EXERCICE 3 (8 points)

Cet exercice porte sur la programmation orientée objet, les graphes et utilise la structure de données dictionnaire.

La direction de la station de ski *Le Lièvre Blanc*, spécialisée dans la pratique du ski de fond, souhaite disposer d'un logiciel lui permettant de gérer au mieux son domaine skiable. Elle confie à un développeur informatique la mission de concevoir ce logiciel. Celui-ci décide de caractériser les pistes de ski à l'aide d'une classe `Piste` et le domaine de ski par une classe `Domaine`.

Le code Python de ces deux classes est donné en **Annexe**.

Partie A – Analyse des classes `Piste` et `Domaine`

1. Lister les attributs de la classe `Piste` en précisant leur type.

La difficulté des pistes de ski de fond est représentée par 4 couleurs : verte, bleue, rouge et noire. La piste verte est considérée comme très facile, la piste bleue comme facile, la piste rouge de difficulté moyenne et la piste noire difficile. Dans la station de ski *Le Lièvre blanc*, l'équipe de direction décide de s'appuyer uniquement sur le dénivelé pour attribuer la couleur d'une piste de ski.

Ainsi, une piste de ski sera de couleur :

- `'noire'` si son dénivelé est supérieur ou égal à 100 mètres ;
- `'rouge'` si son dénivelé est strictement inférieur à 100 mètres, mais supérieur ou égal à 70 mètres ;
- `'bleue'` si son dénivelé est strictement inférieur à 70 mètres, mais supérieur ou égal à 40 mètres ;
- `'verte'` si son dénivelé est strictement inférieur à 40 mètres.

2. Écrire la méthode `set_couleur` de la classe `Piste` qui permet d'affecter à l'attribut `couleur` la chaîne de caractères correspondant à la couleur de la piste.

On exécute à présent le programme suivant afin d'attribuer la couleur adéquate à chacune des pistes du domaine skiable *Le Lièvre Blanc*.

```
1 for piste in lievre_blanc.get_pistes():
2     piste.set_couleur()
```

3. Indiquer, parmi les 4 propositions ci-dessous, le type de l'élément renvoyé par l'instruction Python `lievre_blanc.get_pistes()`.

   - Proposition A : une chaîne de caractères ;
   - Proposition B : un objet de type `Piste` ;
   - Proposition C : une liste de chaînes de caractères ;
   - Proposition D : une liste d'objets de type `Piste`.

En raison d'un manque d'enneigement, la direction de la station est souvent contrainte de fermer toutes les pistes vertes car elles sont situées généralement en bas du domaine.

4. Écrire un programme Python dont l'exécution permet de procéder à la fermeture de toutes les pistes vertes en affectant la valeur `False` à l'attribut `ouverte` des pistes concernées.

5. Écrire une fonction `pistes_de_couleur` prenant pour paramètres une chaîne de caractères `couleur` représentant la difficulté d'une piste et une liste `lst` de pistes de ski de fond. Cette fonction renvoie la liste des noms des pistes dont `couleur` est le niveau de difficulté.

   Exemple : l'instruction `pistes_de_couleur(lievre_blanc.get_pistes(), 'noire')` renvoie la liste `['Petit Bonheur', 'Forêt', 'Duvallon']`.

Un skieur de bon niveau se prépare assidûment pour le prochain semi-marathon, d'une distance de 21,1 kilomètres. À chaque entraînement, il note la liste des noms des pistes qu'il a parcourues et il souhaite disposer d'un outil lui indiquant si la distance totale parcourue est au moins égale à la distance qu'il devra parcourir le jour du semi-marathon.

La fonction `semi_marathon` donnée ci-dessous répond aux attentes du skieur : cette fonction prend en paramètre une liste `L` de noms de pistes et renvoie un booléen égal à `True` si la distance totale parcourue est strictement supérieure à 21,1 kilomètres, `False` sinon.

```
1  def semi_marathon(L):
2      distance = ...
3      liste_pistes = lievre_blanc.get_pistes()
4      for nom in L:
5          for piste in liste_pistes:
6              if piste.get_nom() == ...:
7                  distance = distance + ...
8      return ...
```

On donne ci-dessous deux exemples d'appels à cette fonction :

```
>>> entrainement1 = ['Verneys', 'Chateau enneigé', 'Rois mages',
'Diablotin']
>>> semi_marathon(entrainement1)
True
>>> entrainement2 = ['Esseillon', 'Aigle Royal', 'Duvallon']
>>> semi_marathon(entrainement2)
False
```

