

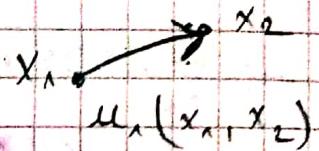
Chapitre 1

Concepts fondamentaux des graphes

• Graphes orientés :

$G_1 = (X, U)$, ensemble fini d'arcs. $m \geq 0$

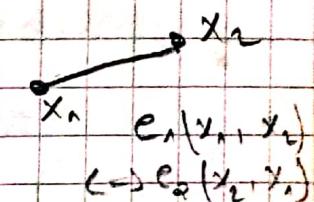
↪ ensemble fini de sommets $n \geq 1$.



• Graphes non orientés :

$G_1 = (X, E)$, ensemble fini d'arêtes $m \geq 0$

↪ ensemble fini de sommets $n \geq 1$.



• L'ordre d'un graphe :

c'est le nombre de sommets.

Si 2 extrémités sont confondues, on appelle e sur u une boucle.

• Arcs parallèles :

Si deux extrémités sont les mêmes.

• Sommets adjacents :

x, y sont deux sommets adjacents si $\exists e(x, y)$ ou $u(x, y)$.

• Ensemble des successeurs :

$\Gamma^+(x) = \{y \in X \mid (x, y) \in U\}$

$u = (x, y) :$

- x prédecesseur de y

• Ensemble des prédecesseurs :

$\Gamma^-(x) = \{y \in X \mid (x, y) \in U\}$

- y successeur de x

• Voisin :

L'ensemble de voisins $V(x)$:

$V(x) = \{y \in X \mid x \neq y \text{ et } (x, y) \in U \text{ ou } (y, x) \in U\}$

• Matrice d'adjacence :

A tout graphe d'ordre n , on associe une matrice M de n lignes et n colonnes.

→ Graphie non orienté :

$$M_{ij} = \begin{cases} 1 & : \text{si } \{x_i, x_j\} \in E \\ 0 & : \text{sinon} \end{cases}$$

M est toujours symétrique.

→ Graphie orienté :

$$M_{ij} = \begin{cases} 1 & : \text{si } (x_i, x_j) \in U \\ 0 & : \text{sinon} \end{cases}$$

• Matrice d'incidence :

A tout graphes G , on peut associer une matrice M de n lignes et m colonnes.

nombre de sommets

nombre d'arcs / arêtes.

→ Graphie non orienté :

$$M_{ij} = \begin{cases} 2 & : \text{si l'arc } j \text{ est une boucle relié au sommet } i \\ 1 & : \text{si le sommet } i \text{ est une extrémité de l'arc } j \\ 0 & : \text{Sinon} \end{cases}$$

→ Graphie orienté :

$$M_{ij} = \begin{cases} 1 & : \text{si le sommet } i \text{ est extrémité initiale de l'arc } j \\ -1 & : \text{si le sommet } i \text{ est extrémité finale de l'arc } j \\ 0 & : \text{Sinon} \end{cases}$$

• Représentation par des listes :

A tout graphe G , on peut associer 2 listes.

- PS : tab de $n+1$ éléments ($n = |X|$) .

qui représente où les arêtes lancent de x_i commence dans LS .

- LS : tab de m éléments ($m = |U|$ ou $|E|$) .

qui représente l'extrémité final d'arêtes.

• Degré d'un graphe :

→ Graph non orienté :

$d_G(x_i)$ = le nombre d'arêtes qui contiennent x_i ;

→ Graph orienté :

$dG^+(x_i)$ = le nombre d'arcs qui ont x_i comme extrémité initial.

$dG^-(x_i)$ = le nombre d'arcs qui ont x_i comme extrémité final.

$dG(x_i) = dG^-(x_i) + dG^+(x_i)$.

• Formule des degrés :

$$\sum_{x \in X} d_G(x) = 2|E| = 2|U|$$

• Graph partiel : On prend X et on prend un sous-ensemble de E/U .

G' est dit graph partiel de G avec $G' = (X, U)$, tel que $U \subset U$ ou $U \in E$. $G' = (X, E') / (X, U)$.

Sous graphe :

Un sous graphe de G est : $G = (X, \underline{E})$

$G_A = (A, \underline{E}_A)$ où A sous ensemble de \underline{U} et A sous ensemble de X .

Isomorphisme :

G_1, G_2 isomorphe :
 $G_1 \cong G_2$.

→ Graphes orientés :

Soit : $G_1 = (X_1, U_1)$, $G_2 = (X_2, U_2)$.

