Graphes sans circuits et applications

GRE

6 - Graphes sans circuits

Abstract

Definition

Table des matières

1. Graophes sans circuits et applications

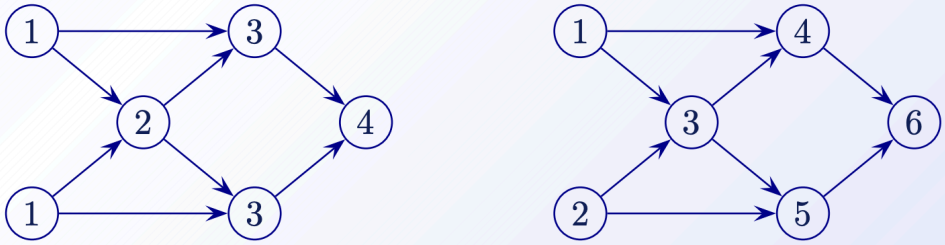
Les graphes sans circuits parfois appelé **graphes orienté acycliques** jouent un rôle important dans de nombreux problèmes tel que de la gestion de projets complexe, compilation séparée ou encore de l'ordonnancement d'activités.

1.1. Rang

Les graphes sans circuits permettent d'introduire une notion de `rang` qui permet d'ordonnancer les sommets d'un graphe.

$$\text{rang}(u) < \text{rang}(v)$$

EXEMPLE. Deux fonctions de rang pour le même graphe sans circuits, seule celle de droite est une *numérotation compatible*.



1.2. Propriétés

Note

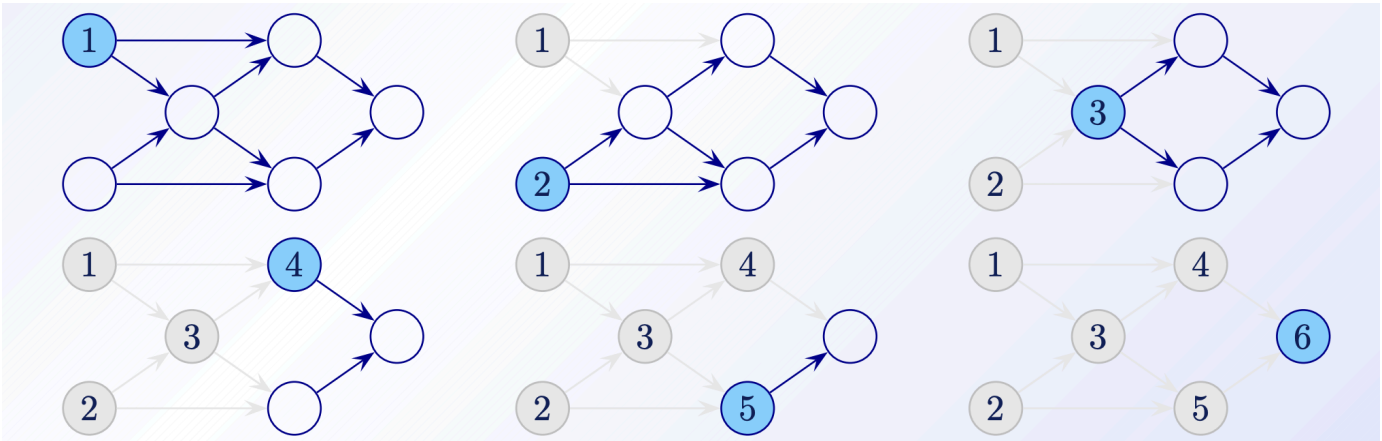
- Tout graphe **fini** sans circuits possède au moins un sommet sans prédécesseurs et au moins un sommet sans successeurs.
- Tout sous-graphe partiel d'un graphe sans circuits est également sans circuits.

1.3. Tri topologique (Kahn)

Le tri topologique est un algorithme qui permet d'ordonner les sommets d'un graphe orienté acyclique de manière à respecter les relations d'ordre entre les sommets. Il existe plusieurs algorithmes pour effectuer un tri topologique, mais l'un des plus courants est l'algorithme de Kahn.

Le concept est le suivant:

```
Tant qu'il reste des sommets non numérotés faire
  Identifier un sommet sans prédécesseurs
  Le numéroter (à la suite de celui de l'itération précédente)
  Le supprimer du graphe
```



1.3.1. Complexité

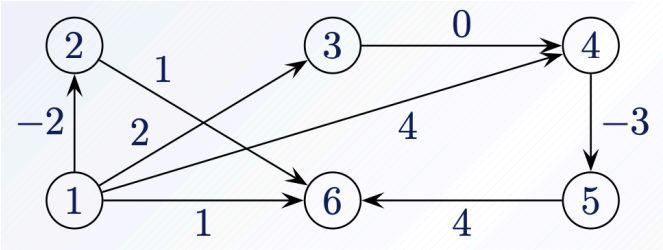
Cet algorithme a une complexité de $O(n+m)$ où n est le nombre de sommets et m le nombre d'arêtes du graphe. On parle donc de complexité linéaire.

