MATHÉMATIQUES

24

Graphes et ordonnancement

BTS | Mathématiques | Groupement D2

Objectifs du chapitre

- Maîtriser le vocabulaire de la théorie des graphes : sommet, arête, degré, chemin, cycle.
- Distinguer graphes orientés et non orientés ; lire et construire une matrice d'adjacence.
- Reconnaître les graphes particuliers : arbres, graphes eulériens, hamiltoniens, bipartis.
- Appliquer l'algorithme de Dijkstra pour trouver le plus court chemin dans un graphe pondéré.
- Construire un arbre couvrant minimal (algorithmes de Kruskal et Prim).
- Modéliser un projet par un réseau PERT : tâches, dépendances, dates au plus tôt et au plus tard.
- Identifier le chemin critique et calculer les marges totale et libre.
- Construire et interpréter un diagramme de Gantt.

Situation professionnelle — Chargé de projet en construction industrielle

Contexte : Maxime est chargé de projet dans une entreprise de construction de bâtiments industriels. Il doit piloter la réhabilitation d'un entrepôt logistique : démolition partielle, gros œuvre, charpente, isolation, installation électrique, plomberie, revêtements, contrôles réglementaires, puis réception du chantier.

Son client exige une livraison dans les délais contractuels sous peine de pénalités. Maxime doit donc :

- Modéliser les tâches et leurs dépendances dans un **réseau PERT**.
- Calculer la **durée minimale** du projet et identifier le **chemin critique** (les tâches qui, si elles prennent du retard, retardent tout le projet).
- Communiquer le planning à ses équipes via un **diagramme de Gantt**.
- Optimiser l'acheminement des matériaux sur le chantier en modélisant le réseau de routes par un **graphe pondéré**.

Ce chapitre vous donne les outils mathématiques pour mener ces analyses.

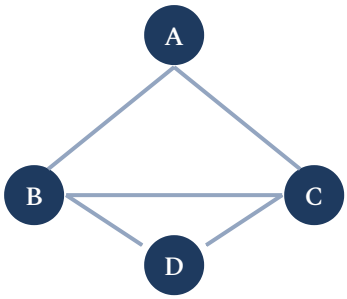
1. Théorie des graphes — notions fondamentales

1.1 Définitions de base

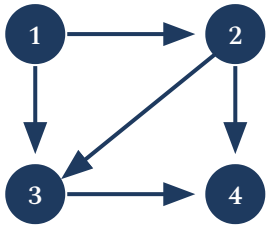
Définition — Graphe Un **graphe** $G = (V, E)$ est un couple formé de :
• un ensemble fini V de **sommets** (vertices, nœuds)
• un ensemble E d'**arêtes** (edges), chaque arête reliant deux sommets

Un graphe est **non orienté** si les arêtes n'ont pas de sens ($\{u, v\} = \{v, u\}$). Il est **orienté** (digraphe) si les arêtes ont un sens ($(u, v) \neq (v, u)$). Dans ce cas les arêtes s'appellent des **arcs**.

Un graphe est **pondéré** si chaque arête porte une valeur numérique (poids, coût, durée, distance).



Graphe non orienté $G = (V, E)$
avec 4 sommets et 5 arêtes



Graphe orienté (digraphe)
avec 4 sommets et 5 arcs

Définition — Degré d'un sommet Dans un graphe non orienté, le **degré** $d(v)$ d'un sommet v est le nombre d'arêtes qui lui sont incidentes (on compte deux fois une boucle).

Dans un graphe orienté :
- **Degré entrant** $d^-(v)$: nombre d'arcs arrivant en v
- **Degré sortant** $d^+(v)$: nombre d'arcs partant de v

Lemme des poignées de mains :

$$\sum_{v \in V} d(v) = 2|E|$$

En particulier, le nombre de sommets de degré impair est toujours pair.