G_1 et G_2 sont isomorphes ssi : $\exists f : X_1 \rightarrow X_2$

$$\exists g : U_1 \rightarrow U_2$$

tel que : $\forall u \in U_1, u = (x, y) \Leftrightarrow g(u) = (f(x), f(y))$

→ Graphes non orientés :

Soit : $G_1 = (X_1, E_1)$, $G_2 = (X_2, E_2)$

G_1 et G_2 sont isomorphes ssi : $\exists \varphi : X_1 \rightarrow X_2$.

tel que : $\forall x, y \in X_1, e = x, y \in E_1 \Rightarrow \varphi(x), \varphi(y) \in E_2$

Soit : $G_1 \cong G_2$

$\forall x \in X_1, \exists y \in X_2$ tel que : $d_{G_1}(x) = d_{G_2}(y)$

Exploration d'un graphe :

- Non visité : sommet pas encore rencontré.

- Visité : sommet rencontré mais ses successeurs n'ont pas été traité.

- Exploré : sommet visité et ses successeurs ont été traité.

Algorithmus Exploration (EG: graphe, s: sommet dep) : but: utile en IA.

1/ Initialisation:

Visited = $\{x_0\}$; Explored = \emptyset ; $i = 1$; pour numérotation

Tant que (Visited $\neq \emptyset$)

Faire:

Choisir un sommet $x \in \text{Visited}$;

Visited = Visited \ { x };

Explored = Explored $\cup \{x\}$;

Pour (chaque sommet y adjacent à x)

Faire:

Si ($y \notin \text{Explored}$) Alors

Visited = Visited $\cup \{y\}$;

pere(y) = x ;

$p(x) = i$; $i = i + 1$; s numérotation des x_i

F. Si → Recherche en profondeur DFS

Fait

Visited est une pile, on choisit tjr un voisin du courant pour la prochaine itération.

Fait

→ Recherche en largeur BFS:

commence par les sommet proche du courant pour la prochaine itération.

• Multigraphie :

m_{xy} = le nombre d'arcs / arêtes qui relie x à y .

$m(G)$ = le max des m_{xy} .

G orienté est dit p-graphie si $m(G) = p$

Si $m(G) > 1$: G est appelé multigraphie.

E/U est un multi-ensemble

→ Graphie simple (non orienté) :

JP ne contient pas de boucle.

Entre chaque sommet il y a au moins une arête.

• Graphie complet :

→ Graphie orienté :

G est complet ssi : $\forall x, y \in X, (x, y) \in U$
 $\Rightarrow (y, x) \in U$

→ Graphie non orienté :

G est complet ssi : $\forall x \neq y \in X, \{x, y\} \subseteq E$

• Graphie régulier :

G est K-régulier si : $\forall x \in X, d_G(x) = k$.

• Graphie biparti :

G est biparti ssi : X admet une partition en 2 sous-ensembles X_1 et X_2 tel que : $X_1 \cap X_2 = \emptyset$ et $X_1 \cup X_2 = X$

→ Graphie orienté :

$\forall (x, y) \in U \Rightarrow x \in X_1$ et $y \in X_2$.

→ Graphhe non orienté :

$\forall x, y \in E \Rightarrow x \in X_1 \text{ et } y \in X_2$

ou:

$x \in X_2 \text{ et } y \in X_1$

bipartie complet

ssi : G bipartie et

$\forall x \in X_1, y \in X_2 \Rightarrow (x, y) \in E$

noté : $K_{p,q}$

$|X_1| = p, |X_2| = q$

• Graphhe symétrique :

ssi : $\forall x, y \in X, (x, y) \in E \Rightarrow (y, x) \in E$

• Graphhe antisymétrique :

ssi : $\forall x, y \in X, (x, y) \in E \Rightarrow (y, x) \notin E$

• Graphhe transitif :

ssi : $\forall x, y, z \in X, (x, y) \in E \text{ et } (y, z) \in E \Rightarrow (x, z) \in E$

• Complément d'un graphhe :

$G = (X, E)$, $\bar{G} = (X, \bar{E})$ complément de G

avec : $\bar{E} = \{(x, y) \in X^2 \mid x \neq y \text{ et } (x, y) \notin E\}$

• Coloration dans les graphes :

→ Stables :

$S \subseteq X$, un sous ensemble qui contient des sommets qui n'ont aucune arête qui les relie.

→ Cliques :

$C \subseteq X$, un sous ensemble qui forme un graphhe complet.