1.4. Plus court chemin dans un graphe

Pour trouver le plus court chemin nous pouvons utiliser l'équation de **Bellman** dans le cas où les sommets sont traités dans un ordre compatible **avec le rang** et que le graphe ne contient pas de circuits.

$$\lambda_j = \min(\lambda_i + c_{ij})$$
$$i \in \mathrm{Pred}[j]$$

En appliquant ces équations nous pouvons trouver les valeurs suivantes:



$$\begin{aligned}
\lambda_1 &= 0 \\
\lambda_2 &= \lambda_1 + c_{12} = 0 + (-2) = -2 \\
\lambda_3 &= \lambda_1 + c_{13} = 0 + 2 = 2 \\
\lambda_4 &= \min(\lambda_1 + c_{14}, \lambda_3 + c_{34}) = \min(0 + 4, 2 + 0) = 2 \\
\lambda_5 &= \lambda_4 + c_{45} = 2 + (-3) = -1 \\
\lambda_6 &= \min(\lambda_1 + c_{16}, \lambda_2 + c_{26}, \lambda_5 + c_{56}) \\
&= \min(0 + 1, -2 + 1, -1 + 4) = -1
\end{aligned}$$

$$\begin{aligned}
p[1] &= — \\
p[2] &= 1 \\
p[3] &= 1 \\
p[4] &= 3 \\
p[5] &= 4 \\
p[6] &= 2
\end{aligned}$$

Une fois que nous avons calculé les valeurs de λ_j pour tous les sommets (grâce à l'agorithme **PCC_SansCircuits**), nous pouvons reconstruire le plus court chemin en suivant les prédécesseurs.

Dans ce cas la, le plus court chemin de 1 à 6 est $1 - 2 - 6$ avec un cout de -1.

1.5. Plus long chemin dans un graphe

Pour trouver le plus long chemin dans un graphe orienté acyclique, nous pouvons utiliser une approche similaire à celle du plus court chemin, mais en maximisant plutôt qu'en minimisant les coûts. L'équation de **Bellman** pour le plus long chemin est la suivante:

$$\lambda_j = \max(\lambda_i + c_{ij})$$
$$i \in \mathrm{Pred}[j]$$

2. Graphes potentiels-tâches

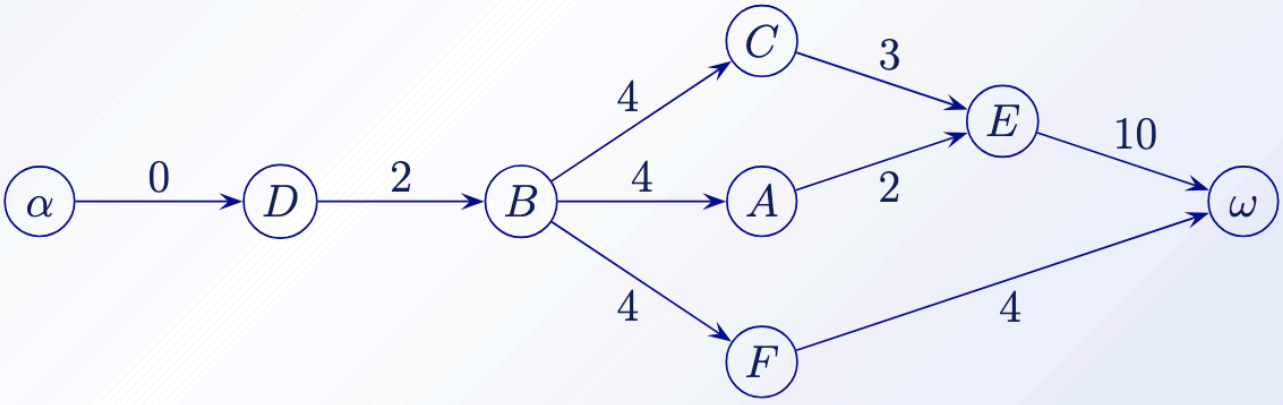
Un graphe potentiel-tâches est un graphe orienté acyclique qui représente les relations entre les tâches d'un projet.

