POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

LOG1810

STRUCTURES DISCRÈTES

TD 11 : ARBRES

E2025

SOLUTIONNAIRE

Exercice 1 :

Un arbre binaire ordonné inconnu a été parcouru, produisant les deux séquences suivantes :

— **Parcours Préfixe :** `G, B, A, D, C, E, F, K, H, M, L`

— **Parcours Postfixe :** `A, C, E, F, D, B, M, H, L, K, G`

Un parcours **infixe** a également été effectué, mais la séquence a été perdue. Votre tâche est de reconstituer les informations manquantes.

a) **Reconstruisez** l'arbre binaire unique qui correspond à ces deux parcours. Dessinez clairement la structure de l'arbre, en identifiant la racine et les relations parent-enfant.

— **Racine :** Le premier élément du préfixe est **G**. Le dernier du postfixe est **G**. La racine de l'arbre est donc G.

— **Sous-arbres :**

— Préfixe : G, [`B, A, D, C, E, F`], [`K, H, M, L`]. Le sous-arbre gauche a pour racine B. Le sous-arbre droit a pour racine K.

— Postfixe : [`A, C, E, F, D, B`], [`M, H, L, K`], G. Le sous-arbre gauche se termine par B. Le sous-arbre droit se termine par K. La partition est cohérente.

— **Sous-arbre Gauche (racine B) :**

— Préfixe : B, [`A`], [`D, C, E, F`]. L'enfant gauche est A (une feuille). L'enfant droit a pour racine D.

— Postfixe : [`A`], [`C, E, F, D`], B. Cohérent.

— **Sous-arbre D (racine D) :**

— Préfixe : D, [`C`], [`E, F`]. L'enfant gauche est C (une feuille). L'enfant droit a pour racine E.

— **Sous-arbre E (racine E) :**

— Préfixe : E, [`F`]. E a un seul enfant, F. Pour maintenir la cohérence avec le préfixe, F doit être l'enfant gauche.

— **Sous-arbre Droit (racine K) :**

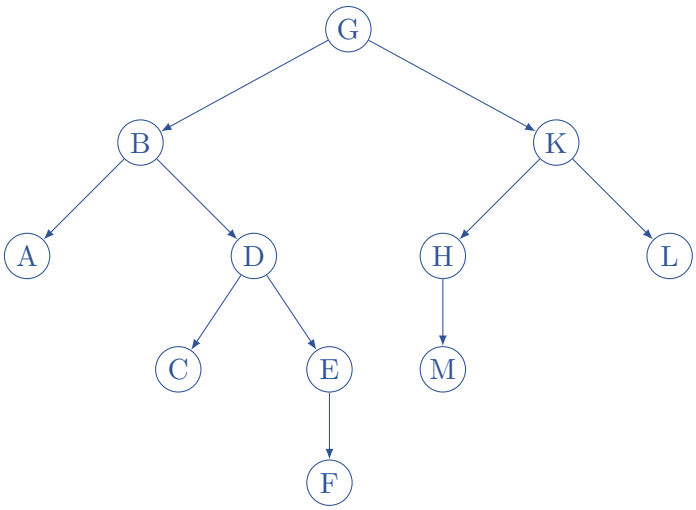
— Préfixe : K, [`H, M`], [`L`]. L'enfant gauche a pour racine H. L'enfant droit est L (une feuille).

— Postfixe : [`M, H`], [`L`], K. Cohérent.

— **Sous-arbre H (racine H) :**

— Préfixe : H, [`M`]. H a un seul enfant, M.

Arbre reconstruit :



b) À partir de l'arbre que vous avez reconstruit, déterminez la séquence de parcours **infixe**.

`A, B, C, D, F, E, G, H, M, K, L`

c) Analysez l'arbre reconstruit :

i) Quelle est la **hauteur** de l'arbre ?

Le chemin le plus long de la racine à une feuille est G → B → D → E → F. Ce chemin contient 4 arêtes. La hauteur de l'arbre est donc **4**.

ii) L'arbre est-il **équilibré** ? Justifiez votre réponse.