→ Coloration :

Colorer les sommets d'un graphhe, tel que $x_i \in C$ n'ont pas la même couleur.

→ Nombre chromatique d'un graphe :

$\chi(G)$ = le plus petit K , tel que G une partition de X en K sous-ensemble stable (nbr min de couleurs)

→ Nombre de stabilité :

$\alpha(G)$ = nbr max des sommets qui constituent un sous ensemble stable (cardinal)

→ Encadrement du nombre chromatique :

$$\chi(G) \geq \frac{|X|}{\alpha(G)}$$

$$\chi(G) \geq \chi(G')$$

avec G' sous graphe de G

$$\chi(G) \geq \max_{\text{chaque}} C_i$$

$$\chi(G) \leq \Delta(G) + 1$$

max de $\deg(v)$

→ Algorithme de Welsh et Powell de coloration:

Etape 1 : Attribuer une couleur à un sommet non encore coloré de degré max, et attribuer cette même couleur à chaque sommet non adjacent à un sommet de cette couleur.

Etape 2 : Si $S' \neq \emptyset$ reste des sommets non colorés; Répéter.
Sinon : Coloration terminée.

= Graph bipartie

\Leftrightarrow

Graph 2 couleurs

Il n'offre pas forcément le nbr min de couleurs

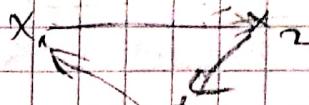
Chapitre 2 : Cheminement dans les graphes.

• Chaîne :

$\mu = x_0 e_1 x_1 \dots x_{k-1} e_k x_k$: suite de sommets et d'arcs

tg : $1 \leq i \leq k$ et $e_k = (x_{k-1}, x_k)$.

La chaîne est de longueur k .



On peut ne passer plusieurs fois par la même arête.
1/m sommets

→ le nbr
d'arêtes

• Chemin :

$\mu = x_0 u_1 x_1 \dots x_{k-1} u_k x_k$: suite de sommets et d'arcs

tg : $1 \leq i \leq k$ et $u_k = (x_{k-1}, x_k)$

Le chemin est de longueur k .

On peut passer plusieurs fois par la même arc / sommet

• Chemin / chaîne simple :

Si tous les arcs / arêtes sont distincts.

• Chemin / chaîne élémentaire :

Si tous les sommets sont distincts.

Chaîne / chemin élémentaire \Rightarrow simple

• Chemin fermé / chaîne fermée :

Si les extrémités sont confondues

• Cycle : Chaîne

Cycle = fermé + simple

La longueur du cycle : $K = \text{nbr d'arcs/arêtes}$

Si $\delta(G) > K \geq 2 \Rightarrow G$ contient un cycle de longeur K .

↓ domin

Circuit : Chemin

Circuit = fermé + simple ($x_0 = x_n$) et $K > 0$.

La longueur de circuit = K .

Boucle :

cycle / circuit
de longueur 1

Circuit élémentaire :

Circuit où tous les sommets sont distincts.

δ^+ δ^-

Connexité : Graphe non orienté

→ Graphe connexe : ~~un sous graphe avec tous les sommets~~

$\forall x, y \in X, x \neq y \Rightarrow \exists$ une chaîne qui les relient.

→ Composante connexe (CC) :

un sous ensemble de sommets qui engendre un

sous graphe connexe.

Un graphe d'ordre n connexe, a au moins $n-1$ arêtes.

→ Graphe K -connexe :

en supprimant moins de k sommets, il reste connexe.

Forte connexité : Graphe orienté

→ Graphe fortement connexe :

$\forall x, y \in X, x \neq y \Rightarrow \exists$ un chemin de x à y ($x \rightarrow y$)
et \exists un chemin de y à x ($y \rightarrow x$)

→ Composante fortement connexe (FC)

un sous ensemble de sommets qui engendre

un sous graphe fortement connexe.

Point d'articulation : si on l'enlève le graphe n'est plus connexe

→ Algorithme de calcul des composante fortement connexes

• L'ensemble des descendants de x :

$$D(x) = \{y \in X \mid x \leq y\}$$

• L'ensemble des ascendants de x :

$$A(x) = \{y \in X \mid y \leq x\}$$

1) On calcule l'ensemble des descendants de x (D).

2) On calcule l'ensemble des ascendants de x (A)

3) La CFC à laquelle appartient x est $D \cap A$.