6. Recopier et compléter la fonction `semi_marathon`.

Partie B – Recherche par force brute

Le plan des pistes du domaine *Le Lièvre Blanc* peut être représenté par le graphe suivant :

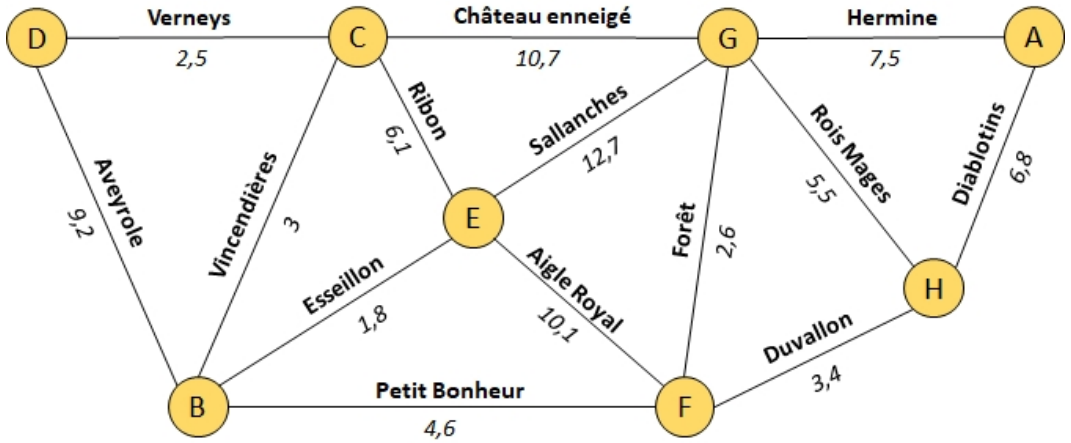

Figure 1. Graphe du domaine *Le Lièvre Blanc*

Sur chaque arête, on a indiqué le nom de la piste et sa longueur en kilomètres. Les sommets correspondent à des postes de secours.

Un pisteur-secouriste de permanence au point de secours D est appelé pour une intervention en urgence au point de secours A. La motoneige de la station étant en panne, il ne peut s'y rendre qu'en skis de fond. Il décide de minimiser la distance parcourue et cherche à savoir quel est le meilleur parcours possible. Pour l'aider à répondre à ce problème, on décide d'implémenter le graphe ci-dessus grâce au dictionnaire de dictionnaires suivant :

```
domaine = {'A' : {'G' : 7.5, 'H' : 6.8},
           'B' : {'C' : 3.0, 'D' : 9.2, 'E' : 1.8, 'F' : 4.6},
           'C' : {'B' : 3.0, 'D' : 2.5, 'E' : 6.1, 'G' : 10.7},
           'D' : {'B' : 9.2, 'C' : 2.5},
           'E' : {'B' : 1.8, 'C' : 6.1, 'F' : 10.1, 'G' : 12.7},
           'F' : {'B' : 4.6, 'E' : 10.1, 'G' : 2.6, 'H' : 3.4},
           'G' : {'A' : 7.5, 'C' : 10.7, 'E' : 12.7, 'F' : 2.6,
                  'H' : 5.5},
           'H' : {'A' : 6.8, 'F' : 3.4, 'G' : 5.5} }
```

7. Écrire une instruction Python permettant d'afficher la longueur de la piste allant du sommet `'E'` au sommet `'F'`.

8. Écrire une fonction `voisins` qui prend en paramètres un graphe `G` et un sommet `s` du graphe `G` et qui renvoie la liste des voisins du sommet `s`.

   Exemple : l'instruction `voisins(domaine, 'B')` renvoie la liste `['C', 'D', 'E', 'F']`.

9. Recopier et compléter la fonction `longueur_chemin` donnée ci-dessous : cette fonction prend en paramètres un graphe `G` et un chemin du graphe `G` sous la forme d'une liste de sommets et renvoie sa longueur en kilomètres.

   Exemple : l'instruction `longueur_chemin(domaine, ['B', 'E', 'F', 'H'])` renvoie le nombre flottant `15.3`.

```
def longueur_chemin(G, chemin):
    precedent = ...
    longueur = 0
    for i in range(1, len(chemin)):
        longueur = longueur + ...
        precedent = ...
    return ...
```

On donne ci-dessous une fonction `parcours` qui renvoie la liste de tous les chemins du graphe `G` partant du sommet `depart` et parcourant les sommets de façon unique, c'est-à-dire qu'un sommet est atteint au plus une fois dans un chemin.