Un arbre de hauteur $h = 4$ est équilibré si toutes ses feuilles sont aux niveaux 4 ou 3.

— Feuilles et leurs niveaux : A (niveau 2), C (niveau 3), F (niveau 4), M (niveau 3), L (niveau 2).

L'arbre contient des feuilles aux niveaux 2, 3 et 4. Comme il y a des feuilles à un niveau qui n'est ni h ni $h - 1$ (le niveau 2), l'arbre n'est **pas équilibré**.

iii) L'arbre est-il un arbre binaire **plein** ? Justifiez.

Un arbre binaire est plein si chaque nœud interne a exactement 2 enfants.

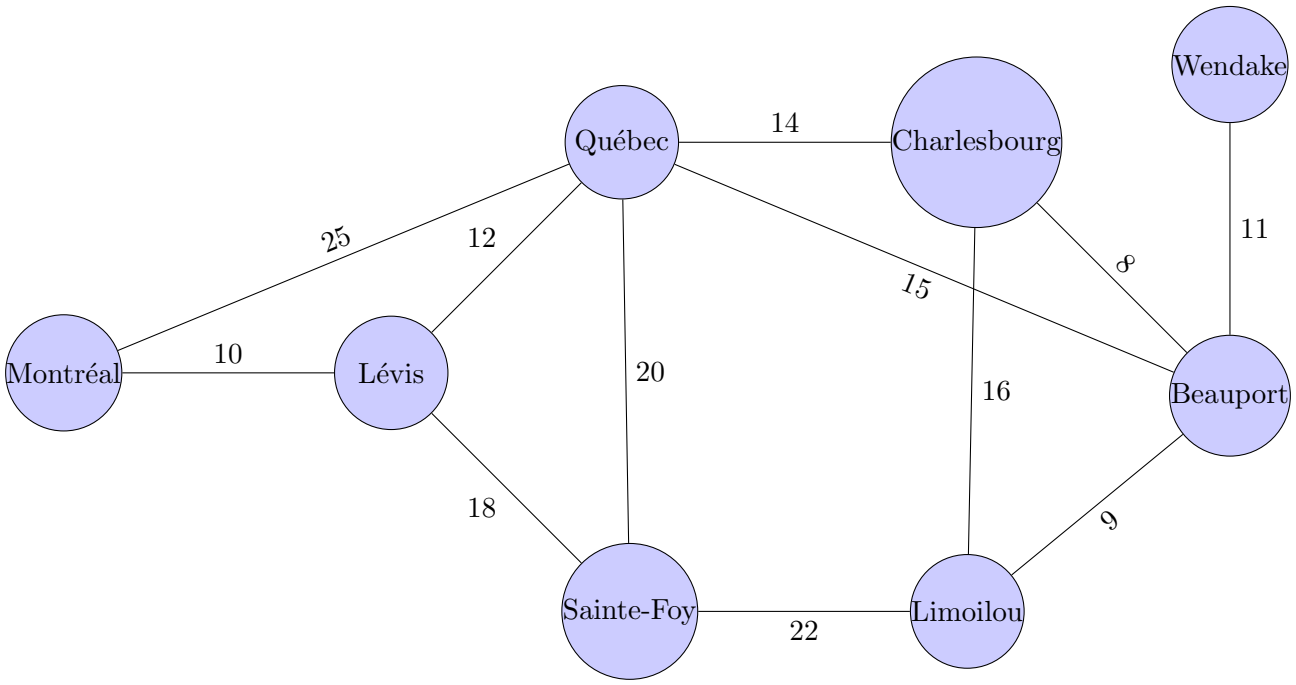
— Nœud E : a 1 enfant (F).

— Nœud H : a 1 enfant (M).

Puisque les nœuds E et H sont internes et n'ont pas 2 enfants, l'arbre n'est **pas plein**.

Exercice 2 :

Hydro-Québec planifie un nouveau réseau électrique pour relier un barrage principal situé à Montréal à plusieurs villes de la région de Québec. En raison de contraintes géographiques et de certaines priorités gouvernementales, certains corridors à haute tension doivent obligatoirement faire partie du tracé. Votre tâche est de trouver l'arbre couvrant de coût minimal tout en respectant ces contraintes.



