RAPPORT RENDU 2 - PSI

Tom DEHE--COHEN, Quentin DE WOLF, Nils DEMOUGEOT | TD G

Lien github : https://github.com/quentin-dw/livin_paris

1 - Comparaison des 3 algorithmes de plus court chemin :

Pour les 3 algorithmes suivants nous allons utiliser ces notations :
V = nombre de noeuds / stations
E = nombre d'arêtes

A) Algorithme de Dijkstra : O((V + E)log(V))

L'algorithme de Dijkstra est de loin le plus performant et adapté à notre besoin. En effet c'est celui qui possède une complexité la plus faible.

Suite à des tests de benchmark chronométrés à l'aide de la classe Stopwatch sur ces 3 algorithmes, nous avons pu observer l'utilité d'une complexité basse : il trouvait la plupart du temps le chemin le plus court en 2 à 3 ms.

B) Algorithme de Bellman-Ford : O(VE)

Le 2e meilleur en termes de complexité, c'est cependant dans les tests qu'on remarque qu'il reste bien plus lent que l'algorithme de Dijkstra avec un temps d'exécution moyen entre 80 et 100 ms, ce qui reste raisonnable malgré tout dans notre cas.

Cependant, notre utilisation n'inclut pas des calculs avec des poids négatifs, donc cette fonctionnalité inutilisée fait perdre en efficacité et en intérêt à cet algorithme, qui prend alors un rôle comparable à celui de DIjkstra, mais avec un temps d'exécution 25 à 50 fois plus long.

C) Algorithme de Floyd-Warshall : O(V^3)

Enfin, nous avons également implémenté l'algorithme de Floyd-Warshall. Celui-ci fonctionne différemment des 2 premiers car au lieu de renvoyer le plus court chemin entre 2 sommets pré-définis, il renvoie l'intégralité des plus courts chemins pour atteindre chaque sommet depuis n'importe quel sommet.

Cette façon de fonctionner peut sembler très attrayante, cependant lors des tests, l'algorithme mettait en moyenne 7 secondes à s'exécuter, soit 2000 à 3500 fois plus long que l'algorithme de Dijkstra.

Malgré tout, étant donné que sur notre graphe, l'algorithme renvoie 107584 chemins, on réalise alors qu'il est finalement plus rapide que l'algorithme de Dijkstra lorsqu'on considère le temps par chemin qui passe alors à 0.065 ms.

Cependant, même si l'option de l'algorithme de Floyd-Warshall peut sembler intéressante sur le papier, il faut aussi considérer le temps que mettra le programme à aller chercher le plus court chemin entre 2 stations dans le Dictionary renvoyé par l'algorithme.
De plus, le temps d'exécution de 7 secondes se fait vraiment ressentir par l'utilisateur en comparaison à un temps de 3 ms qui est totalement négligeable de ce point de vue, même s'il à lieu plusieurs fois.

Après réflexion, nous avons donc décidé d'utiliser l'algorithme de Dijkstra car nous avons besoin d'un algorithme rapide qui donne le plus court chemin entre 2 stations prédéfinies.