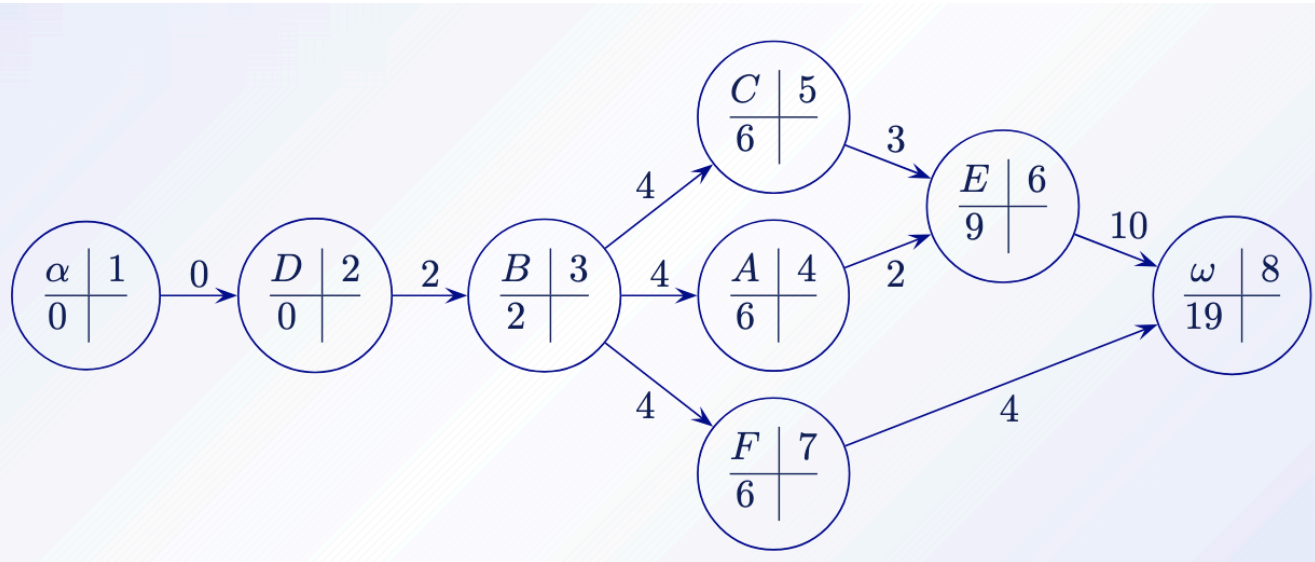
- chaque tâche est représentée par un sommet
- la contraite "la tâche i doit être terminée avant que la tâche j ne commence" est modélisée par un arc du sommet i vers le sommet j.
- le poids de l'arc (i,j) est égal à la durée d_i de la tâche i (ou, de manière plus générale, à la durée minimale à respecter entre le début de la tâche i et celui de la tâche j).

À ce graphe on ajoute

- un sommet α modélisant le début de la réalisation du projet et relié à tous les sommets sans prédécesseurs par un arc de poids nul
- un sommet w modélisant la fin de la réalisation du projet et relié à tous les sommets sans successeurs par un arc de poids nul

Tâche	Description	Durée (sem.)	Antériorités
D	Choix des stations	2	—
B	Accord administratif	4	D
C	Commande des décodeurs	3	B
A	Installation des antennes	2	B
E	Installation des décodeurs	10	C,A
F	Modification de la facturation	4	B





2.1. Composition d'un noeud

Note
Dans chaque exercice il est **crucial** de bien préciser la composition d'un noeud. Voici un exemple pour une majorité des cas:

Nom des tâches	Numéros topologiques
Date de début au plus tôt	Date de début au plus tard

2.2. Méthode du chemin critique

Cela s'applique à un graphe potentiel-tâches sans circuits dans les dommets ont été numérotés de manière compatible avec le rang.

- Dans une première phase on calcule les dates t_i de début au plus tôt des différentes tâches en calculant la longueur d'un plus long chemin entre le sommet a et le sommet i.
- Pour cela on pose $t_1 = 0$ et pour chaque sommet i on calcule la valeur de t_i en fonction des sommets prédécesseurs de i.

$$t_j = \max(t_i + d_i)$$
$$i \in \mathrm{Pred}[j]$$

- Dans une seconde phase on calcule les dates T_i de début au plus tard des différentes tâches en calculant la longueur d'un plus court chemin entre le sommet i et le sommet w. On utilise donc l'**ordre topologique inverse** et on calcule:

$$T_j = \min\big(T_k - d_j\big) = \min(T_k) - d_j$$

Une tâche i est **critique** si $t_i = T_i$. Tout retard dans l'exécution de cette tâche retarde l'exécution du projet.