1.2 Matrice d'adjacence

Définition — Matrice d'adjacence Pour un graphe G à n sommets numérotés $1, \dots, n$, la **matrice d'adjacence** M est la matrice $n \times n$ telle que :

$$M_{ij} = \begin{cases} 1 & \text{s'il existe une arête (ou un arc) de } i \text{ vers } j \\ 0 & \text{sinon} \end{cases}$$

Pour un graphe pondéré, on remplace 1 par le poids de l'arête (et ∞ si pas d'arête).

Exemple — Graphe orienté à 4 sommets :
Arcs : $1 \to 2$, $1 \to 3$, $2 \to 4$, $3 \to 4$, $4 \to 1$.

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

La matrice est en général *non symétrique* pour un graphe orienté, et *symétrique* pour un graphe non orienté.

2. Chemins, chaînes et cycles

Définitions — Chemins et chaînes Dans un graphe non orienté :
• Une **chaîne** est une suite de sommets $v_0, v_1, \ldots, v_k$ tels que $\{v_{i-1}, v_i\} \in E$ pour tout i. La **longueur** est k.
• Un **cycle** est une chaîne dont les extrémités coïncident ($v_0 = v_k$).

Dans un graphe orienté :
• Un **chemin** est une suite d'arcs consécutifs orientés dans le même sens.
• Un **circuit** est un chemin dont les extrémités coïncident.

Définition — Connexité Un graphe non orienté est **connexe** s'il existe une chaîne entre toute paire de sommets. Un graphe orienté est **fortement connexe** s'il existe un chemin orienté dans chaque sens entre toute paire de sommets.

2.1 Parcours de graphes

Algorithme BFS — Parcours en largeur (Breadth-First Search) Explore le graphe niveau par niveau à partir d'un sommet source s. Utilise une **file** (FIFO).

Pseudo-code :

```
BFS(G, s):
  marquer s comme visité
  file ← [s]
  TANT QUE file non vide :
    v ← défiler(file)
    POUR chaque voisin u de v :
      SI u non visité :
        marquer u comme visité
        enfiler(file, u)
```

Complexité : $O(|V| + |E|)$.
Applications : plus court chemin (graphe non pondéré), test de connexité.

Algorithme DFS — Parcours en profondeur (Depth-First Search) Explore aussi loin que possible avant de revenir en arrière. Utilise une **pile** (LIFO) ou la récursion.

Pseudo-code récursif :

```
DFS(G, v):
  marquer v comme visité
  POUR chaque voisin u de v :
    SI u non visité :
      DFS(G, u)
```

Applications : détection de cycles, tri topologique, composantes connexes.

3. Graphes particuliers

3.1 Arbres et forêts

Définition — Arbre Un **arbre** est un graphe connexe sans cycle. Un arbre à n sommets a exactement $n - 1$ arêtes.
Une **forêt** est un graphe sans cycle (union d'arbres disjoints). Un **arbre couvrant** d'un graphe connexe G est un arbre contenant tous les sommets de G.

3.2 Graphe eulérien

Critère — Graphe eulérien Un graphe connexe admet un **cycle eulérien** (parcourant chaque arête exactement une fois et revenant au départ) si et seulement si **tous ses sommets sont de degré pair**.

Il admet une **chaîne eulérienne** (sans revenir au départ) si et seulement si il possède exactement deux sommets de degré impair (départ et arrivée).

Exemple historique : le problème des 7 ponts de Königsberg n'a pas de solution car les 4 sommets du graphe sont tous de degré impair.

3.3 Graphe hamiltonien

Définition Un **cycle hamiltonien** passe par chaque *sommet* exactement une fois (et revient au départ). Un graphe admettant un tel cycle est **hamiltonien**.
Contrairement au problème eulérien, il n'existe pas de critère simple en général (problème NP-complet).

3.4 Graphe biparti

Définition Un graphe est **biparti** si ses sommets peuvent être séparés en deux ensembles disjoints U et V tels que chaque arête relie un sommet de U à un sommet de V (aucune arête entre deux sommets du même ensemble).
Exemple : graphe d'affectation (opérateurs — machines).

