Projet ALMF51

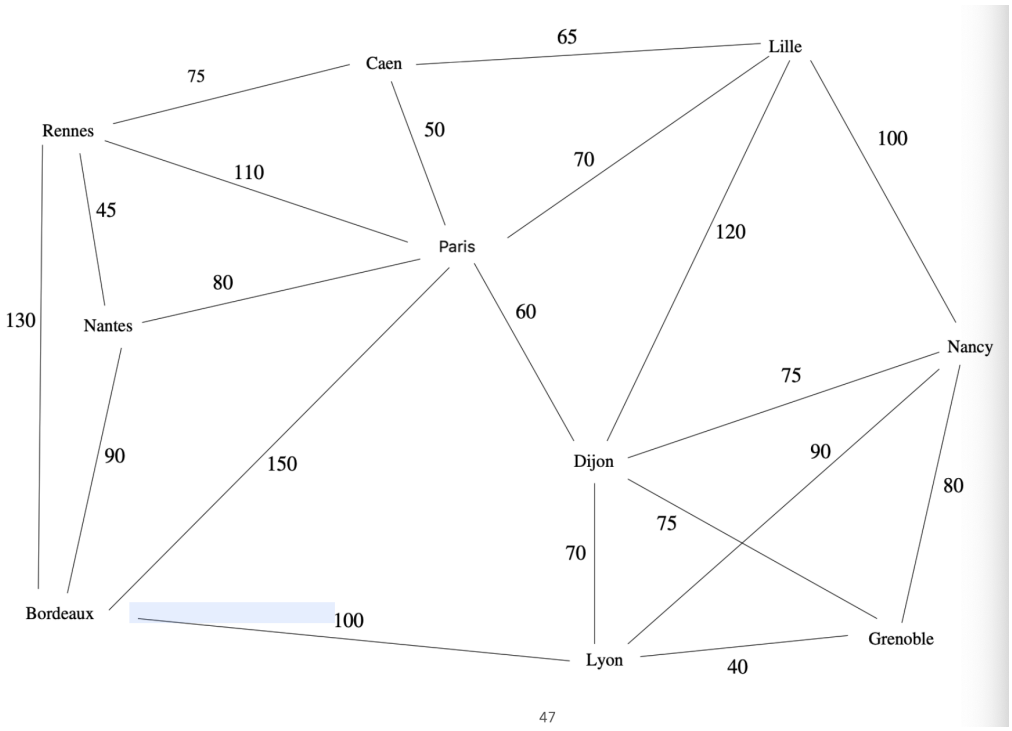
Algorithmes de graphes — Parcours, optimisation et applications

Deadline : 08/02/2026 à minuit
Dépôt : via Moodle

Contexte

Ce projet a pour objectif de vous permettre d'appliquer de manière concrète les algorithmes de graphes étudiés en cours. Vous travaillerez sur un graphe pondéré représentant un réseau routier : les **villes** sont les sommets et les **routes** entre deux villes sont les arêtes, pondérées par une distance ou un coût.

La carte suivante est fournie comme base possible :



Important : la carte est **indicative**. Vous pouvez tout à fait :

▷ utiliser la carte fournie,

▷ ou utiliser votre **propre dataset** (JSON, CSV, ou construit manuellement),

▷ ou choisir un graphe thématique (réseau informatique, réseau de transport, etc.).

L'essentiel est que vos données soient cohérentes, correctement présentées, et utilisées dans toutes les parties du projet.

Objectifs du projet

Ce projet vous guidera progressivement à travers l'étude de plusieurs classes d'algorithmes fondamentaux sur les graphes :

◇ parcours (BFS, DFS),

◇ arbres couvrants minimum (Prim, Kruskal),

◇ plus courts chemins (Dijkstra, Bellman–Ford, Floyd–Warshall),

◇ ordonnancement de projets (méthode PERT).

Chaque partie vous permettra :

⋆ de comprendre le rôle de l'algorithme,

⋆ d'implémenter une version correcte et propre,

⋆ de l'expérimenter sur un cas réel issu de votre dataset,

⋆ d'interpréter les résultats.