Fonction Descendants (sommets) Fonction Ascendants (sommets)

Debut:

$$U = \{x\};$$

while ($U \neq \emptyset$) do :

 Choisir un sommet $s \in U$;

$$U = U \setminus \{s\};$$

 foreach $s' \in \text{succ}(s)$ do :

 if ($s' \notin D$) Alors :

$$U = U \cup \{s'\};$$

$$D = D \cup \{s'\};$$

 end if;

end foreach;

end while;

Fin

Debut:

$$U = \{x\};$$

while ($U \neq \emptyset$) do :

 Choisir un sommet $s \in U$;

$$U = U \setminus \{s\};$$

 foreach $s' \in \text{pred}(s)$ do :

 if ($s' \in A$) Alors :

$$U = U \cup \{s'\};$$

$$A = A \cup \{s'\};$$

 end if;

end foreach;

end while.

Fin

Programme Calcul CFC

Debut :

$$N = \{X\}; R = 1;$$

While ($U \neq \emptyset$) do :

 choisir un sommet $x \in U$;

$$C_R = \{x\} \cup \text{Descendants}(x) \cap \text{Ascendants}(x);$$

$$N = U \setminus C_R;$$

$$R++;$$

endwhile;

Fin

→ Graphie réduit :

Soit : $G = (X, U)$.

Un graphie réduit ne possède pas de circuits

$G_R = (X_R, U_R)$ est le graphie réduit de G .

Avec : $X_R = \{C_i, C_j\}$ chaque CFC C_i de G , correspond

un sommet C_i .

- $U_R = \{(C_i, C_j) \mid i \neq j, \text{ et } \exists x \in C_i \text{ et } \exists y \in C_j$
et $(x, y) \in U\}$.

• Cocycles et cocircuits :

→ Cocycle :

Soit : $G = (X, E)$ un graphie non orienté, $S \subseteq X$.

$\omega(S) = \{e = \{x, y\} \in E \mid (x \in S, y \notin S) \text{ ou } (x \notin S, y \in S)\}$

Cocycle dans G associé à S .

~~soit $G = (X, U)$~~

- Soit : $G = (X, U)$ un graphe orienté, $S \subset X$:

$$w^+(S) = \{u = (x, y) \in U \mid x \in S, y \notin S\}$$

$$w^-(S) = \{u = (x, y) \in U \mid x \notin S, y \in S\}$$

$$w(S) = w^+(S) \cup w^-(S).$$

→ Cocircuit :

- Un cocircuit dans $G = (X, U)$ est un cocycle dans G , où tous les arcs sont orientés dans le même sens.

- Un ensemble d'arcs : $\delta \subset U$ est un cocircuit si : $\exists S \subset X \mid \delta = w(S)$ et : $w^+(S) = \emptyset$ ou $w^-(S) = \emptyset$

~~représentation graphique~~

→ représentation vectorielle d'un cocycle:

Soit : $G = (X, U)$, $|U| = m$

$\bar{\theta}$: vecteur de m éléments représentant un cocycle

$$\theta = w(S) = w^+(S) \cup w^-(S)$$

$$\sum_{u_i \in w^+(S)} 1 \quad u_i \in w^+(S)$$

$$\sum_{u_i \in w^-(S)} -1 \quad u_i \in w^-(S)$$

$$0 \quad u_i \notin w(S)$$

Chaque w_i du vecteur

w correspond à un arc $u_i \in U$

La coloration des arcs :

Deux arcs ayant une même extrémité doivent être de couleurs différentes ($u = (a, b)$ et $u' = (a, c)$).

Parcours Hamiltoniens :

→ Chaîne / Chemin Hamiltonien :

Soit : G un graphe d'ordre n .

Une chaîne / chemin élémentaire de longueur $n - 1$.

Parcours Hamiltoniens

La coloration des arcs :

Deux arcs ayant une même extrémité, doivent être de couleur différente ($u = (a, b)$, $u' = (a, c)$).

Parcours Eulériens :

- Parle une seule fois par chaque arête / arc.
- Une chaîne / chemin / cycle / circuit simple de longueur m est appelé Eulérien. $m = nb \text{ arcs / arêtes}$

→ Graphes Eulériens :

G est Eulérien, si il admet un cycle / circuit Eulérien.

- Un graphe non orienté G admet un $\frac{1}{2}$ chaîne Eulérienne du sommet $x \in V(x+y)$, ssi : $d_G(x) + d_G(y)$: impair et $\forall z \in X, z \neq x$ et $z \neq y$: $d_G(z)$: pair.

