université virtuelle Burkina Faso

Master Fouilles de données & Intelligence Artificielle

Projet du cours de ***Complexité Algorithmique, Théorie des Graphes***

Sujet 2 : Plus court chemin

Analyse du Projet de Construction de M. Clément B.

Réalisée :
BASSOLE Yipéné Harold Ezekiel
BAZIE Dureel Donaldson
COULIBALY Cheick Ahmed
OUATTARA Arnauld
KOALA Valentin

Enseignant : Dr. Azise Oumar Diallo, Ph.D

June 22, 2025

1 Modélisation du projet sous forme de graphe

Nous modélisons le projet sous forme d'un graphe, noté :

$$G = (V, E)$$

— V est l'ensemble des sommets représentant les tâches du projet :

$$V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

D'où l'ordre du graphe : $O(G) = |V| = 11$

— E est l'ensemble des arcs orientés représentant les contraintes de précédence entre les tâches. Ces arcs sont déduits du dictionnaire des prédécesseurs :

$$E = \{(0,2), (0,4), (1,2), (1,4), (2,3), (3,5), (3,6), (3,7), (4,5), (4,6), (4,7), (6,8), (8,9), (5,10), (9,10)\}$$

Chaque sommet (nœud) est associé à un poids représentant la durée de la tâche correspondante.

Les arcs ne sont pas pondérés, car ils ne traduisent que des relations de dépendance (ordre d'exécution), et non des coûts ou des durées.

Ce graphe est un **graphe orienté acyclique** (*DAG*, Directed Acyclic Graph), car :

— orienté : les arcs vont dans un seul sens (d'une tâche vers sa dépendante),

— acyclique : aucune tâche ne dépend directement ou indirectement d'elle-même.

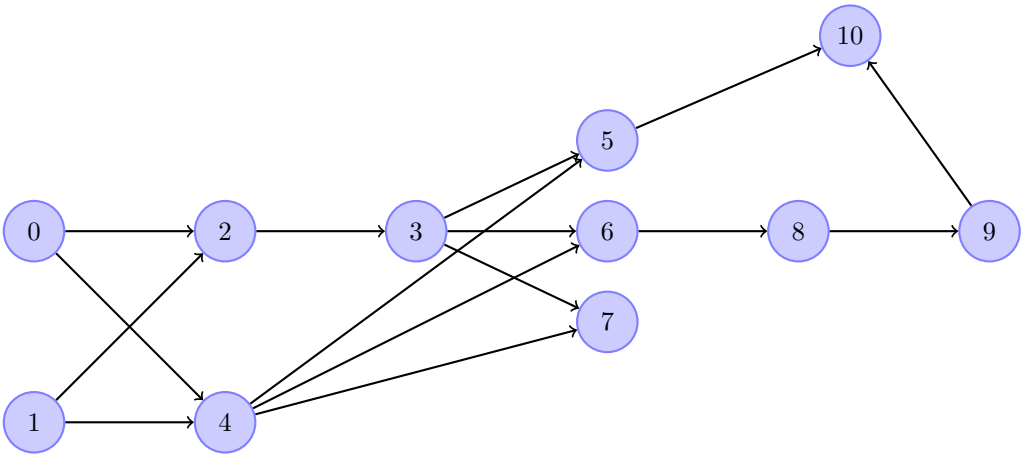

FIGURE 1 – Graphe orienté des tâches du projet

2 Chemin critique et durée minimale du projet

La durée minimale du projet est égale à la durée du chemin critique, c'est-à-dire le plus long chemin (en termes de durée) entre une tâche de départ (sans prédécesseur) et la tâche finale (sans successeur).

Pour déterminer cette durée, on cherche le chemin le plus long dans le graphe orienté et acyclique (DAG), du nœud de départ jusqu'au nœud final, en tenant compte des durées des tâches.

