***Résolution du problème algorithmique du postier chinois :***

Baptiste PERIAUT, Mathéo CORTEZ, Célestin CECILIEN, Alexis RAMOND

Groupe : B10

Mardi 26 mars

**Présentation et explication de la problématique :**

Le problème choisit est celui du Postier Chinois. Le code de l’algorithme doit parcourir et utiliser le graphe connexe non orienté pour trouver le chemin le plus court possible et revenir au point initial au sein de ce même graphe en passant au moins une fois par chaque arrête constituant le graphe. Pour ce faire, il nous faut créer un graphe quelconque et ensuite, à l’aide d’un programme algorithmique en langage python, d’améliorer ce graphe en modifiant et rajoutant des arêtes aux sommets pour le rendre parcourable.

**Technique n°=1 : Algorithme du cycle eulérien**

Explication de la technique :

Cette technique va prendre un graphe et le transformer en graphe eulérien (si nécessaire et si possible) en rajoutant des arêtes (il n’en supprime pas). Ensuite il va parcourir ce graphe et renvoyer le chemin le plus court qui passe par toutes les arêtes.

Points-clés : Il y a 3 fonctions principales dans cette technique.

La première, va vérifier si le graphe est eulérien (VerifEuler.degre), s’il ne l’est pas alors il renvoie les sommets qui empêchent le graphe d’être eulérien (VerifEuler.retournePB).

La seconde fonction, va transformer le graphe en graphe eulérien en rajoutant seulement des arêtes. Il n’en supprime pas (transfEuler.transfEuler). Cette fonction va d’abord rajouter des arêtes entre les sommets qui posent des problèmes. Si ce n’est plus possible, il prend des sommets aléatoirement.

Et la dernière, va renvoyer le chemin le plus court du graphe en passant par toutes les arêtes et va revenir à son point de départ (CycleEulerien.cycleEuler). Cette fonction va d’abord privilégier les boucles, pour ensuite essayer de prendre une arête qui ne retourne pas au sommet de départ, et va éviter les voisins qui peuvent retourner au sommet de départ.

Avantages : Cette technique, crée un chemin eulérien et hamiltonien, soit un chemin qui parcourt tous les sommets une seule fois et passe par toutes les arêtes. Cette version est rapide et optimisée.

Inconvénients : Cette technique ne traite pas les graphes complets avec des sommets pairs.

**Technique n°=2 : Algorithme de graphe complet**

Explication de la technique :

Cette technique va prendre un graphe et le transformer en graphe complet (si nécessaire) en rajoutant des arêtes. Ensuite il va parcourir ce graphe et renvoyer un chemin qui passe par toutes les arêtes.

Points-clés : Il y a 2 fonctions principales dans cette méthode

La première, va transformer le graphe en graphe complet en rajoutant des arêtes (AlgoComplet.grapheComplet).

Et la seconde fonction, va renvoyer un chemin en passant par toutes les arêtes et qui revient à son point de départ (AlgoComplet.chemin). Cette fonction va d’abord privilégier les sommets qui ont encore des arêtes à parcourir (si le sommet sur laquelle on est ne possède plus de voisin, on se dirige vers un sommet qui en a encore), quand toutes les arêtes ont été parcourues, on retourne au sommet de départ.

Avantages : Cette technique, permet d’obtenir toujours un chemin qui passe au moins une fois par chaque arête.

Inconvénients : Si le graphe complet est de plus en plus complexe et grand, la résolution prendra de plus en plus de temps (complexité algorithmique proportionnellement strictement croissante). De plus, elle ne respecte la consigne qui dit de créer le moins d’arête possible.

**Différences des deux ensembles algorithmiques :**

L’algorithme cycle eulérien permet de créer efficacement le chemin de n’importe quel graphe s’il n’est pas complet ou si les sommets ne sont pas impairs. Quant à l’algorithme de graphe complet, il s’occupe uniquement des graphes complets, qu’importe le nombre de sommet. La seule différence notable est que l’algorithme de cycle Eulérien est plus optimisé et plus rapide que l’algorithme de graphe complet. Effectivement, si le graphe complet possède une centaine de sommet, l’algorithme de graphe complet deviendra plus lent à trouver le chemin adéquat. Ceci est causé par le fait que l’algorithme de graphe complet cherche un chemin, pas forcément le plus court, qui passe par toutes les arrêtes. Tandis que l’algorithme de cycle eulérien va renvoyer le chemin le plus court tout en passant par toutes les arrêtes une seule fois (un chemin hamiltonien).

Lien du Canva : [Diaporama](https://www.canva.com/design/DAF9VffICIw/6O85hC5ID9hyxmkEvdSyVg/view?utm_content=DAF9VffICIw&utm_campaign=designshare&utm_medium=link&utm_source=editor)