- G admet un cycle Eulerien, ssi : $\forall x \in X$, $d_G(x)$ pair.
- $G = (X, U)$ admet un circuit Eulerien, ssi : $\forall x \in X$, $d_G^-(x) = d_G^+(x)$. $\Rightarrow G$ est pseudo-symétrique
- $G = (X, U)$ admet un chemin Eulerien, ssi : $\exists a, b \in X$ tq : $d_G^+(a) = d_G^-(a) + 1$.
et : $d_G^-(b) = d_G^+(b) + 1$
et : $\forall x \in X \quad x \neq a, b \quad : \quad d_G^+(x) = d_G^-(x)$.

Parcours Hamiltoniens :

- Passe une fois par chaque sommet de G .
- G d'ordre n , une chaîne / chemin élémentaire de longeur $n-1$ est appelé chaîne / chemin Hamiltonien (ne).
- Un cycle / circuit élémentaire d'ordre n est appelé cycle / circuit Hamiltonien.

Graphes Hamiltoniens :

G est Hamiltonien, ssi : il admet un cycle / circuit Hamiltonien.

- $G(X, E)$ un graphe Hamiltonien, $\forall S \subset X$, on a : $D(G|_{X-S}) \leq 15$.
(le nombre de composantes connexes de $X-S$)
Condition nécessaire.

Un graphe simple à $n \geq 3$ sommet, avec $\min d_G \geq \frac{n}{2} \Rightarrow$ Hamiltonien.

Un graphe est Hamiltonien, ssi : sa fermiture est Hamiltonienne.

Fermiture d'un graphe :

TQ : il reste 2 sommet u et v non adjacent et $d_G(u) + d_G(v) \leq n$, on ajoute l'arête $\{u, v\}$.

Quand on ne peut plus ajouter, le graphe est donc fermé.

Chapitre 3:

Réseaux et Cheminement Optimaux

Problèmes de
cheminement

Graphes sans circuits :

Un graphe sans circuit possède :

- au moins un sommet source
- et un sommet puits.

Sommet source : $d_G(x) = \infty$

Sommet puits : $d_G(x) = 0$

Niveau d'un sommet :

noté $n(x)$ = la longueur du chemin le plus long
terminant à x

- Niveau de sommet source = 0.

- L'ensemble : $N_i = \{x \mid n(x) = i\}$, pour tout $i \geq 0$

- Si on supprime les sommet source d'un graphe,
ses nul sommets source seront de niveau 1 ensuite
jusqu'à ce que les niveaux de tout les
sommets soient déterminés.

Ordre topologique (niveaux topologiques) :

A chaque sommet x d'un graphe sans circuit

on associe un numéro : $t(x)$ pour $u = (x, y) : t(x) < t(y)$

Pour cela : on numérote niveau par niveau en
commençant par N_0, N_1, \dots, N_k .

le dernier niveau

Algorithm de Bellman

S'applique aux graphes sans circuits.

→ L'équation de Bellman:

$$\pi(y) = \min_{\forall x \in T^-(y)} (\pi(y), \pi(x) + p(x, y))$$

ensemble des

→ L'algorithme: Preu de y

Début :

$\pi(x_0) = 0$; initialiser x_0 à $[0, -]$

$\pi(x_i) = +\infty$, $\forall 1 \leq i \leq n$; initialiser la suite à $+\infty$.

Pour $i \in [1, n]$ Pour tout les x_i :

Faire:

Pour tout $x_j \in T^-(x_i)$) tout les prédecesseurs de x_i :

Faire:

Si $(\pi(x_i) > \pi(x_j) + p(x_i, x_j))$ Alors:

$$\pi(x_i) = \pi(x_j) + p(x_i, x_j);$$

$$\text{pred}(x_i) = x_j;$$

F. Si sommet précédent

Fait

$[y, x_2]$

$T(x_2)$

$+ \infty$

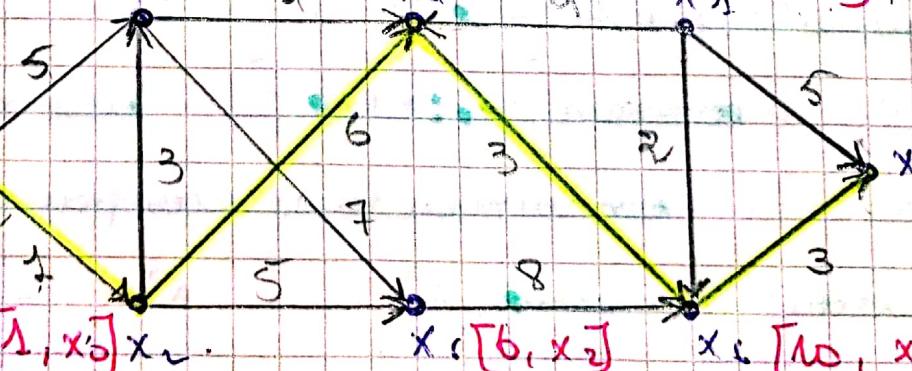
$-$

pas de

prédecesseur

Fait

$[0, -]$



le plus court chemin entre x_7 et x_0 , déduit par les pred.
 $\text{pred}(x_7) = x_6$, $\text{pred}(x_6) = x_4$, $\text{pred}(x_4) = x_1$, $\text{pred}(x_1) = x_0$