Voici un résumé des chemins et des durées cumulées à partir des tâches initiales :

Chemin	Somme des durées
$\mathbf{1 \to 2 \to 3 \to 5 \to 10}$	$7+3+1+2+1=14$
$\mathbf{1 \to 4 \to 5 \to 10}$	$7+8+2+1=18$
$\mathbf{0 \to 4 \to 5 \to 10}$	$2+8+2+1=13$
$\mathbf{1 \to 4 \to 6 \to 8 \to 9 \to 10}$	$7+8+1+3+2+1=22$ (critique)
$\mathbf{0 \to 4 \to 6 \to 8 \to 9 \to 10}$	$2+8+1+3+2+1=17$
$\mathbf{1 \to 2 \to 3 \to 6 \to 8 \to 9 \to 10}$	$7+3+1+1+3+2+1=18$
$\mathbf{0 \to 2 \to 3 \to 6 \to 8 \to 9 \to 10}$	$2+3+1+1+3+2+1=13$

La durée minimale du projet est donc de **22 semaines**, correspondant au **chemin critique** suivant :

$$\boxed{1 \to 4 \to 6 \to 8 \to 9 \to 10}$$

Ce projet est un cas classique de la méthode du chemin critique (*Critical Path Method – CPM*).

Le chemin critique est la suite de tâches qui déterminent la durée minimale du projet. Ce sont les tâches critiques : si l'une d'elles prend du retard, tout le projet est retardé.

3 Principe de Bellman adapté pour résoudre le problème

Principe de Bellman

Le principe de Bellman, dans le cadre du problème du plus court chemin, est une règle fondamentale qui sous-tend la programmation dynamique et l'algorithme de Bellman-Ford. Selon ce principe, dans un chemin optimal entre deux points, toute portion de ce chemin est elle aussi optimale entre les points qu'elle relie. Ce qui permet de décomposer le problème du plus court chemin en problèmes plus petits, plus faciles à résoudre, puis de les combiner pour trouver la solution globale.

L'algorithme de Bellman permet de rechercher les plus courts chemins entre deux nœuds d'un graphe orienté acyclique et dont les arêtes sont pondérés. Il est classiquement appliqué pour des problèmes d'optimisation.

Cet algorithme repose sur le principe de relaxation, que l'on peut expliquer comme suit. Soient :

- $p[]$ le tableau des poids de chaque arête ;
- $d[]$ le tableau de distance de chaque sommet à S_0, initialisé avec $d[S_0] = 0$ et $d[S_i] = \infty$ pour tout i différent de 0 ;
- $pred[]$ le tableau des prédécesseurs de chaque sommet sur le plus long chemin, initialisé à 0 pour tous les sommets.

```
Relaxation(u, v){
    Si d[u] + p[u,v] > d[v] Alors
        d[v] = d[u] + p[u,v]
        pred[v] = u
    FinSi
}
```

L'idée de l'algorithme est la suivante :

- étant donné un sommet u, la procédure relaxation u est appelée pour tous les sommets v adjacents à u ;
- le traitement précédent est répété sur chaque sommet ordonné suivant un tri topologique (c'est-à-dire qu'un sommet sera toujours traité avant ses successeurs) ;
- une fois tous les nœuds traités, on peut reconstruire le(s) plus long(s) chemin(s) en suivant les prédécesseurs depuis le sommet final dans le tableau $pred[]$.

Notons que si les sommets ne sont pas ordonnés selon un tri topologique, le traitement devra être répété jusqu'à ce que la relaxation ne fasse plus effet.

Algorithme de Bellman adapté :

L'algo suivant est basé sur le graphe précédent du projet.

```

#Initialisation:
    d[ ]=dates au plus tot
    pred[]=predecesseurs

L :liste des taches triees topologiquement
# Calcul des dates au plus tot par  relaxation
Pour u dans L
    Pour chaque successeur v de u
        Si d[u] + duree(u) > d[v] Alors
                d[v] = d[u] + duree(u)
                pred[v] = u
        FinSi
    FinPour
FinPour
```

```
16
17 # Calcule de la duree maximale
18 duree_totale =max(d[i] + duree(i)) Pour les taches i
       sans successeurs
19
20 # Reconstruction du chemin critique
21 #Initialisation:
22     chemin_critique =  [ ]
23     tache_finale = argmax(d[i] + duree}(i))
24     u=tache_finale
25 TantQue u !=0
26     Inserer u  au debut de chemin_critique
27     u = pred[u]
28 FinTantQue
29
30 Retourner duree_totale, chemin_critique
```

