**Compte rendu n°2**

# Donner le pseudo-code de chaque algorithme approché.

Classe LargestCliqueFinder {

#graph; // Le graphe d'entrée

// Constructeur de la classe

constructor(graph) {

this.#graph = graph;

}

// Méthode privée pour vérifier si un sommet peut être ajouté à la clique actuelle

#canAddToClique(currentClique, vertex) {

Pour chaque sommet neighbor dans currentClique {

Si vertex n'est pas adjacent à neighbor {

Retourner faux; // Le sommet ne peut pas être ajouté à la clique

}

}

Retourner vrai; // Le sommet peut être ajouté à la clique

}

// Méthode privée pour trier les sommets par degré décroissant

#sortVerticesByDegree() {

Trier les sommets en fonction de leur degré (nombre de connexions) décroissant;

Retourner les sommets triés;

}

// Méthode publique pour trouver la plus grande clique

findLargestClique() {

Tri des sommets par degré décroissant;

Initialisation d'une clique vide maxClique;

Pour chaque sommet vertex dans les sommets triés {

Si vertex peut être ajouté à maxClique (appel de la fonction privée canAddToClique(maxClique, sommet)){

Ajouter vertex à maxClique;

}

} Retourner la plus grande clique (maxClique);

}

}

# Expliquer l'utilité des différents paramètres de l'algorithme.

Graph : Le paramètre d'entrée de la classe LargestCliqueFinder est un objet qui contient à la fois la liste des sommets et la liste d'adjacence du graphe. Il est utilisé pour représenter la structure du graphe sur lequel l'algorithme va opérer.

# Comment peut-on appliquer chaque algorithme approché pour résoudre votre problème d'optimisation?

Détails du graphe:

Sommets du graphe: 4, 3, 2, 1, 0

Liste d'adjacence du graphe:

0: 1, 2, 3

1: 0, 3, 2

2: 0, 1, 3

3: 0, 1, 2

4: 1, 2

Étape 1 : Tri des sommets par degré décroissant (nombre de connexions)

Itération sur les sommets triés 3, 2, 1, 0, 4

Étape 2 : Vérification si le sommet 3 peut être ajouté à la clique:

- Le sommet 3 est adjacent à tous les sommets de la clique:

- 3 a été ajouté à la clique: 3

Étape 2 : Vérification si le sommet 2 peut être ajouté à la clique: 3

- Le sommet 2 est adjacent à tous les sommets de la clique: 3

- 2 a été ajouté à la clique: 3,2

Étape 2 : Vérification si le sommet 1 peut être ajouté à la clique: 3,2

- Le sommet 1 est adjacent à tous les sommets de la clique: 3,2

- 1 a été ajouté à la clique: 3,2,1

Étape 2 : Vérification si le sommet 0 peut être ajouté à la clique: 3,2,1

- Le sommet 0 est adjacent à tous les sommets de la clique: 3,2,1

- 0 a été ajouté à la clique: 3,2,1,0

Étape 2 : Vérification si le sommet 4 peut être ajouté à la clique: 3,2,1,0

- Le sommet 4 n'est pas adjacent à 3, donc le sommet 4 n'est pas adjacent à tous les sommets de la clique: 3,2,1,0

- 4 n'a pas été ajouté à la clique: 3,2,1,0

Étape 3 : Affichage de la clique maximale trouvée

Clique maximale trouvée: 3, 2, 1, 0