Algorithme de Dijkstra

$P(y)$ = ϵ p de ses
arcs

→ Circuit absorbant :

Un circuit y est dit absorbant, si : $P(y) < 0$.

→ L'algorithme : Circuit non absorbant et des poids non négatifs

$\pi(x_0) = 0$, initialiser x_0 à 0

$\pi(x_i) = +\infty$, $\forall 1 \leq i \leq n$; initialiser la suite π

$S = X$; $S := \emptyset$

Tant que $(S \neq \emptyset)$ Tant que il reste des sommets à traiter

Faire :

Choisir un sommet $x \in S$ tel que : $\pi(x) = \min(\pi(y))$;
retirer x de S ;

Pour tout $y \in \Gamma^+(x)$ Des successeurs de x .

Faire :

Si $(\pi(y) > \pi(x) + P(x,y))$ Alors :

$$\pi(y) = \pi(x) + P(x,y);$$

$$\text{pred}(y) = x;$$

F.Si

Fait

Fait

Detecte les
circuit lucide
absorbant

Bellman-Ford:

$$d_x^0 = +\infty, \forall x \neq s, d_s^0 = 0$$

$$d_x^K = \min \{ d_x^{K-1}, \min_{y \in \Gamma^+(x)} \{ d_y^{K-1} + p(x,y) \} \}$$

plus court chemin entre s et x comportant K arcs

L'algorithme :

but :

$$d_s^K = 0; d_x^K = +\infty, \forall x \neq s;$$

Pour ($K \leftarrow 1$ à $n-1$)

Faire :

Pour tout arc (x, y) faire

Pour tout y faire

Faire :

Si $(dy > dx + p(x,y))$ Alors :

$$dy = dx + p(x,y);$$

$$\text{OPT}(y) = x;$$

F.Si

Fait

Fait

Pour tout arc (x, y) faire

Faire :

Si $(dy > dx + p(x,y))$ Alors :

Afficher ("Existe un circuit absorbant");

F.Si

Fait

Chapitre 4 : Problème d'ordonnancement

Une tâche :

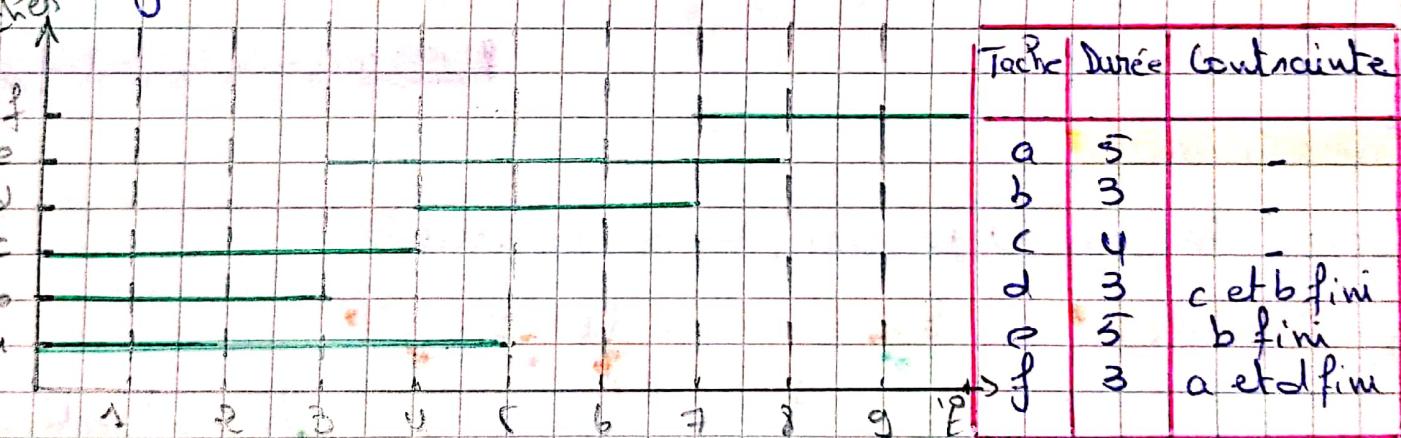
C'est l'activité élémentaire caractérisée par :

- Sa durée d_i
- Sa date de début t_i et sa date de fin f_i .
 $\Rightarrow f_i = t_i + d_i$
- Les ressources nécessaires.