Contraintes :

— Les liens **(Lévis, Québec)** et **(Limoilou, Beauport)** doivent impérativement faire partie du réseau, conformément à certaines directives d'aménagement priorisées par les autorités.

a) Appliquez l'algorithme de **Prim** pour déterminer un arbre couvrant de coût minimal, en commençant depuis le sommet **Montréal**.

L'algorithme de Prim est exécuté normalement, mais les arêtes obligatoires sont traitées avec une priorité absolue si elles connectent un sommet visité à un sommet non visité.

— **Init :** Visités = {}, Arbre = {}, Coût = 0, Départ : **Montréal**.

— **1.** Visités = {Montréal}
Arêtes candidates : (Montréal, Lévis)[10], (Montréal, Québec)[25]
⇒ Choisir **(Montréal, Lévis)** [coût 10]
⇒ Visités = {Montréal, Lévis}, Coût = 10

— **2.** Arêtes candidates : (Montréal, Québec)[25], (Lévis, Québec)[12], (Lévis, SainteFoy)[18]
⇒ Choisir **(Lévis, Québec)** [coût 12] (obligatoire et la moins chère)
⇒ Visités = {Montréal, Lévis, Québec}, Coût = 10 + 12 = 22

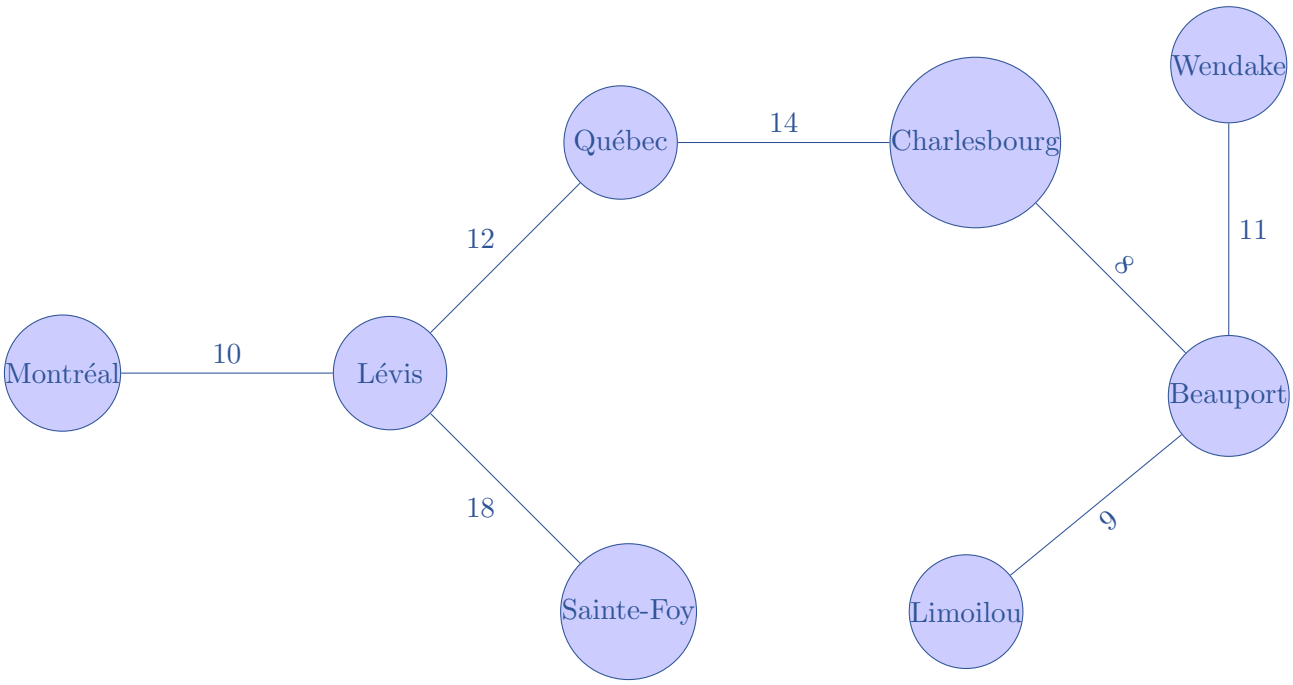
— **3.** Arêtes candidates : (Lévis, SainteFoy)[18], (Québec, Charlesbourg)[14], (Québec, Beauport)[15], (Québec, SainteFoy)[20]
⇒ Choisir **(Québec, Charlesbourg)** [coût 14]
⇒ Visités = {Montréal, Lévis, Québec, Charlesbourg}, Coût = 22 + 14 = 36

