EFS ISN3
2ème année - Janvier 2024

Durée totale : *2h*
Documents autorisés : *Synthèse personnelle de 4 pages format A4 (2 feuilles).*

— Le barème est sur 20 points.
— Le sujet est sur 11 pages - il y a 5 exercices.

Exercice 1 : PIX (1 point)

Exercice 2 : APP (Evaluation par les pairs, 1 point)

Exercice 3 : QCM (9 points)

Attention ! Certaines questions admettent plusieurs réponses justes. Il faut toutes les donner. Chaque mauvaise réponse est sanctionnée de **-0.25 point**

3.1 Dictionnaires (2 pts)

Nous considérons un dictionnaire `person` donné ci-dessous.

```
1 person = {
2     "name": "Alice",
3     "age": 25,
4     "city": "Wonderland"
5 }
```

(3.1) Quelles instructions Python permettent d'accéder à l'âge de la personne ? (0.5 pts)

- ☐ `person('age')`
- ☐ `person.get('age', 42)`
- ☐ `person['age']`
- ☐ `person.age`
- ☐ `person[1]`

En utilisant le dictionnaire `person_info` ci-dessous, on affiche à l'écran la phrase `Charlie lives in Happyville`.

```
person_info = {
    "name": "Charlie",
    "address": {
        "street": "123 Main St",
        "city": "Happyville"
    }
}
```

(3.2) Lequel de ces codes fait cet affichage ? (0.5 pts)

- ☐ `person_info['city']`
- ☐ `print(f"{person_info['name']} lives in {person_info['city']}")`
- ☐ `person_info['address']['city']`
- ☐ `print(f"{person_info['name']} lives in {person_info['address']['city']}")`
- ☐ `print(person_info['age'])`

Nous exécutons le code suivant :

```
d = {
    "a" : 3,
    "e" : 5,
    "k" : 2,
    "m" : 6,
    "x" : 1
}
d["y"] = 1
d["k"] = 3
d["o"] = 8
```

(3.3) Quelle est alors la valeur de `len(d)` ? (0.5 pts)

☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ impossible de dire ☐ un message d'erreur

Nous exécutons le code suivant :

```
d = {
    "a" : 3,
    "e" : 5,
    "k" : 2,
    "m" : 6,
    "x" : 1
}

s = 0
for i in range(len(d)):
    s += d[i]

print(s)
```

(3.4) Qu'est-ce qui va être affiché lors de l'exécution ? (0.5pts)

☐ 0 ☐ 5 ☐ 17 ☐ impossible de dire ☐ un message d'erreur

3.2 Propriétés de graphes (3 pts)

Soit $G = (V, E)$ un graphe non orienté.

(3.5) Quelles affirmations suivantes sont vraies ? (1pt)

- ☐ La matrice d'adjacence est toujours symétrique
- ☐ Le degré entrant d'un sommet est égal à son degré sortant
- ☐ Chaque paire de sommets est reliée par une arête dirigée
- ☐ La somme des degrés est toujours paire
- ☐ Le nombre d'arcs est toujours pair
- ☐ Le degré d'un sommet est toujours supérieur au nombre de sommets

Soit $p =< u_0, u_1, ..., u_k >$ un chemin simple dans un graphe non orienté $G = (V, E)$.

(3.6) Quelles affirmations suivantes sont vraies ? (1 pts)

- ☐ La longueur de p est forcément inférieure à $|V|$
- ☐ $\forall i, j \leqslant k, u_i$ et u_j sont connectés
- ☐ $\forall i < k, \{u_i, u_{i+1}\} \in E$
- ☐ $\forall i, j \leqslant k, u_i \neq u_j$

On considère les séquences suivantes : a) 2 3 4 7 b) 3 3 3 3 3 3 c) 2 2 3 3 3 5 d) 2 3 3 3

(3.7) Lesquelles peuvent être les degrés des sommets d'un graphe non-orienté ? (1 pt)