4. Plus court chemin — Algorithme de Dijkstra

Algorithme de Dijkstra Trouve le plus court chemin depuis une source s vers tous les autres sommets dans un graphe à arêtes de **poids positifs**.

Principe : greedy (glouton) — à chaque étape, on choisit le sommet non encore traité ayant la distance provisoire minimale.

```
Dijkstra(G, s):
  dist[s] ← 0 ; dist[v] ← ∞ pour tout v ≠ s
  Q ← ensemble de tous les sommets
  TANT QUE Q non vide :
    u ← sommet de Q avec dist[u] minimale
    retirer u de Q
    POUR chaque voisin v de u :
      alt ← dist[u] + poids(u, v)
      SI alt < dist[v] :
        dist[v] ← alt
        parent[v] ← u
  RETOURNER dist, parent
```

Complexité : $O(|V|^2)$ naïf ; $O((|V| + |E|) \log |V|)$ avec file de priorité.

Exemple — Réseau de livraison

Un livreur part du dépôt A et doit rejoindre le client F. Le graphe pondéré (poids = distance en km) est :

A-B : 4 • A-C : 2 • B-D : 5 • B-E : 1 • C-B : 1 • C-D : 8 • D-F : 2 • E-F : 3

Initialisation

dist[A]=0, dist[B]=∞, dist[C]=∞, dist[D]=∞, dist[E]=∞, dist[F]=∞

Q = {A,B,C,D,E,F}

Étape 1 — choisir A (dist=0)

Voisins : B(4), C(2)

dist[B]=4, dist[C]=2

parent[B]=A, parent[C]=A

Étape 2 — choisir C (dist=2)

Voisins : B(2+1=3 < 4), D(2+8=10)

dist[B]=3, dist[D]=10

Étape 3 — choisir B (dist=3)

Voisins : D(3+5=8 < 10), E(3+1=4)

dist[D]=8, dist[E]=4

parent[B]=C

parent[D]=B, parent[E]=B

Étape 4 — choisir E (dist=4)

Voisins : F(4+3=7)

dist[F]=7, parent[F]=E

Étape 5 — choisir D (dist=8)

Voisins : F(8+2=10 > 7) — pas de mise à jour

Résultat : chemin le plus court A → C → B → E → F, distance = **7 km**.

Limitation L'algorithme de Dijkstra ne fonctionne pas avec des *poids négatifs*. Dans ce cas, utiliser l'algorithme de **Bellman-Ford** (complexité $O(|V| \cdot |E|)$) qui détecte aussi les cycles absorbants négatifs.

5. Arbre couvrant minimal

Un **arbre couvrant minimal** (ACM, ou MST en anglais) d'un graphe connexe pondéré est un arbre couvrant dont la somme des poids des arêtes est minimale. Application : relier n sites avec un câble réseau au coût minimum.

5.1 Algorithme de Kruskal

Algorithme de Kruskal Stratégie : trier les arêtes par poids croissant et les ajouter une à une, en évitant les cycles.

```
Kruskal(G):
  Trier les arêtes E par poids croissant
  T ← ensemble vide (arbre à construire)
  POUR chaque arête (u, v) dans l'ordre trié :
    SI l'ajout de (u,v) à T ne crée pas de cycle :
      ajouter (u,v) à T
  RETOURNER T
```

On utilise une structure Union-Find pour détecter les cycles efficacement.
Complexité : $O(|E| \log |E|)$.

5.2 Algorithme de Prim

Algorithme de Prim Stratégie : partir d'un sommet quelconque et à chaque étape ajouter l'arête de poids minimal reliant un sommet déjà dans l'arbre à un sommet extérieur.

```
Prim(G, r):
  T ← {r}  (sommet de départ)
  TANT QUE T ne contient pas tous les sommets :
    (u, v) ← arête de poids minimal avec u ∈ T et v ∉ T
    ajouter v à T et l'arête (u,v) à l'ACM
```

Complexité : $O(|E| \log |V|)$ avec file de priorité.