La relation de précédence entre tâche est :

$$t_i \geq t_j + d_j \quad / \quad t_i - t_j \geq d_j \quad (j \text{ fini}, i \text{ commence})$$

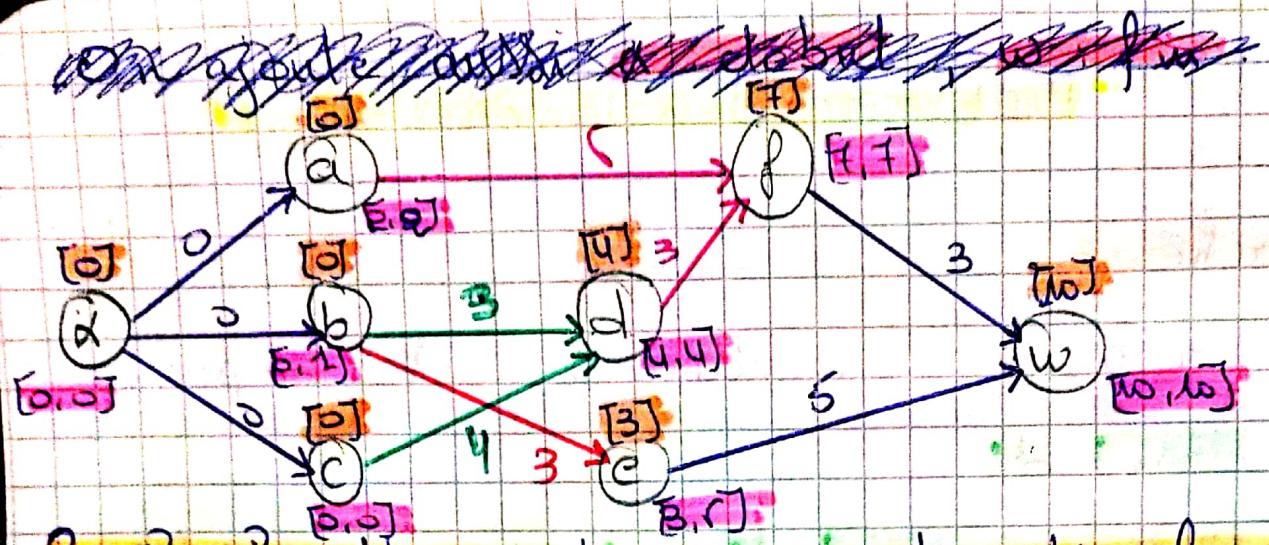
Diagramme de Gantt :



Représentation par un graphe potentiels-tâches :

- On peut représenter un problème d'ordonnancement simple par un graphe orienté $G = (X, U)$
 - X : ne représente les tâches
 - U : la contrainte de potentiels (précédence)
- α : début
- ω : fin
- $P_{ij} = a_{ij}$: poids

$$\mu = \sum \mu_i = \sum a_{ij} \quad / \quad \exists \text{ une contrainte } t_i - t_j \geq a_{ij}$$



Recherche d'un ordonnancement optimal !

- TP existe un ordonnancement réalisable, ssi : il n'existe pas de circuit de valeur ≥ 0 dans G .
- La détermination d'un calendrier optimal : déterminer $(t_\alpha, t_1, t_2, \dots, t_n, t_w)$ pour minimiser $(t_w - t_\alpha)$ sous les conditions : $t_j - t_i \geq a_{ij}$, $t_\alpha, t_1, t_2, \dots, t_w \geq 0$.

Théorème :

Dans le cas d'un graphe sans circuit :

$$\rightarrow t_\alpha = 0, \quad t_i = \max_{j \in \Gamma^-(i)} (t_j + v_{ji})$$

La date de début au plus tôt = la longueur du plus long chemin de α à i .

Évaluation de l'arc (i,j) (comme la durée)

$$\rightarrow t_w = t_w, \quad E_i = \min_{j \in \Gamma^+(i)} (F_j + v_{ij})$$

La date de début au plus tard.