— **4.** Arêtes candidates : (Lévis, SainteFoy)[18], (Québec, Beauport)[15], (Charlesbourg, Beauport)[8], (Charlesbourg, Limoilou)[16]
⇒ Choisir **(Charlesbourg, Beauport)** [coût 8]
⇒ Visités = {Montréal, Lévis, Québec, Charlesbourg, Beauport}, Coût = 36 + 8 = 44

— **5.** Arêtes candidates : (Lévis, SainteFoy)[18], (Charlesbourg, Limoilou)[16], (Beauport, Limoilou)[9], (Beauport, Wendake)[11]
⇒ Choisir **(Beauport, Limoilou)** [coût 9] (obligatoire et la moins chère)
⇒ Visités = {Montréal, Lévis, Québec, Charlesbourg, Beauport, Limoilou}, Coût = 44 + 9 = 53

— **6.** Arêtes candidates : (Lévis, SainteFoy)[18], (Beauport, Wendake)[11], (Limoilou, SainteFoy)[22]
⇒ Choisir **(Beauport, Wendake)** [coût 11]
⇒ Visités = {Montréal, Lévis, Québec, Charlesbourg, Beauport, Limoilou, Wendake}, Coût = 53 + 11 = 64

— **7.** Arêtes candidates : (Lévis, SainteFoy)[18], (Limoilou, SainteFoy)[22], (Québec, SainteFoy)[20]
⇒ Choisir **(Lévis, SainteFoy)** [coût 18] (Tous les sommets sont connectés)
⇒ **Coût final = 64 + 18 = 82**



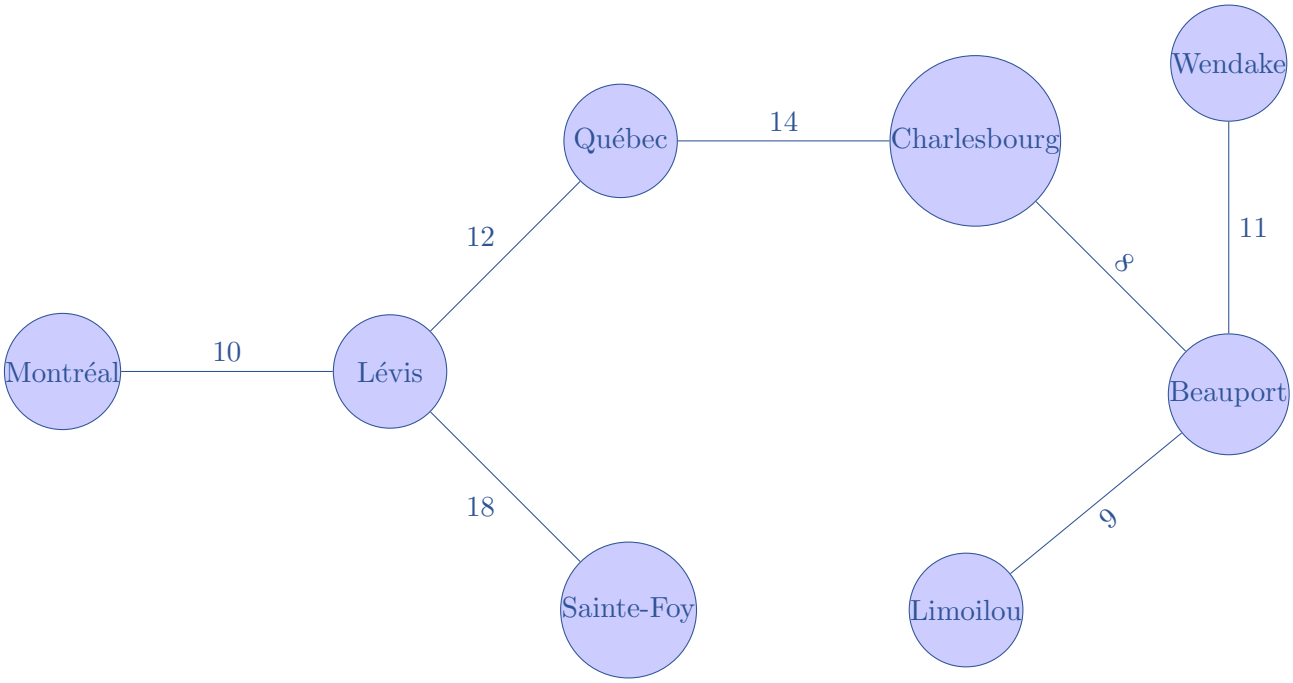
b) Appliquez l'algorithme de **Kruskal**, en adaptant la méthode pour garantir que les arêtes obligatoires soient incluses.