2 - SQL

```
-- creation des tables

CREATE TABLE Compte(
  id_compte INT AUTO_INCREMENT,
  prenom VARCHAR(50),
  nom VARCHAR(50),
  telephone VARCHAR(50),
  rue VARCHAR(50),
  numero INT,
  code_postal INT,
  ville VARCHAR(50),
  metro_le_plus_proche VARCHAR(50),
  email VARCHAR(50),
  mot_de_passe VARCHAR(50),
  PRIMARY KEY(id_compte)
);

CREATE TABLE Cuisinier(
  id_cuisinier INT AUTO_INCREMENT,
  id_compte INT NOT NULL,
  PRIMARY KEY(id_cuisinier),
  UNIQUE(id_compte),
  FOREIGN KEY(id_compte) REFERENCES Compte(id_compte)
);

CREATE TABLE Ingrédient(
  id_ingredient INT AUTO_INCREMENT,
  nom_ingredient VARCHAR(50) NOT NULL,
  type VARCHAR(50),
  PRIMARY KEY(id_ingredient)
);

CREATE TABLE Client(
```

```
  id_client INT AUTO_INCREMENT,
  entreprise BOOLEAN,
  nom_entreprise VARCHAR(50),
  id_compte INT NOT NULL,
  PRIMARY KEY(id_client),
  UNIQUE(id_compte),
  FOREIGN KEY(id_compte) REFERENCES Compte(id_compte)
);

CREATE TABLE Recette(
  id_recette INT AUTO_INCREMENT,
  nom_recette VARCHAR(50),
  declinaison BOOLEAN,
  id_recette_declinee VARCHAR(50),
  PRIMARY KEY(id_recette)
);

CREATE TABLE Plat(
  id_plat INT AUTO_INCREMENT,
  nom_plat VARCHAR(50),
  type VARCHAR(50),
  date_fabrication DATE,
  date_peremption DATE,
  prix DECIMAL(15,2),
  nationnalite_de_cuisine VARCHAR(50),
  regime_alimentaire VARCHAR(50),
  photo VARCHAR(50),
  id_cuisinier INT NOT NULL,
  id_recette INT NOT NULL,
  PRIMARY KEY(id_plat),
  FOREIGN KEY(id_cuisinier) REFERENCES Cuisinier(id_cuisinier),
  FOREIGN KEY(id_recette) REFERENCES Recette(id_recette)
);

CREATE TABLE Commande(
  id_commande INT AUTO_INCREMENT,
  avis_client VARCHAR(50),
  note_client INT,
  cout_total DECIMAL(15,2),
  id_client INT NOT NULL,
  PRIMARY KEY(id_commande),
  FOREIGN KEY(id_client) REFERENCES Client(id_client)
);

CREATE TABLE Ligne_de_commande(
  id_ligne_de_commande INT AUTO_INCREMENT,
  lieu_livraison VARCHAR(50),
  quantite VARCHAR(50),
```

```
  date_livraison VARCHAR(50),
  cout DECIMAL(15,2),
  id_commande INT NOT NULL,
  id_plat INT NOT NULL,
  PRIMARY KEY(id_ligne_de_commande),
  FOREIGN KEY(id_commande) REFERENCES Commande(id_commande),
  FOREIGN KEY(id_plat) REFERENCES Plat(id_plat)
);

CREATE TABLE Livre(
  id_cuisinier INT,
  id_client INT,
  PRIMARY KEY(id_cuisinier, id_client),
  FOREIGN KEY(id_cuisinier) REFERENCES Cuisinier(id_cuisinier),
  FOREIGN KEY(id_client) REFERENCES Client(id_client)
);

CREATE TABLE compose(
  id_ingredient INT,
  id_recette INT,
  quantite VARCHAR(50),
  PRIMARY KEY(id_ingredient, id_recette),
  FOREIGN KEY(id_ingredient) REFERENCES Ingrédient(id_ingredient),
  FOREIGN KEY(id_recette) REFERENCES Recette(id_recette)
);




-- Peuplement des tables

INSERT INTO Compte (id_compte, telephone, nom, prenom, rue, numero, code_postal,
ville, metro_le_plus_proche, email, mot_de_passe) VALUES
(1,'0123456789', 'Dupont', 'Jean','Rue de la Paix', 1, 75001, 'Paris', 'Chatelet',
'cuisinier1@example.com', 'mdp123'),
(2,'0987654321', 'Martin', 'Pierre','Avenue Victor Hugo', 10, 75016, 'Paris', 'Champs-Elysees
- Clemenceau', 'cuisinier2@example.com', 'azerty'),
(3,'0147852369', 'Bernard', 'Luc','Boulevard Saint-Germain', 5, 75006, 'Paris', 'Porte de
Vanves', 'cuisinier3@example.com', 'qwerty'),
(4,'0234567891', 'Durand', 'Marie','Rue de Rivoli', 20, 75004, 'Paris', 'Les Halles',
'cuisinier4@example.com', 'mdp'),
(5,'0345678912', 'Lefevre', 'Sophie','Rue de la République', 15, 69001, 'Lyon', 'Olympiades',
'cuisinier5@example.com', '@pple'),
(6,'0456789123', 'Legrand', 'Alice','Rue des Entrepreneurs', 8, 33000, 'Bordeaux', 'Jaures',
'client1@example.com', 'test'),
(7,'0567891234', 'Petit', 'Bob','Rue de la Gare', 12, 59000, 'Lille', 'Couronnes',
'client2@example.com', 'coucou'),
```

(8,'0678912345', 'Roux', 'Claire','Avenue de la Liberté', 22, 31000, 'Toulouse', 'Duroc', 'client3@example.com', 'dssssd5'),
(9,'0789123456', 'Moreau', 'David','Rue Victor Hugo', 30, 13000, 'Marseille', 'Saint-Lazare', 'client4@example.com', 'let2565'),
(10,'0891234567', 'Girard', 'Emma','Rue Nationale', 3, 35000, 'Rennes', 'Pyramides', 'client5@example.com', 'mdp987');

INSERT INTO Cuisinier (id_compte) VALUES
(1),
(2),
(3),
(4),
(5);