Exemple — Réseau de câblage de 5 sites :
Arêtes : A-B:4, A-C:2, B-C:5, B-D:10, C-D:3, C-E:8, D-E:7

Kruskal :
1. A-C:2 → ajouter (pas de cycle)
2. C-D:3 → ajouter
3. A-B:4 → ajouter
4. B-C:5 → cycle A-B-C-A, REJETER
5. D-E:7 → ajouter

ACM = {A-C, C-D, A-B, D-E}, poids total = 2+3+4+7 = **16**

6. Ordonnancement de projets — Méthode PERT

6.1 Principe

Définition — Réseau PERT La méthode **PERT** (Program Evaluation and Review Technique) représente un projet comme un graphe orienté sans circuit où :

• Les **arcs** représentent les **tâches** (avec leur durée).

• Les **nœuds** représentent des **étapes** (débuts/fins de tâches).

• Une arête fictive (durée 0) peut représenter une contrainte d'antériorité sans tâche réelle.

On distingue le nœud initial (début du projet) et le nœud final (fin du projet).

6.2 Dates au plus tôt et au plus tard

Calcul des dates

1. **Date au plus tôt (T^+)** d'une étape i : la date la plus tôt à laquelle cette étape peut être atteinte.

$$T_i^+ = \max_{j \in \text{prédécesseurs}} \left(T_j^+ + d_{ji} \right)$$

   On calcule par *propagation vers l'avant* (du nœud initial au nœud final), en prenant le **maximum** des chemins arrivant à i.

2. **Date au plus tard (T^-)** d'une étape i : la date la plus tard à laquelle cette étape peut être atteinte sans retarder le projet.

$$T_i^- = \min_{j \in \text{successeurs}} \left(T_j^- - d_{ij} \right)$$

   On calcule par *propagation vers l'arrière* (du nœud final au nœud initial), en prenant le **minimum** des chemins partant de i.

6.3 Marge totale et marge libre

Définition — Marges d'une tâche ij de durée d_{ij}

- **Marge totale :** $MT_{ij} = T_j^- - T_i^+ - d_{ij}$
  Retard maximum possible sur la tâche sans retarder la fin du projet.
- **Marge libre :** $ML_{ij} = T_j^+ - T_i^+ - d_{ij}$
  Retard maximum possible sans retarder le début au plus tôt des tâches suivantes.

6.4 Chemin critique

Définition — Chemin critique Le **chemin critique** est le chemin le plus long (en durée) du réseau PERT, du nœud initial au nœud final. Il donne la **durée minimale** du projet.

Une tâche est **critique** si et seulement si :

$$MT_{ij} = 0 \quad \Leftrightarrow \quad T_i^+ = T_i^- \text{ et } T_j^+ = T_j^- \text{ et } T_j^+ = T_i^+ + d_{ij}$$

Tout retard sur une tâche critique entraîne un retard de même durée sur la fin du projet.

7. Application industrielle — Réseau PERT complet

Projet : Réhabilitation d'un entrepôt industriel. Le tableau ci-dessous liste les 16 tâches, leurs durées (en jours) et leurs tâches antérieures.