On inclut d'abord les arêtes obligatoires, puis on continue avec les arêtes restantes par ordre de poids, en évitant les cycles.

- **Init :** Arbre = {**(Lévis, Québec)**[12], **(Limoilou, Beauport)**[9]}
  Coût = 21
  Composantes : {L, Q}, {Li, B}, {M}, {SF}, {C}, {W}
- **Arêtes triées (non obligatoires) :**
  (Charlesbourg, Beauport)[8], (Montréal, Lévis)[10], (Beauport, Wendake)[11],
  (Québec, Charlesbourg)[14], (Québec, Beauport)[15], (Charlesbourg, Limoilou)[16],
  (Lévis, Sainte-Foy)[18], ...
- **1.** Ajouter **(Charlesbourg, Beauport)** [coût 8]
  ⇒ Connecte C à {Li, B}
  ⇒ Coût = 21 + 8 = 29
- **2.** Ajouter **(Montréal, Lévis)** [coût 10]
  ⇒ Connecte M à {L, Q}
  ⇒ Coût = 29 + 10 = 39
- **3.** Ajouter **(Beauport, Wendake)** [coût 11]
  ⇒ Connecte W à {Li, B, C}
  ⇒ Coût = 39 + 11 = 50
- **4.** Ajouter **(Québec, Charlesbourg)** [coût 14]
  ⇒ Connecte {M, L, Q} à {Li, B, C, W}
  ⇒ Coût = 50 + 14 = 64
- **5.** (Québec, Beauport)[15] Rejeter (forme un cycle)
- **6.** (Charlesbourg, Limoilou)[16] Rejeter (forme un cycle)
- **7.** Ajouter **(Lévis, Sainte-Foy)** [coût 18]
  ⇒ Connecte SF au reste

⇒ Tous les sommets sont connectés
⇒ Coût = 64 + 18 = 82

Coût final = 82



c) Comparez les deux arbres couvrants obtenus. Sont-ils identiques ? Si non, **expliquez pourquoi** ils peuvent différer malgré un même objectif de minimisation du coût.

— Arbre de Prim : {(M,Le), (Le,Q), (Q,C), (C,B), (B,Li), (B,W), (Le,SF)}
— Arbre de Kruskal : {(Le,Q), (Li,B), (C,B), (M,Le), (B,W), (Q,C), (Le,SF)}

Les deux ensembles d'arêtes sont identiques. Dans ce cas particulier, les deux algorithmes ont produit le **même arbre**.

Cependant, ce n'est **pas une garantie** en général. Les algorithmes peuvent produire des arbres différents (mais de même coût minimal) si le graphe contient plusieurs arêtes de même poids.

— **Kruskal** examine les arêtes globalement. Si deux arêtes ont le même poids, son choix dépend de l'ordre de tri.
— **Prim** grandit à partir d'un seul composant. S'il a le choix entre deux arêtes de même poids sortant du composant actuel, son choix peut être arbitraire.