Travail demandé

I — Parcours de graphe : BFS et DFS

Objectif pédagogique : comprendre les deux stratégies d'exploration les plus utilisées pour visiter un graphe.

Travail à réaliser :

- Implémenter BFS et DFS dans votre langage de choix.
- Lancer les deux parcours à partir d'une ville de référence.
- Comparer l'ordre de visite obtenu.

Exemple demandé : ordre de visite à partir de Rennes.

Dans le rapport :

— explication des différences conceptuelles,

— ordre de visite obtenu,

— commentaires sur les éventuelles différences.

II — Arbre couvrant de poids minimum : Prim et Kruskal

Objectif pédagogique : comprendre comment construire un réseau couvrant efficace au coût minimal.

Travail à réaliser :

- Implémenter les algorithmes de Prim et de Kruskal.
- Construire un arbre couvrant minimal pour votre graphe.
- Afficher :
  - — la liste des routes sélectionnées,
  - — le coût total de l'arbre.

Dans le rapport :

— capture ou schéma du réseau minimal,

— comparaison éventuelle entre Prim et Kruskal.

III — Plus court chemin : algorithme de Dijkstra

Objectif pédagogique : calculer un trajet optimal dans un graphe à poids positifs.
Travail à réaliser :

- Implémenter Dijkstra avec file de priorité.
- Permettre à l'utilisateur de choisir deux villes.
- Afficher le chemin optimal et sa distance.

Exemple suggéré : Bordeaux → Lille.
Attendu : chemin, distance, visualisation si possible.

IV — Plus courts chemins avancés : Bellman–Ford et Floyd–Warshall

Objectif pédagogique : traiter des graphes généraux, y compris avec poids négatifs.

A — Bellman–Ford

Travail à réaliser :

- Ajouter un ou deux poids négatifs (cas contrôlé).
- Implémenter Bellman–Ford.
- Vérifier l'absence ou la présence d'un cycle négatif.

Attendu : distances minimales et détection des cycles négatifs.

B — Floyd–Warshall

Travail à réaliser :

- Implémenter l'algorithme.
- Calculer toutes les distances entre toutes les paires de villes.
- Afficher la matrice résultante.

Attendu : une analyse simple (ex. ville la plus "centrale").

V — Ordonnancement d'un projet : méthode PERT

Objectif pédagogique : modéliser un ensemble de tâches dépendantes et identifier le chemin critique d'un projet.
Travail à réaliser :

- Proposer un mini-projet (ex. construction d'une route, installation de bornes, planning d'un évènement).
- Définir chaque tâche :
  - durée,
  - dépendances directes.
- Construire le graphe PERT.
- Calculer :
  - dates au plus tôt,
  - dates au plus tard,

  - marges,
  - chemin critique.
- Fournir un diagramme PERT et, si possible, un diagramme de Gantt.

Attendu : explication claire du chemin critique (tâches sans marge).

Bonus — Interface graphique

Ce bonus vise à donner une dimension interactive à votre projet.
Vous pouvez proposer :

- un affichage dynamique du graphe,
- un menu permettant de choisir l'algorithme,
- une visualisation en couleur des résultats (arbre couvrant, chemin optimal, etc.).

Technologies possibles :

- **Desktop** : Python (Tkinter, PyQt5), NetworkX, Matplotlib,
- **Web** : Flask/Django + D3.js, Vis.js.

Livrables attendus

1. **Code source complet** : propre, commenté, structuré.
2. **Rapport clair et pédagogique** comprenant :
   - explication de chaque algorithme,
   - extraits d'implémentation,
   - résultats expérimentaux illustrés,
   - analyse critique.
3. **Bonus (optionnel)** : interface graphique fonctionnelle.

Notation et critères d'évaluation

- **Qualité de l'implémentation** (40%)
- **Qualité du rapport écrit** (30%)
- **Pertinence des résultats et démonstrations** (20%)
- **Bonus — Interface graphique** (10%)

Remarque : des points supplémentaires peuvent récompenser la créativité, la clarté méthodologique ou des fonctionnalités innovantes.