Par exemple, l'appel `parcours(domaine, 'A')` renvoie la liste de tous les chemins partant du sommet A dans le graphe `domaine` sans se soucier ni de la longueur du chemin, ni du sommet d'arrivée. Ainsi, `['A', 'G', 'C']` est un chemin possible, tout comme `['A', 'G', 'C', 'B', 'E', 'F', 'H']`.

```
def parcours(G, depart, chemin = [], lst_chemins = []):
    if chemin == []:
        chemin = [depart]
    for sommet in voisins(G, depart):
        if sommet not in chemin:
            lst_chemins.append(chemin + [sommet])
            parcours(G, sommet, chemin + [sommet])
    return lst_chemins
```

10. Expliquer en quoi la fonction `parcours` est une fonction récursive.

Un appel à la fonction `parcours` précédente renvoie une liste de chemins dans laquelle figurent des doublons.

11. Recopier et compléter la fonction `parcours_dep_arr` ci-après qui renvoie la liste des chemins partant du sommet `depart` et se terminant par le sommet `arrivee` dans le graphe `G` entrés en paramètres. La liste renvoyée ne doit pas comporter de doublons. Attention, plusieurs lignes de code sont nécessaires.

```
def parcours_dep_arr(G, depart, arrivee):
    liste = parcours(G, depart)
    ...
```

12. Recopier et compléter la fonction `plus_court` donnée ci-dessous. La fonction `plus_court` prend pour paramètres un graphe `G`, un sommet de départ `depart` et un sommet d'arrivée `arrivee` ; elle renvoie un des chemins les plus courts sous la forme d'une liste de sommets.

```
def plus_court(G, depart, arrivee):
    liste_chemins = parcours_dep_arr(G, depart, arrivee)
    chemin_plus_court = ...
    minimum = longueur_chemin(G, chemin_plus_court)
    for chemin in liste_chemins:
        longueur = longueur_chemin(G, chemin)
        if ...:
            minimum = ...
            chemin_plus_court = ...
    return chemin_plus_court
```

13. Expliquer en quoi le choix fait par le pisteur-secouriste de choisir la distance minimale pour arriver le plus rapidement possible sur le lieu de l'incident est discutable. Proposer un meilleur critère de choix.

Annexe

```
1  # Pistes
2  class Piste:
3      def __init__(self, nom, denivele, longueur):
4          self.nom = nom
5          self.denivele = denivele   # en mètres
6          self.longueur = longueur   # en kilomètres
7          self.couleur = ''
8          self.ouverte = True

9      def get_nom(self):
10         return self.nom

11     def get_longueur(self):
12         return self.longueur

13     def set_couleur(self):
14         # À compléter

15     def get_couleur(self):
16         return self.couleur

17 # Domaine skiable
18 class Domaine:
19     def __init__(self, a):
20         self.nom = a
21         self.pistes = []

22     def ajouter_piste(self, nom_piste, denivele, longueur):
23         self.pistes.append(Piste(nom_piste, denivele, longueur))

24     def get_pistes(self):
25         return self.pistes

26 # Programme principal
27 lievre_blanc = Domaine("Le Lièvre Blanc")
28 lievre_blanc.ajouter_piste('Aveyrole', 62, 9.2)
29 lievre_blanc.ajouter_piste('Verneys', 10, 2.5)
30 lievre_blanc.ajouter_piste('Vincendières', 45, 3)
31 lievre_blanc.ajouter_piste('Ribon', 70, 6.1)
32 lievre_blanc.ajouter_piste('Esseillon', 8, 1.8)
33 lievre_blanc.ajouter_piste('Petit Bonheur', 310, 4.6)
34 lievre_blanc.ajouter_piste('Aigle Royal', 85, 10.1)
35 lievre_blanc.ajouter_piste('Château enneigé', 54, 10.7)
36 lievre_blanc.ajouter_piste('Sallanches', 78, 12.7)
37 lievre_blanc.ajouter_piste('Forêt', 145, 2.6)
```

```
lievre_blanc.ajouter_piste('Hermine', 27, 7.5)
lievre_blanc.ajouter_piste('Rois mages', 42, 5.5)
lievre_blanc.ajouter_piste('Diablotin', 76, 6.8)
lievre_blanc.ajouter_piste('Duvallon', 200, 3.4)
```