Dans ce problème, bien qu'il y ait des poids non uniques, les contraintes et la structure du graphe ont guidé les deux algorithmes vers la même solution. Si par exemple l'arête (Québec, Charlesbourg) avait aussi un poids de 8, Kruskal aurait pu la choisir avant (Beauport, Charlesbourg), menant potentiellement à un arbre final différent.

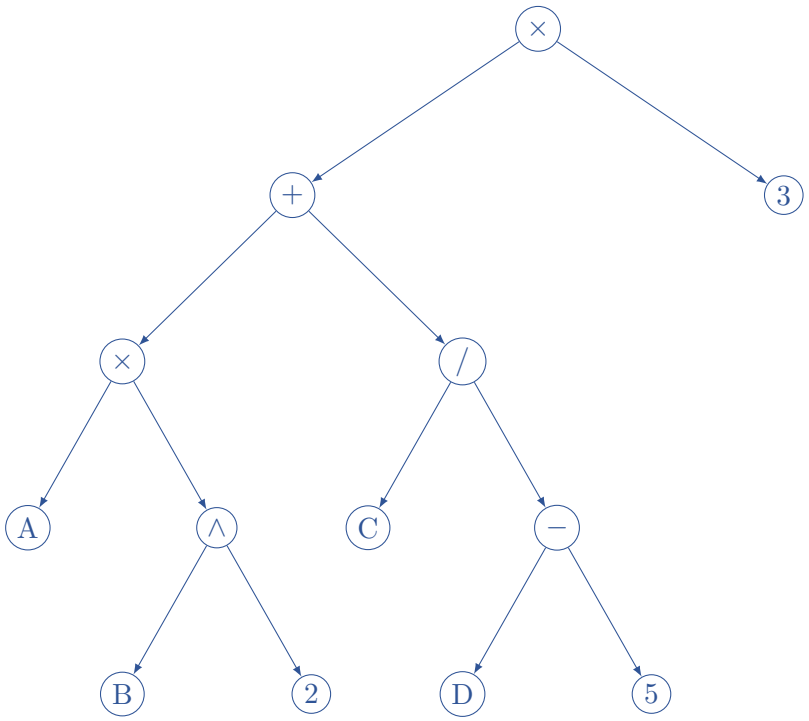
Exercice 3 :

Partie A :

Considérez l'expression arithmétique suivante, où $\wedge$ dénote l'exponentiation :

$$((A \times (B \wedge 2)) + (C/(D - 5))) \times 3$$

a) Construisez l'arbre d'expression binaire qui représente cette expression.



b) À partir de l'arbre, déterminez la notation polonaise et la notation polonaise inversée de l'expression.

Préfixe : `* + * A ^ B 2 / C - D 5 3`
Postfixe : `A B 2 ^ * C D 5 - / + 3 *`

c) Évaluez l'expression en utilisant la notation **polonaise inversée** obtenue. Montrez le contenu de la pile étape par étape pour les valeurs suivantes : $A = 2,\ B = 3,\ C = 21,\ D = 8$.

Expression : `2 3 2 ∧ * 21 8 5 - / + 3 *`

Token	**Opération**	**État de la pile (bas → haut)**
2	push 2	[2]
3	push 3	[2, 3]
2	push 2	[2, 3, 2]
∧	pop 2, pop 3 → push(3^2 = 9)	[2, 9]
×	pop 9, pop 2 → push(2×9 = 18)	[18]
21	push 21	[18, 21]
8	push 8	[18, 21, 8]
5	push 5	[18, 21, 8, 5]
−	pop 5, pop 8 → push(8 - 5 = 3)	[18, 21, 3]
/	pop 3, pop 21 → push(21÷3 = 7)	[18, 7]
+	pop 7, pop 18 → push(18+7 = 25)	[25]
3	push 3	[25, 3]
×	pop 3, pop 25 → push(25×3 = 75)	[75]

Le résultat final est **75**.