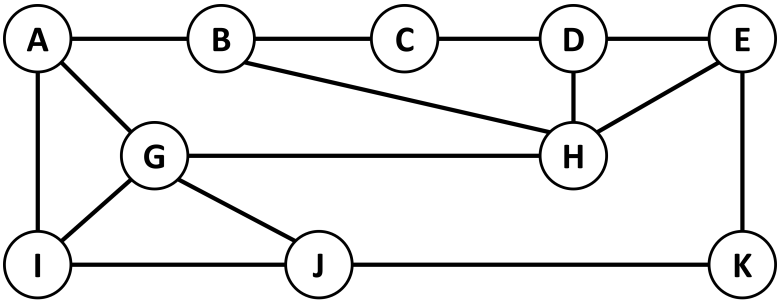
☐ La séquence a) ☐ La séquence b) ☐ La séquence c) ☐ La séquence d)

3.3 Parcours de graphe : BFS (3 pts)

Nous considérons le graphe $Graph = (V, E)$ représenté ci-dessous, avec

$V = \{A, B, C, D, E, G, H, I, J, K\}$

$E = \{(A, B), (A, G), (A, I), (B, C), (B, H), (C, D), (D, E), (D, H), (E, H), (E, K), (G, H), (G, I), (G, J), (I, J), (J, K)\}$.



Lors d'un parcours en largeur, les sommets sont visités dans un certain ordre.

(3.8) Parmi les séquences ci-dessous, indiquez celle qui peut constituer un parcours en largeur partant du sommet G. (1.5 pts)

- ☐ $GAHJIBCEKD$
- ☐ $GAHJIBCDEK$
- ☐ $GAHIJBDEKC$
- ☐ $GAHIJBCEKD$

Soit T l'arbre des prédécesseurs calculé lors de ce parcours en largeur.

(3.9) Parmi les affirmations suivantes, laquelle est vraie ? (0.5 pts)

- ☐ T permet de calculer le plus court chemin entre toute paire de sommets
- ☐ T permet de calculer le plus court chemin entre G et tous les autres sommets
- ☐ T permet de calculer le plus court chemin entre les feuilles de T
- ☐ T permet de calculer le plus long chemin sur le graphe

On considère le parcours en largeur (BFS) du graphe non-orienté $G = (V, E)$ à partir du sommet 1.
$V = \{1, 2, 3, 4, 5, 6, 7\}$ et
$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}, \{4, 6\}, \{5, 6\}, \{6, 7\}\}$.

(3.10) Parmi les arêtes suivantes, lesquelles vont forcément faire partie de l'arbre prédécesseur ? (1 pts)

- ☐ $\{1, 4\}$
- ☐ $\{2, 4\}$
- ☐ $\{2, 5\}$
- ☐ $\{3, 4\}$
- ☐ $\{4, 5\}$
- ☐ $\{5, 6\}$

3.4 Appariement (1 pts)

Soit $G = (A \cup B, E)$ avec $|A| = |B|$ et $E \subset A \times B$ un graphe biparti. On calcule un appariement avec l'algorithme de Gale-Shapley sur ce graphe en prenant en compte des préférences de A vers B et des scores de B vers A.

(3.11) Quelles affirmations sont vraies ? (0.5 pts)

- ☐ Le coût total de l'appariement est minimal
- ☐ L'appariement est un couplage parfait
- ☐ L'appariement est stable
- ☐ Les participants sont toujours appariés à leur choix préféré.

Durant l'algorithme de Gale-Shapley, les éléments de A sont proposés aux éléments de B.

(3.12) Que se passe-t-il si un élément de B reçoit plusieurs propositions ? (0.5 pts)

- ☐ Il rejette toutes les propositions
- ☐ Il accepte la première proposition et rejette les suivantes
- ☐ Il garde la proposition qu'il préfère
- ☐ Il accepte toutes les propositions successivement

Exercice 4 : Réflexions sur les arbres (3 points)

On considère un arbre non-orienté, $G = (V, E)$.

(4.1) Quel est $|E|$, le nombre d'arcs de G ? (0.5pts)

☐ $|V|^2$ ☐ $\frac{|V|*(|V|-1)}{2}$ ☐ $|V| - 1$ ☐ $|V| + 1$

(4.2) Que vaut $\sum_{u \in V} d(u)$, la somme des degrés de G ? (0.5pts)

☐ $|V| * (|V| - 1)$ ☐ $|E|^2$ ☐ $2|E|$ ☐ $|V||E|$

On suppose que dans cet arbre 2 sommets sont de degré 4, 1 sommet est de degré 3, 2 sommets sont de degré 2. Tous les autres sommets ont degré 1.

