Compte rendu Graphes et Image (Dijkstra et modélisation)

Ramzy CHIBANI

December 2024

1 Objectif du Projet :

Le projet consiste à représenter une image sous forme de graphe, où chaque pixel devient un sommet, et les relations de voisinage entre pixels représentent des arêtes pondérées par les différences d'intensité entre ces pixels. L'objectif est de construire une interface permettant de désigner deux pixels comme départ et arrivée, de calculer le plus court chemin entre ces deux sommets grâce à l'algorithme de Dijkstra, et de visualiser ce chemin sur l'image de notre choix.

2 Étapes de Réalisation :

2.1 Modélisation du Graphe :

- Chaque pixel de l'image est considéré comme un sommet du graphe.
- Les voisins directs des pixels (haut, bas, gauche, droite) sont reliés par des arêtes.
- Le poids des arêtes est déterminé par la différence absolue des intensités des pixels adjacents.
- Pour pouvoir faire ça, on transforme notre image en noir et blanc, car moins de calculs à faire qu'en RGB.

2.2 Algorithme de Dijkstra :

- J'ai pu optimiser le code qui prennait au départ un énorme temps graçe à l'utilisation d'une file de priorité **(heap)**.
- La file de priorité gère efficacement les distances minimales grâce au module **heapq**.
- Dès qu'on atteint le sommet final, le calcul s'arrête pour économiser des ressources.

2.3 Interface Graphique :

- Créée avec **Tkinter** pour permettre à l'utilisateur de charger une image, définir les sommets de départ et d'arrivée, et calculer le chemin.
- Le chemin calculé est tracé sur une copie de l'image d'entrée.

2.4 Optimisation :

- L'utilisation d'un heap (file de priorité) optimise la complexité de Dijkstra en réduisant les insertions et extractions de la file à O(logV), où V est le nombre de sommets.
- Compléxité totale : O((V+E)logV), où E est le nombre d'aretes.

3 Justification des Choix :

- La transformation en niveaux de gris simplifie les calculs en réduisant chaque pixel à une seule intensité, facilitant le traitement des poids pour l'algorithme de Dijkstra.
- Représentation des Pixels en Sommets : Simple et directe pour une grille 2D.
- Heap pour Dijkstra : Permet une gestion rapide des sommets non visités.

- Tkinter : Bibliothèque légère pour développer des interfaces en Python et faciel à utiliser.
- La bibliothèque cv2 est utilisée pour sa puissance et sa simplicité dans le traitement d'images, notamment pour lire, convertir en niveaux de gris, manipuler les pixels, et visualiser les résultats efficacement, j'ai voulu changé de Pillow pour aprendre plus de cette bibliothéque.

4 Résultats :

L'application permet :

- De charger des images en niveaux de gris.
- De visualiser le chemin le plus court calculé entre deux pixels.
- Une interface intuitive avec des résultats rapides pour des images de taille raisonnable.

5 Difficultés Rencontrées :

- Gestion des indices lors de la conversion des coordonnées d'image en sommets du graphe.
- Adapter mon code apres l'utilisation d'un **heapq**
- Utilisation de tkinter pour pouvoir afficher l'image et pouvoir modifier les points de départs et d'arriver sans fermer tout le GUI à chaque fois.

6 Sources Utilisées :

- **Le TP 07 et Wikipedia** : Utilisé pour comprendre les concepts fondamentaux des graphes et de l'algorithme de Dijkstra.
- **GeeksforGeeks** : Référence pour la documentation et les exemples pratiques de la bibliothèque Tkinter pour créer une interface graphique.
- **Stack Overflow** : Consulté pour résoudre des problèmes spécifiques liés à la manipulation des structures de graphes et à l'optimisation de l'algorithme.
- **OpenAI (GPT)** : Aide fournie pour la compréhension et l'implémentation de la file de priorité avec le module **heapq** pour optimiser l'algorithme de Dijkstra.
- **Un tuto youtube sur OpenCV** : Utilisée pour les manipulations d'images, notamment la conversion en niveaux de gris et la visualisation avec des lignes tracées.

7 Conclusion :

Ce projet a permis d'approfondir mes connaissances sur les graphes, les algorithmes de cheminement (Dijkstra) et le développement d'interfaces graphiques en Python. L'utilisation de la file de priorité a notablement amélioré les performances de l'algorithme, rendant le programme efficace pour des images de taille modérée.