Tâche	Description	Durée (j)	Antérieures	T^+ début	T^- début	T^+ fin	T^- fin	MT
A	Préparation du chantier	3	—	0	0	3	3	**0**
B	Démolition partielle	5	A	3	3	8	8	**0**
C	Évacuation des gravats	2	B	8	8	10	10	**0**
D	Fondations (gros œuvre)	8	C	10	10	18	18	**0**
E	Maçonnerie des murs	6	D	18	18	24	24	**0**
F	Charpente métallique	7	E	24	24	31	31	**0**
G	Couverture et toiture	5	F	31	31	36	36	**0**
H	Isolation thermique	4	G	36	36	40	40	**0**
I	Installation électrique (tableau)	3	E	24	38	27	41	14
J	Câblage électrique	5	I, G	36	41	41	46	5
K	Plomberie (alimentation)	4	G	36	36	40	40	**0**
L	Menuiserie et cloisons	6	H, K	40	40	46	46	**0**
M	Revêtements sol et mur	5	J, L	46	46	51	51	**0**
N	Contrôle électrique (CONSUEL)	1	J	41	50	42	51	9
O	Peinture et finitions	4	M, N	51	51	55	55	**0**
P	Réception et contrôle final	2	N, O	55	55	57	57	**0**

Chemin critique du projet Le chemin critique (marges totales nulles) est :

$$\mathbf{A} \to \mathbf{B} \to \mathbf{C} \to \mathbf{D} \to \mathbf{E} \to \mathbf{F} \to \mathbf{G} \to \mathbf{H} \to \mathbf{L} \to \mathbf{M} \to \mathbf{O} \to \mathbf{P}$$

Durée minimale du projet : **57 jours**.

Les tâches I, J, N ont des marges positives : un léger retard sur celles-ci n'allonge pas la durée du projet, mais doit rester dans les limites de la marge.

8. Diagramme de Gantt

Définition — Diagramme de Gantt Le **diagramme de Gantt** est une représentation graphique du planning d'un projet. Chaque tâche est représentée par une barre horizontale dont :

• la position en x correspond à la date de début

• la longueur correspond à la durée

Il complète le PERT en offrant une lecture directe des plages temporelles, des ressources mobilisées et des possibilités de décalage.

Le diagramme de Gantt ci-dessous représente les premières tâches du projet de réhabilitation (tâches du chemin critique en rouge, tâches avec marge en bleu) :

Tâche	j1-3	j4-8	j9-10	j11-18	j19-24	j25-31	j32-36	j37-41	j4
A — Préparation (3j)	▮								
B — Démolition (5j)		▮							
C — Évacuation (2j)			▮						
D — Fondations (8j)				▮					
E — Maçonnerie (6j)					▮				
F — Charpente (7j)						▮			
G — Couverture (5j)							▮		
I — Électricité tableau (3j) ▸ marge 14j					▒				
H — Isolation (4j) critique								▮	
J — Câblage (5j) ▸ marge 5j								▒	

Tâche critique (marge = 0) • Tâche non critique (marge > 0)

9. Coloration de graphes

Définition — Coloration Une **coloration** d'un graphe est une affectation d'une couleur à chaque sommet de sorte que deux sommets adjacents (reliés par une arête) aient des couleurs différentes. Le **nombre chromatique** $\chi(G)$ est le nombre minimal de couleurs nécessaires.

Applications :

• **Planification d'examens :** les épreuves ayant des candidats communs doivent être placées à des créneaux différents (sommets = épreuves, arêtes = conflits, couleurs = créneaux).

• **Attribution de fréquences radio :** deux antennes proches ne peuvent pas utiliser la même fréquence (sommets = antennes, arêtes = interférences potentielles).

Propriétés

- Si G est biparti et connexe : $\chi(G) = 2$.
- Si G contient un triangle (clique de taille 3) : $\chi(G) \geq 3$.
- **Théorème des 4 couleurs :** tout graphe planaire est 4-coloriable.
- Algorithme glouton : donne une coloration avec au plus $\Delta(G) + 1$ couleurs ($\Delta(G)$ = degré maximal), mais pas forcément optimale.