Partie B :

Soit G un graphe non orienté avec n sommets, m arêtes et k composantes connexes. On définit le *surplus cyclique* de G, noté c, comme étant le nombre minimum d'arêtes qu'il faut retirer de G pour le transformer en une forêt (c'est-à-dire, pour éliminer tous les cycles).

Démontrez que le nombre d'arêtes m du graphe G peut être exprimé par la formule suivante :

$$m = n - k + c$$

1. Soit G un graphe avec n sommets, m arêtes et k composantes connexes notées $G_1, G_2, \ldots, G_k$.
2. Pour chaque composante connexe G_j, on peut construire un **arbre couvrant** T_j. Par définition, un arbre ayant n_j sommets contient exactement $n_j - 1$ arêtes.
3. L'union de tous ces arbres couvrants,

   $$F = T_1 \cup T_2 \cup \cdots \cup T_k,$$

   forme une **forêt couvrante** de G. Cette forêt est un sous-graphe acyclique de G.
4. Le nombre total d'arêtes dans cette forêt, noté m_F, est la somme des arêtes de chaque arbre :

   $$m_F = \sum_{j=1}^{k} (n_j - 1) = \left(\sum_{j=1}^{k} n_j \right) - \left(\sum_{j=1}^{k} 1 \right) = n - k.$$

5. Le *surplus cyclique* c est le nombre minimal d'arêtes à retirer de G pour le rendre acyclique. Cela correspond aux arêtes qui ne sont pas dans F : toute arête de $G \setminus F$ introduirait un cycle si elle était ajoutée à F.
6. Ainsi, le surplus cyclique est :

   $$c = m - m_F.$$

7. En substituant $m_F = n - k$, on obtient :

   $$c = m - (n - k).$$

8. En réarrangeant cette équation, on déduit la formule :

   $$m = n - k + c.$$

■

Exercice 4 :

Un arbre m-aire plein est un arbre enraciné où chaque nœud interne a exactement m enfants. Démontrez la propriété suivante : pour tout arbre m-aire plein avec $i > 0$ nœuds internes et l feuilles, la relation suivante est toujours vraie :

$$l = (m-1)i + 1$$

Nous utilisons une preuve par **double dénombrement**, en comptant le nombre total de nœuds n de deux manières.

1. **Comptage par type de nœud :**
   Le nombre total de nœuds est la somme des nœuds internes (i) et des feuilles (l) :

   $$n = i + l \quad (*)$$

2. **Comptage par relation parent-enfant :**
   Dans un arbre enraciné, chaque nœud sauf la racine est enfant d'un parent. Seuls les nœuds internes ont des enfants.
   Dans un arbre m-aire plein, chaque nœud interne a exactement m enfants. Donc, le nombre total de relations parent-enfant (arêtes) est :

   $$m \times i$$

   Comme un arbre avec n sommets contient $n - 1$ arêtes, on a :

   $$n - 1 = m \times i \quad \Rightarrow \quad n = mi + 1 \quad (**)$$

3. **Mise en équation :**
   En égalant les expressions $(*)$ et $(**)$:

   $$i + l = mi + 1$$

4. **Isolation de l :**

   $$l = mi - i + 1 = (m-1)i + 1$$

■

Exercice 5 (facultatif) :

Partie A :

Un étudiant propose un nouvel algorithme "Prim-Kruskal-Fusion" (PKF) pour trouver un arbre couvrant de poids minimum (ACPM) dans un graphe connexe non-orienté $G = (V, E)$. L'algorithme fonctionne comme suit :

1. **Partitionnement :** Diviser l'ensemble des sommets V en k sous-ensembles disjoints non vides, $V_1, V_2, \ldots, V_k$.
2. **Phase locale (Prim) :** Pour chaque sous-ensemble V_i, exécutez l'algorithme de Prim sur le sous-graphe induit par V_i pour trouver un ACPM local, T_i. Le résultat de cette phase est une forêt de k arbres, $\{T_1, T_2, \ldots, T_k\}$.
3. **Phase de fusion (Kruskal) :** Créer un ensemble d'arêtes E_{inter} contenant toutes les arêtes de E qui relient des sommets de V_i à V_j pour $i \neq j$. Exécutez une version modifiée de l'algorithme de Kruskal sur ces arêtes E_{inter} pour connecter les k arbres de la forêt en un seul arbre, en n'ajoutant que des arêtes qui connectent des composantes (arbres) distinctes.