Évaluation de l'arc (i,j) (comme la durée)

Xa mange :

$$St_i = T_i - t_i$$

La marge totale
d'une tâche i

Tâche	a	b	c	d	e	f
marge	2	1	0	0	2	0
totale						

-> Les tâches critiques :

$St_i = 0$: tout retard sur eux, entraîne un retard global dans le projet.

-> Un chemin critique :

1 - relie s à w .

2 - ne contient que des tâches critiques

-> La marge libre :

$$mi = \min_{K \in T^+(i)} (t_K - t_i - d_i)$$

le retard total qui on peut se permettre sur i sans remettre en cause la date de début des autres tâches.

Chapitre 5 : Les arbres

Graphe acyclique :

un graphe sans cycle

Un arbre :

un graphe connexe sans cycle.

Une forêt :

un graphe sans cycle.

Propriétés des arbres :

un arbre $T = (V, E)$ avec : $|E| = |V| - 1$.

graphie connexe minimal
(\hookrightarrow la suppression de tout les autres b décompte) graphie sans cycle comportant au moins $|E| = |V| - 1$

graphie acyclique maximal

(\hookrightarrow l'ajout de toute autre b rend cyclique)

Codage de Prüfer :

\rightarrow le codage :

Soit l'arbre : $T = (X, E)$, $X = \{x_1, x_2, \dots, x_n\}$.

Le code de T est une suite S avec $|S| = n - 2$, qui contient des nombres choisis entre $\{1 \text{ et } n\}$.

L'algorithme :

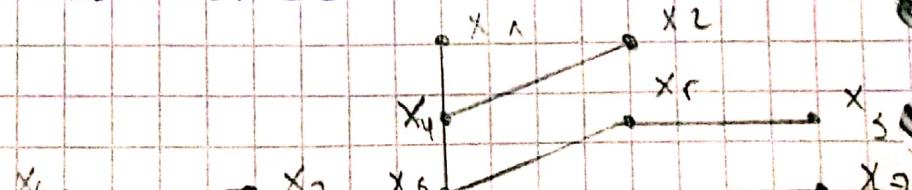
Tant que $|X| > 2$:

- Identifier la feuille de T ayant le numéro minimum.

- Ajouter à S le seul sommet adjacent à x .

- Enlever de T le sommet x et son arête incidente.

$$S = 4, 4, 5, 6, 6$$



→ Déroulage:

Soit : s , $|s| = n - 2$. Chaque élément de s est un

Possons : $I = \{1, \dots, n\}$

X l'algorithme :

Tant que ($|s| > 0$, $|I| > 2$):

Pour n sommets ($n \geq 1$)
numérotés, on peut
construire:
 n^{n-2} arbres

- Identifier le plus petit élément $i \in I$ et $i \notin s$.
- Relier par une arête de T , le sommet i à s qui est le premier élément de la suite s .
- Enlever i de I et de s .

Les 2 éléments restant dans I , c'est des extrémités de la dernière arête de T .

• L'arborescence:

→ Une arborescence : une arbre comportant un sommet, nommé racine, à partir duquel existe un chemin unique vers tous les autres sommets.

→ Une racine : ne possède pas de prédecesseurs. Ses arcs sont tous orientés de la racine vers les feuilles.

→ Les feuilles : c'est des sommets sans successeurs.

- Un sommet peut avoir plusieurs successeurs, mais un seul prédecesseur.

→ x_6 : racine, x_1, x_2, x_3, x_7 : feuilles.

• Arbre couvrant :

Un arbre couvrant pour un graphe $G = (V, E)$ est construit uniquement à partir de ses arêtes E et qui connecte (couvre) tous les sommets de V .

→ T est un arbre.

→ T est un graphe partiel de G

• Arbre couvrant de poids minimum :

→ Algorithmique Kruskal:

E : Graphe: $G = (V, E)$, C : valuation ≥ 0 des arêtes

S : $T = (V, F)$: arbre couvrant de poids minimum.

Début:

$A \leftarrow E$; $F \leftarrow \emptyset$; $i \leftarrow 0$;

Tant que ($i < |V|$) Faire:

Choisir $e \in A$ de poids minimum;

$A \leftarrow A - \{e\}$;

Si ($T = (V, F \cup \{e\})$ est acyclique) Alors:

$F \leftarrow F \cup \{e\}$; $i \leftarrow i + 1$;

Fini

Fait:

 Retourner(T);

Fin.

Graffes:

$(n-1)$
Fin d'algo

5, 5, 6, 7, 8, 9, 9, 10, 11, 12, 13