4 Les dates "au plus tôt" et "au plus tard"

Notée t_i c'est la date de début le plus tôt possible de l'évènement i. Elle se détermine comme suit : On note 1 le sommet initial et dij la durée de la tâche comprise entre les sommets consécutifs i et j on a :

$$\begin{cases} t_1 = 0 \\ t_j = \max(t_i + d_{ij}) \\ i \in P(j) \end{cases}$$

Notée t_i^* c'est la date la plus tardive de début de l'évènement i sans modifier la date de fin du projet. On traite les sommets en partant de "Fin"(dernier sommet). On a :

$$\begin{cases} t_n = t_n^* \\ t_i^* = \min(t_j^* - d_{ij}) \\ j \in S(i) \end{cases}$$

5 Calcul des dates au plus tôt et au plus tard du projet, et détermination des tâches critiques

Tâche	Date au plus tôt	Date au plus tard
0	0	min(11 - 2, 7 - 7) = 0
1	0	7 - min(11 - 2, 7 - 7) = 0
2	max(0+2, 0+7) = 7	14 - 3 = 11
3	7 + 3 = 10	min(19 - 1, 15 - 1, 21 - 1) = 14
4	max(0+2, 0+7) = 7	min(19 - 8, 15 - 8, 21 - 8) = 7
5	max(10+1, 7+8) = 15	21 - 2 = 19
6	max(10+1, 7+8) = 15	16 - 1 = 15
7	max(10+1, 7+8) = 15	22 - 1 = 21
8	15 + 1 = 16	19 - 3 = 16
9	16 + 3 = 19	21 - 2 = 19
10	max(15+2, 19+2) = 21	22 - 1 = 21
FIN	max(21+1, 15+1) = 22	22

TABLE 1 – Dates au plus tôt et au plus tard pour chaque tâche

Une tâche est critique si sa date au plus tôt est égale à sa date au plus tard. Les tâches critiques sont donc : 0, 1, 4, 6, 8, 9, 10.

Diagramme MPM

Dans le graphe MPM ci-dessous, chaque tâche est représentée par un nœud rectangulaire contenant trois informations :

— **en haut à gauche** : la date au plus tôt de la tâche (en vert) ;
— **en haut à droite** : la date au plus tard (en violet) ;
— **au centre** : le numéro de la tâche (en marron et en gras).

Les arcs représentent les dépendances entre les tâches, avec un poids égal à la durée de la tâche source.

Le chemin critique est mis en évidence en rouge : il représente la suite de tâches ne pouvant subir aucun retard sans allonger la durée totale du projet. Pour ces tâches, la marge (*slack*) est nulle.

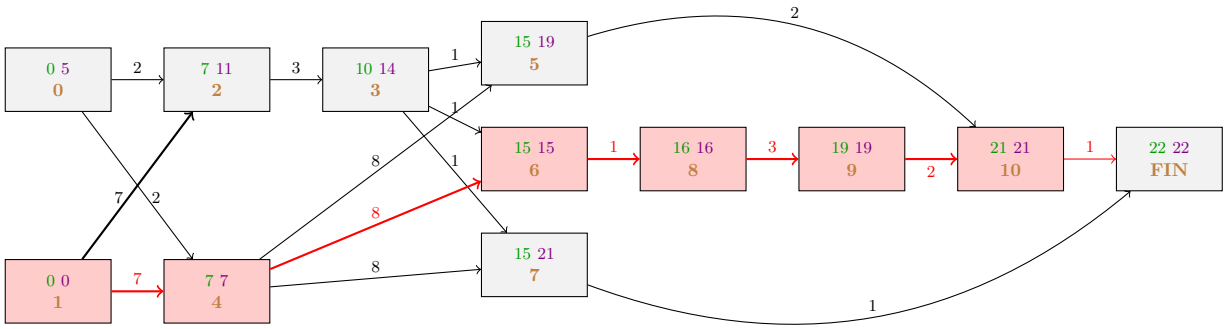

FIGURE 2 – Représentation MPM du graphe

6 Code

Voir fichier ***PlusCourtChemin.py*** joint au dossier

Références

[1] LEDOUX, Sébastien. *Algorithmes de plus court chemin*. UTC, Institut de Cybersécurité. Disponible à l'adresse : `https://ics.utc.fr/~led/co/anx_recherche_plus_long_chemin.html`
Consulté le : 20 juin 2025.

[2] PODA, Joseph. *Support de Cours de Graphes et Optimisation*. Université Nazi-Boni, École Supérieure d'Informatique, Année académique 2020-2021.