(4.3) En utilisant les réponses aux deux questions précédentes, calculez x, le nombre de sommets de degré 1. Écrivez votre raisonnement. (2 pts)

Exercice 5 : Algorithmique de graphe (6 points)

Soit $G = (V, E)$ un graphe connexe non pondéré.

On définit le diamètre de G comme la longueur du plus long plus court chemin :

$$\varnothing(G) = \max_{u,v \in V} dist(u, v)$$

où $dist(u, v)$ est la longueur d'un plus court chemin de u à v.

Dans cet exercice, on va écrire un algorithme qui calcule le diamètre de G.

On considère que la fonction `BFS(G,u)` existe déjà. Elle calcule un parcours en largeur de G en partant d'un sommet u donné en paramètre et renvoie l'arbre des prédécesseurs sous forme d'un dictionnaire. Sa signature en python est la suivante :

```python
def BFS(G,u):
    """
       Calcule un parcours BFS de G en partant de u et renvoie un dictionnaire représentant l'arbre
           des prédécesseurs

       Entrées:
          G : un dictionnaire représentant un graphe sous forme de listes d'adjacence
          u : une clé du dictionnaire G représentant un sommet du graphe
       Renvoie:
          un dictionnaire dont les clés sont les sommets de G et les valeurs les prédécesseurs de
              chaque sommet dans le parcours en partant de u. La valeur de la clé u est None.
    """
```

Puisque G est connexe, tous les sommets de G seront dans l'arbre des prédécesseurs. On peut se servir de cela dans la suite.

> Les questions suivantes vous demandent d'écrire des fonctions pour arriver progressivement au calcul du diamètre. **Écrivez les en python mais, si vous n'y arrivez pas, donnez un pseudo-code qui décrit l'algorithme de la fonction, vous aurez une partie des points**.

(5.1) *Écrivez la fonction `distance(v,T)` qui prend en paramètre un somment v et l'arbre des prédécesseurs obtenu par un appel à `BFS(G,u)`, où u est un sommet de G, et renvoie $dist(u,v)$ = la longueur du plus court chemin de u à v obtenu en utilisant T (3 pts)*

.

```
def distance(v,T):
    """
        Renvoie dist(u,v) où u est le sommet ayant servi à calculer T par un BFS
        Entrées:
            v : un sommet d'un graphe G
            T : un dictionnaire obtenu par l'appel à BFS(G,u)
        Renvoie:
            un entier valant la longueur du plus court chemin de u à v calculé en utilisant T
    """
```

Maintenant qu'on a une fonction qui donne la longueur d'un plus court chemin entre deux sommets, on peut calculer *l'excentricité* d'un sommet u, c'est-à-dire la longueur du plus grand plus court chemin partant de u : $excentricite(u) = \max_{v \in V} dist(u, v)$.