INSERT INTO Ingrédient (nom_ingredient, type) VALUES
('Tomate', 'Légume'),
('Basilic', 'Herbe'),
('Pomme de terre', 'Légume'),
('Carotte', 'Légume'),
('Oignon', 'Légume'),
('Ail', 'Assaisonnement'),
('Poulet', 'Viande'),
('Poivron', 'Légume'),
('Champignon', 'Légume'),
('Riz', 'Céréale');

INSERT INTO Client (entreprise, nom_entreprise, id_compte) VALUES
(FALSE, NULL, 6),
(FALSE, NULL, 7),
(FALSE, NULL, 8),
(FALSE, NULL, 9),
(TRUE, 'Girard SA', 10);

INSERT INTO Recette (nom_recette, declinaison, id_recette_declinee) VALUES
('Salade de tomates', FALSE, NULL),
('Poulet rôti', FALSE, NULL),
('Ratatouille', TRUE, '1'),
('Tarte aux champignons', FALSE, NULL),
('Riz cantonais', TRUE, '2');

INSERT INTO Plat (nom_plat, type, date_fabrication, date_peremption, prix, nationnalite_de_cuisine, regime_alimentaire, photo, id_cuisinier, id_recette) VALUES
('Macaronnis au fromage', 'Entrée', '2025-02-20', '2025-02-28', 12.50, 'Française', 'Végétarien', 'plat1.jpg', 1, 1),
('Pizza 4 fromages','Plat principal', '2025-02-21', '2025-02-27', 18.00, 'Italienne', 'Standard', 'plat2.jpg', 2, 2),
('Cassoulet','Dessert', '2025-02-22', '2025-03-01', 9.00, 'Française', 'Végétarien', 'plat3.jpg', 3, 3),

('Tapas variés','Entrée', '2025-02-23', '2025-03-02', 10.00, 'Espagnole', 'Standard', 'plat4.jpg', 4, 4),
('Escargots','Plat principal', '2025-02-24', '2025-03-03', 20.00, 'Française', 'Vegan', 'plat5.jpg', 5, 5),
('Epinards à la creme','Dessert', '2025-02-25', '2025-03-04', 8.50, 'Française', 'Standard', 'plat6.jpg', 1, 2),
('Salade césar','Entrée', '2025-02-26', '2025-03-05', 11.00, 'Italienne', 'Végétarien', 'plat7.jpg', 2, 3),
('Burritos piquant','Plat principal', '2025-02-27', '2025-03-06', 22.00, 'Mexicaine', 'Standard', 'plat8.jpg', 3, 4),
('Burger Végé','Dessert', '2025-02-28', '2025-03-07', 7.00, 'Américaine', 'Végétarien', 'plat9.jpg', 4, 5),
('Champignons de paris cuisinés','Entrée', '2025-03-01', '2025-03-08', 13.50, 'Française', 'Standard', 'plat10.jpg', 5, 1),
('Spaghetti bolognes','Plat principal', '2025-03-04', '2025-06-30', 8.20, 'Italienne', 'Standard', 'plat11.jpg', 5, 1);

INSERT INTO Commande (avis_client, note_client, cout_total, id_client) VALUES
('Très bon', 5, 45.50, 1),
('Bon service', 4, 30.00, 2),
('Moyen', 3, 25.00, 3),
('Excellent', 5, 50.00, 4),
('Peut mieux faire', 2, 20.00, 5),
('Très bon', 4, 35.00, 1),
('Bon', 4, 40.00, 2);

