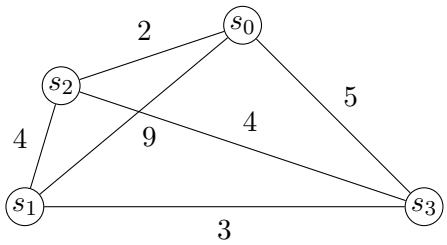
Devoir surveillé d'informatique

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 : *Stratégie gloutonne pour le problème du voyageur de commerce*

On considère un graphe *complet* (c'est à dire dans lequel il existe un arc entre toutes les paires de sommet) non orienté, pondéré, et on s'intéresse au problème du voyageur de commerce c'est à dire à la recherche d'un chemin de longueur minimale qui démarre d'un sommet donné, passe par tous les sommets et revient à son point de départ. Dans toutes la suite du problème, on note, N le nombre de sommet du graphe et $(s_0, \dots, s_{N-1})$ les sommets et on supposera que le sommet de départ est s_0. Par exemple si on considère le graphe suivant :



Le problème est de trouver la longueur d'un chemin de longueur minimal partant de s_0, passant par tous les autres sommets et revenant à s_0. Dans les questions de programmation, on suppose que les graphes sont représentés par matrice d'adjacence en utilisant des listes de listes de Python, par exemple le graphe donné en exemple est représenté par la matrice `[[0, 9, 2, 5], [9, 0, 4, 3], [2, 4, 0, 4], [5, 3, 4, 0]]`.

Q1– Montrer que dans l'exemple précédent, la longueur minimale d'un chemin solution est 14 et donner un chemin réalisant ce minimum.

Q2– Ecrire une fonction de signature `longueur(mat_adj, chemin)` qui prend en argument un graphe pondéré représenté par sa matrice d'adjacence `mat_adj` ainsi qu'un chemin (représenté par une liste de sommet) et calcule la longueur de ce chemin. Par exemple si `mat_adj` est la matrice du graphe précédent alors `longueur{mat_adj,[0,1,3,2,0]}` renvoie 9+3+4+2=16.

Q3– Déterminer la complexité de la longueur de la fonction `longueur` en fonction de la longueur de la liste chemin. En déduire la complexité en fonction de N de la méthode par force brute qui consiste à énumérer tous les chemins possibles partant et revenant à s_0 et passant par tous les autres sommets, à calculer leur longueur afin d'en trouver un de longueur minimale.

Q4– On propose à présent de mettre en place la stratégie de résolution gloutonne suivante : on part de s_0 et à chaque étape on se dirige vers le sommet *non encore parcourue* le plus proche, et on revient à s_0 une fois tous les sommets parcourus. Quel est le chemin obtenu sur le graphe donné en exemple ?

Q5– En dessinant un exemple montrer que cette stratégie ne conduit pas forcément à la solution optimale.

Q6– Ecrire une fonction de signature `glouton(mat_adj)` qui prend en argument une matrice d'adjacence représentant un graphe et qui renvoie le chemin obtenu en utilisant la stratégie gloutonne.

❑ 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
...	...	...	...	...

Q7– Écrire une requête SQL permettant de connaitre tous les titres des ressources déposées par « `jclarke` » classées par année de publication croissante.

Q8– Écrire une requête SQL permettant de connaitre le nombre total de ressources de type cours présentes sur le site.

Q9– Que fait la requête suivante : ?

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

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`.

Q10– Ecrire une requête SQL permettant de lister toutes les ressources déposées par des utilisateurs nés après 2000.

Q11– Ecrire une requête SQL permettant de lister tous les utilisateurs n'ayant déposé aucune ressource.

❑ Exercice 3 : *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 $k_0, \dots, k_{r-1}$, on dit que la complexité est $\mathcal{O}(f(k_0, \dots, k_{r-1}))$ s'il existe une constante $C > 0$ telle que, pour toutes les valeurs $k_0, \dots, k_{r-1}$ suffisamment grandes, ce nombre d'opérations élémentaires est majoré par $C \times f(k_0, \dots, k_{r-1})$.

■ **Partie I** : Programmes divers

Q12– 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).

Q13– Ecrire une fonction `indice_min` qui prend en argument une liste d'entiers `li` et renvoie l'indice d'un de ses minimums.

Q14– Que renverra `indice_min([1, 0, 2, 0])` avec votre programme ?

Q15– 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`

■ **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).

Q16– 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`).

Q17– Donner un exemple de liste avec exactement deux sauts de valeur maximale et préciser ces sauts.

Q18– 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.

Q19– 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)`.

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.

Q20– Pour $k <$`len(li)`, expliquer comment on peut calculer efficacement m_{k+1} à partir de m_k et des valeurs dans `li`.

Q21– Justifier que la relation suivante est correcte.

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

Q22– 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 6.

Q23– Déterminer la complexité de votre programme dans le pire cas, puis comparer cette complexité avec celle du programme donnée en question 4