Méthode — Calcul du chemin critique (PERT)

1. **Lister les tâches** avec durées et antériorités (tableau).
2. **Construire le réseau PERT** : nœuds = étapes, arcs = tâches.
3. **Propagation vers l'avant :** calculer T_i^+ de gauche à droite. À chaque nœud, prendre le **maximum** des (T^+ prédécesseur + durée arc).
4. **Propagation vers l'arrière :** calculer T_i^- de droite à gauche. Au nœud final : $T^- = T^+$. À chaque nœud, prendre le **minimum** des (T^- successeur − durée arc).
5. **Calculer les marges totales :** $MT = T_j^- - T_i^+ - d_{ij}$.
6. **Identifier le chemin critique :** suite de tâches de marge totale nulle.
7. **Interpréter :** durée du projet = T^+ du nœud final = longueur du chemin critique.

À retenir — Graphes et ordonnancement

Graphes

- $G = (V, E)$: sommets et arêtes (ou arcs si orienté)
- Degré : $\sum d(v) = 2|E|$
- Eulérien (cycle) $\Leftrightarrow$ tous les degrés pairs
- Arbre : connexe, sans cycle, $n - 1$ arêtes
- Dijkstra : plus court chemin, poids ≥ 0
- Kruskal / Prim : arbre couvrant minimal

PERT / Gantt

- T^+ : propagation $\rightarrow$, prendre le MAX
- T^- : propagation $\leftarrow$, prendre le MIN
- MT = T^-(fin) – T^+(début) – durée
- Chemin critique : MT = 0 pour toutes les tâches
- Durée projet = longueur du chemin critique
- Gantt : visualisation du planning en barres temporelles

MATHÉMATIQUES

24

Graphes et ordonnancement

Graphes et ordonnancement | BTS | Mathématiques

Rappels : degré d'un sommet = nombre d'arêtes incidentes ; somme des degrés $= 2\times$ (nombre d'arêtes). **Dijkstra** : plus court chemin dans un graphe pondéré à poids positifs. **PERT** : ordonnancement, dates au plus tôt, chemin critique.

Exercice 1 — Degrés

Un graphe a 5 sommets de degrés respectifs 2, 3, 3, 2, 4.

1. Calcule la somme des degrés. **2.** En déduire le nombre d'arêtes.

Exercice 2 — Vocabulaire

Dans un graphe, qu'est-ce qu'un **cycle** ? Qu'est-ce qu'une **chaîne** ?

Exercice 3 — Plus court chemin (Dijkstra)

Graphe pondéré : arêtes $A-B=2$, $A-C=5$, $B-C=1$, $B-D=4$, $C-D=2$.

Détermine le plus court chemin de A à D et sa longueur.

Exercice 4 — Ordonnancement PERT

Tâches : A (durée 3, sans antécédent), B (2, après A), C (4, après A), D (2, après B et C).

1. Calcule les dates de fin au plus tôt de chaque tâche.

2. Donne la durée minimale du projet et le chemin critique.

Exercice 5 — Arbre couvrant (type BTS)

Pour relier 4 sites par fibre au moindre coût, on cherche un arbre couvrant minimal. Un arbre couvrant d'un graphe à n sommets possède combien d'arêtes ?

MATHÉMATIQUES

24

Graphes et ordonnancement

Graphes et ordonnancement | BTS | Mathématiques

Durée : 1 heure **Calculatrice :** autorisée **Barème :** 20 points

Documents : non autorisés

Exercice 1 — Degrés et arêtes *(5 points)*

Un graphe a 6 sommets de degrés 1, 2, 3, 3, 4, 5.

1. Somme des degrés ? *(2 pts)* **2.** Nombre d'arêtes ? *(3 pts)*

Exercice 2 — Plus court chemin *(8 points)*

Arêtes pondérées : $A-B=4$, $A-C=2$, $C-B=1$, $B-D=5$, $C-D=8$.

Détermine le plus court chemin de A à D et sa longueur.

Exercice 3 — PERT *(7 points)*

Tâches : A (4, —), B (3, après A), C (5, après A), D (2, après B), E (3, après C et D).

1. Dates de fin au plus tôt. *(4 pts)*

2. Durée du projet et chemin critique. *(3 pts)*