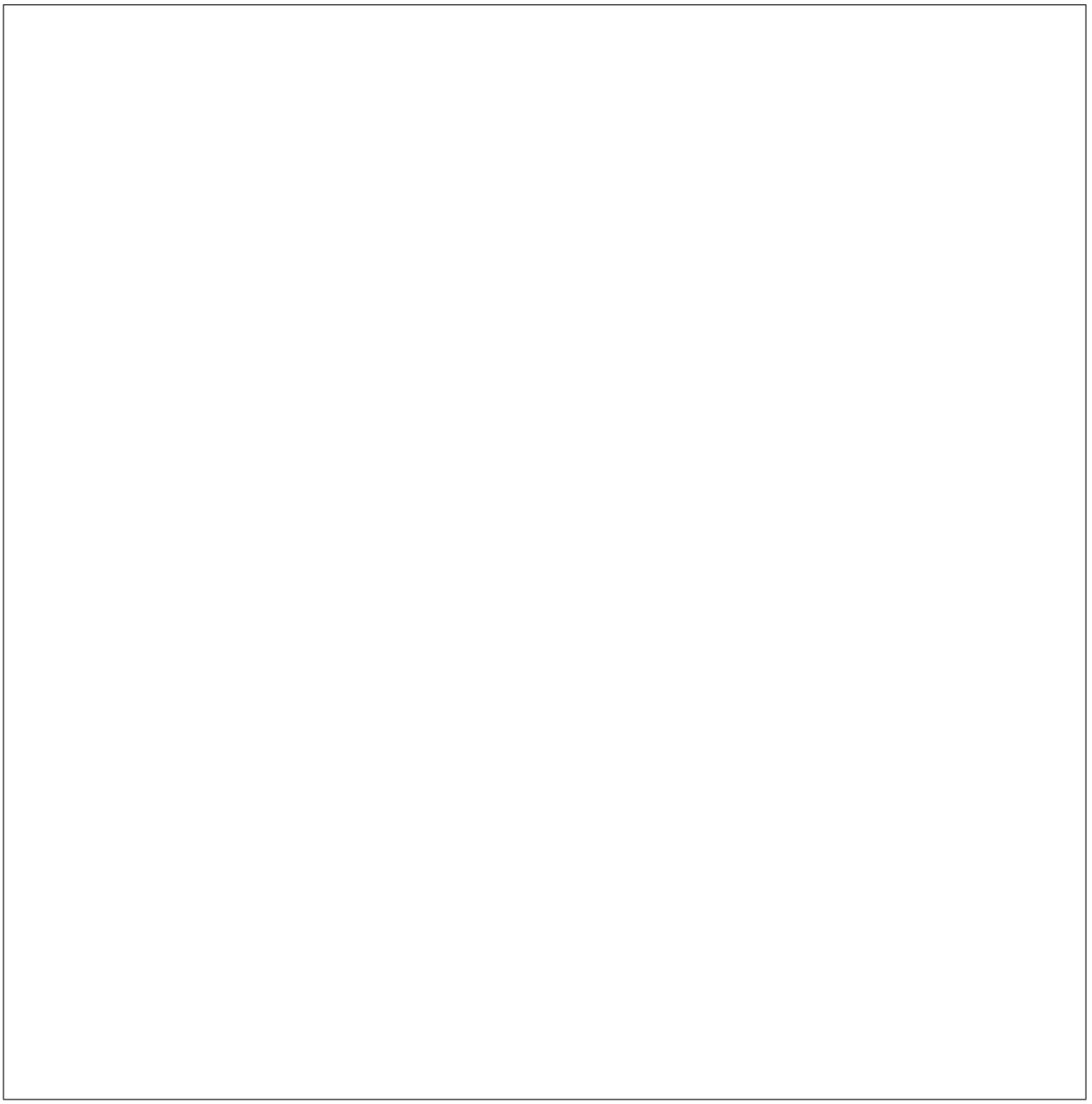
(5.2) Écrivez la fonction `excentricite(G,u)` qui prend en paramètre le graphe G et un sommet u et renvoie l'excentricité de u. Utilisez la fonction `distance` de la question précédente. (1.5 pt)

```
1  def excentricite(G,u):
2      """
3         Renvoie la longueur maximale d'un plus court chemin partant de u
4         Entrées:
5            G : un dictionnaire représentant un graphe sous forme de listes d'adjacence
6            u : une clé du dictionnaire G représentant un sommet du graphe
7         Renvoie:
8              un entier valant la longueur du plus long plus court chemin partant de u
9      """
```

Vous pouvez maintenant calculer le diamètre de G :
$\varnothing(G) = \max_{u,v \in V} dist(u,v) = \max_{u \in V} excentricite(u)$.

(5.3) Écrivez la fonction `diametre(G)` qui prend en paramètre le graphe G et renvoie son diamètre. Utilisez la fonction `excentricite` de la question précédente. (1.5 pt)

```
def diametre(G):
    """
       Renvoie le diametre d'un graphe G
       Entrées:
          G : un dictionnaire représentant un graphe sous forme de listes d'adjacence
       Renvoie:
            un entier valant le diametre de G
    """
```



(5.4) BONUS : le calcul de l'excentricité tel qu'il est décomposé pourrait être optimisé pour réduire la quantité de calculs à faire. Pourquoi ? Quels sont les calculs redondants ? (1 pt)

(5.5) BONUS : proposez une solution pour diminuer ces calculs redondants (1 pt pour l'idée, 3 pts pour une implémentation)