` sont aux indices 1 et 3 respectivement).

Q5– Ecrire une fonction `valeur` qui prend en argument une liste et un saut et renvoie la valeur de ce saut. Par exemple `valeur([2.0, 0.2, 3.0, 5.3, 2.0],(0,2))` renvoie `1.0` (car `li[2]-li[0] = 1.0`).

```
def valeur(li,saut):
    i,j = saut
    return li[j]-li[i]
```

Q6– Donner un exemple de liste avec exactement deux sauts de valeur maximale et préciser ces sauts.

> La liste `[2, 6, 1, 5]` possède deux sauts de valeurs maximale : `(0,1)` et `(2,3)` (ces deux sauts ont une valeur de 4)

Q7– À l'aide d'un contre-exemple, montrer qu'on ne peut pas se contenter de chercher le minimum et le maximum d'une liste pour trouver un saut de valeur maximale.

> Dans la liste `[2, 6, 1, 5]` le minimum est à l'indice 2 (c'est 1) et le maximum à l'indice 1 (c'est 3) et comme le minimum est après le maximum ce n'est pas le saut maximal.

Q8– Écrire une fonction `saut_max_naif` qui renvoie un saut de valeur maximale en testant tous les couples (i, j) tels que $0 \leq i \leq j <$ `len(li)`.

```
def saut_max_naif(li):
    vsaut, imax, jmax = 0, 0, 0
    for i in range(len(li)):
        for j in range(i, len(li)):
            if valeur(li, (i,j)) > vsaut:
                vsaut, imax, jmax = valeur(li,(i,j)), i, j
    return (imax,jmax)
```

On décrit ici un algorithme utilisant le paradigme de la programmation dynamique pour résoudre ce problème : pour chaque k entre 1 et `len(li)`, on va calculer m_k l'indice du minimum de `li[0:k]`, et le couple (i_k, j_k) un saut de valeur maximale dans `li[0:k]`. Ainsi, on aura $m_1 = i_1 = j_1 = 0$ car `li[0:1]` ne comporte qu'un seul élément.

Q9– Pour $k <$`len(li)`, expliquer comment on peut calculer efficacement m_{k+1} à partir de m_k et des valeurs dans `li`.

On sait que $(i_k; j_k)$ est le saut maximal de `li[0:k]` pour déterminer le saut de valeur maximal dans `li[0:k+1]` on doit prendre le saut maximal entre : (i_k, j_k) et le nouveau saut maximal disponible qui est (mk, k) ce dernier a pour valeur li[k]-li[m_k]. On doit donc comparer la valeur de ce saut à la valeur du saut (i_k, j_k) ce qui correspond bien à la relation précédente.

Q10– Justifier que la relation suivante est correcte.

$$(i_{k+1}, j_{k+1}) = \begin{cases} (i_k, j_k) \text{ si } \texttt{li[}k\texttt{]-li[}m_k\texttt{]} < \texttt{li[}j_k\texttt{]-li[}i_k\texttt{]} \\ (m_k, k) \text{ sinon} \end{cases}$$

On sait que $(i_k; j_k)$ est le saut maximal de `li[0:k]` pour déterminer le saut de valeur maximal dans `li[0:k+1]` on doit prendre le saut maximal entre : (i_k, j_k) et le nouveau saut maximal disponible qui est (mk, k) ce dernier a pour valeur li[k]-li[m_k]. On doit donc comparer la valeur de ce saut à la valeur du saut (i_k, j_k) ce qui correspond bien à la relation précédente.

Q11– Ecrire une fonction `saut_max_dynamique` qui prend en argument une liste `li` et renvoie un saut de valeur maximale en utilisant la relation de la question précédente.

```
def saut_max_dynamique(li):
    mk, ik, jk = 0, 0, 0
    for k in range(1,len(li)):
        if li[k]<li[mk]:
            mk = k
        if valeur(li,(ik,jk))<li[k]-li[mk]:
            ik, jk = mk, k
    return (ik, jk)
```

Q12– Déterminer la complexité de votre programme dans le pire cas, puis comparer cette complexité avec celle de la fonction `saut_max_naif`

En notant n la longueur de la liste, la boucle est executée $n-1$ fois et ne contient que des opérations élémentaires donc la complexité est en $O(n)$. Pour le programme de la question 4, par contre on a deux boucles imbriquées et donc une complexité en $O(n^2)$.

❑ Exercice 2 : *Base de données et* SQL

CAPES NSI 2021, épreuve 1

On s'intéresse dans cette partie à un site Internet d'échange de supports de cours entre enseignants de MP2I/MPI. Chaque personne désirant proposer ou récupérer du contenu doit commencer par se créer un compte sur ce site et peut ensuite accéder à du contenu ou en proposer.

Ce site repose sur une base de données contenant en particulier une table, nommée `ressources`. Elle possède un enregistrement par document téléversé sur le site. Ses attributs sont :

- `id`, un identifiant numérique, unique pour chaque ressource ;
- `owner`, le pseudo de la personne ayant créé la ressource ;
- `annee`, l'année de publication de la ressource ;
- `titre`, une chaine de caractères décrivant la ressource ;
- `type`, chaine de caractères pouvant être cours, ds, tp ou td.

Voici un extrait de cette table :

id	owner	annee	titre	type
4	dknuth	2020	Machine à décalage	cours
13	alovelace	2022	Intelligence artificielle	td
...	...	...	...	...

Q13– Écrire une requête SQL permettant de connaitre tous les titres des ressources déposées par « `jclarke` » classées par année de publication croissante.

```
SELECT titre
FROM ressources
WHERE owner="jclarke"
ORDER BY annee ASC;
```

Q14– Écrire une requête SQL permettant de connaitre le nombre total de ressources de type cours présentes sur le site.

```
SELECT COUNT(*)
FROM ressources
WHERE type = "cours";
```

Q15– Que fait la requête suivante : ?

```
SELECT R.owner
FROM Ressources AS R
WHERE R.type = 'td'
GROUP BY R.owner
ORDER BY COUNT(*) DESC
LIMIT 3
```

> Cette requête permet d'afficher les trois premiers propriétaires de ressources ayant posté le plus de ressources de type `td`

Cette base de données contient également une table `utilisateurs` qui contient les informations sur les utilisateurs du site. Elle possède un enregistrement par utilisateur. Ses attributs sont :

— `nom`, le nom de l'utilisateur (clé primaire) ;
— `mdp`, le mot de passe de l'utilisateur.
— `email`, l'adresse email de l'utilisateur.
— `naissance`, l'année de naissance de l'utilisateur.

Voici un extrait de cette table :

nom	mdp	email	naissance
dknuth	chepas123	dknuth@bigboss.com	1938
...	...	...	...

L'attribut `owner` de la table `ressources` est une clé étrangère qui référence l'attribut `nom` de la table `utilisateurs`.

Q16– Ecrire une requête SQL permettant de lister toutes les ressources déposées par des utilisateurs nés après 2000.

```
SELECT * FROM ressources
JOIN utilisateurs
ON ressources.owner = utilisateurs.nom
WHERE utilisateurs.naissance>2000
```

Q17– Ecrire une requête SQL permettant de lister tous les utilisateurs n'ayant déposé aucune ressource.

```
SELECT nom FROM utilisateurs
EXCEPT
SELECT owner FROM ressources
```