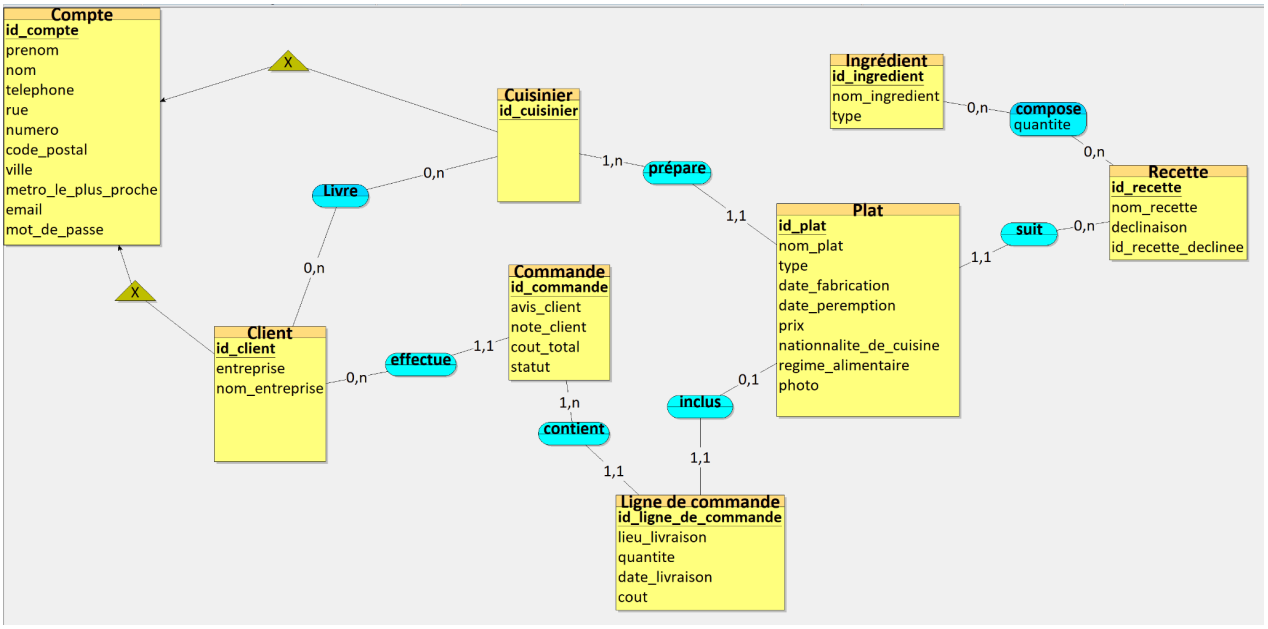
INSERT INTO Ligne_de_commande (lieu_livraison, quantite, date_livraison, cout, id_commande, id_plat) VALUES
('Concorde', '2', '2025-03-10', 25.00, 1, 1),
('Montparnasse Bienvenue', '1', '2025-03-11', 18.00, 2, 2),
('Vavin', '3', '2025-03-12', 27.00, 3, 3),
('Republique', '2', '2025-03-13', 20.00, 4, 4),
('Pigaille', '1', '2025-03-14', 20.00, 5, 5),
('Duroc', '2', '2025-03-15', 17.00, 6, 6),
('Varenne', '1', '2025-03-16', 11.00, 7, 7);

INSERT INTO Livre (id_cuisinier, id_client) VALUES
(1, 1),
(1, 2),
(2, 3),
(2, 4),
(3, 5),
(4, 1),
(5, 3),
(5, 4);

INSERT INTO compose (id_ingredient, id_recette, quantite) VALUES
(1, 1, '200g'),

(2, 1, '10 feuilles'),
(3, 2, '300g'),
(4, 3, '150g'),
(5, 3, '1 oignon'),
(6, 2, '2 gousses'),
(7, 2, '500g'),
(8, 4, '100g'),
(9, 5, '50g'),
(10, 5, '250g');

3) Diagramme Entité-Association



4) IA générative pour la visualisation du graphe

a- Compréhension des solutions proposées (System.Drawing et Skiasharp)

Pour un projet en C# je cherche à visualiser un graphe. J'ai déjà le code pour lire ce graphe sous la forme d'une liste de liens et d'une liste de nœuds (Noeud et Lien sont 2 classes que j'ai créées). Je souhaite utiliser Skia Sharp ou System.Drawing, peux tu me dire lequel est les avantages et inconvénients de ces 2 solutions.

Réponse :
SkiaSharp : positif (performance, modernité) - négatif (difficulté, package NuGet)
System.Drawing : positif (simplicité, compatibilité WinForms) - négatif (performance limitées)

Choix de la solution System.Drawing principalement du fait de la compatibilité avec l'environnement windows déjà bien éprouvée.

b- Compréhension du fonctionnement de System.Drawing

Dans un premier temps, je souhaite opter pour System.Drawing qui me semble une solution plus simple à mettre en œuvre. Peux-tu m'expliquer son fonctionnement général ?

Réponse :
Explication du principe de Bitmap vs WinForms

Choix de la solution WinForms car elle permet de l'interactivité (bouton…)

c- Demandes de codes pour initialiser le WinForms

```
internal static class Program
  {
    [DllImport("kernel32.dll")]
    static extern bool AllocConsole();  // Permet d'ouvrir une console

    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);

      AllocConsole(); // Ouvre la console
      Console.WriteLine("Console attachée !");

      Application.Run(new Form1()); // Lance la fenêtre
    }
  }


public partial class GrapheForm : Form
  {
    private Graphe<int> graphe;

    public GrapheForm(string noeuds, string arcs)
    {
      InitializeComponent();
      graphe = new Graphe<int>(noeuds, arcs);
    }
  }
```

d- Aide à l'amélioration du WinForms à travers des questions guidés

- Explication du fonctionnement du HashSet<>
- Aide à la création de boutons