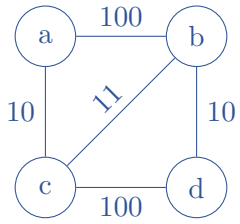
L'algorithme PKF garantit-il toujours de trouver un ACPM du graphe original G ?

Non, l'algorithme PKF ne garantit pas de trouver un ACPM.

Justification par contre-exemple :

La faiblesse de l'algorithme réside dans le fait qu'une décision « gloutonne » optimale localement (Phase 2) peut empêcher la sélection d'arêtes globalement meilleures plus tard.

Considérons le graphe suivant :



ACPM correct (par Kruskal/Prim standard) :

L'ACPM est constitué des arêtes $\{(a, c), (b, d), (b, c)\}$ pour un coût total de $10 + 10 + 11 = 31$.

Exécution de l'algorithme PKF :

1. **Partitionnement :** Divisons les sommets en $V_1 = \{a, b\}$ et $V_2 = \{c, d\}$.
2. **Phase locale (Prim) :**
   - Sur le sous-graphe induit par V_1, l'ACPM local T_1 est l'arête **(a,b)** de poids 100.
   - Sur le sous-graphe induit par V_2, l'ACPM local T_2 est l'arête **(c,d)** de poids 100.

   La forêt intermédiaire est donc $\{(a, b), (c, d)\}$ avec un coût total de 200.
3. **Phase de fusion (Kruskal) :** Les arêtes inter-partitions sont $(a, c)[10]$, $(b, d)[10]$, $(b, c)[11]$. Pour connecter les deux arbres, Kruskal choisira une arête de poids minimal, par exemple **(a,c)**.

L'arbre final produit par PKF est $\{(a, b), (c, d), (a, c)\}$ avec un coût total de

$$100 + 100 + 10 = \mathbf{210}.$$

Conclusion :

$210 \neq 31$. L'algorithme PKF a fait un choix local optimal (connecter a et b) qui s'est révélé très mauvais globalement, et il était impossible de revenir sur ce choix.

Partie B :

La société **FinBetter Inc.**, récemment apparue sur les réseaux sociaux, lance un programme de recrutement de membres promettant des revenus exponentiels. Le système commence avec le fondateur initial (au sommet de la pyramide). Chaque nouveau membre est fortement incité à **recruter exactement 4 nouvelles personnes** pour "monter en grade" et accéder à des privilèges financiers spéciaux.

On suppose que tous les membres qui s'engagent dans le recrutement réussissent à parrainer **exactement 4 personnes**. La pyramide cesse de croître une fois que **601 personnes ont été recrutées sans en recruter d'autres à leur tour** (elles sont qualifiées de "non-actives").

Chaque personne n'est recrutée qu'**une seule fois** (pas de doublons dans la pyramide).

a) Modélisez la structure de ce système à l'aide d'un arbre enraciné. Quel est le nom spécifique de ce type d'arbre dans ce contexte ? Justifiez.

La structure est un **arbre 4-aire plein**.

- **Arbre enraciné :** Il y a une origine unique (le fondateur).
- **4-aire :** Chaque recruteur parraine 4 personnes, donc chaque nœud parent a 4 enfants ($m = 4$).
- **Plein :** L'énoncé précise que les recruteurs parrainent *exactement* 4 personnes, donc chaque nœud interne a exactement 4 enfants.

b) En utilisant les propriétés des arbres m-aires pleins, calculez combien de membres ont été des recruteurs (c'est-à-dire, des nœuds internes).

Nous savons que $m = 4$ et que le nombre de feuilles (membres non-actifs) est $l = 601$. Nous utilisons la formule qui relie les feuilles (l) et les nœuds internes (i) :

$$i = \frac{l-1}{m-1}$$

$$i = \frac{601-1}{4-1} = \frac{600}{3} = \mathbf{200}$$

Il y a eu **200** recruteurs.

c) Calculez le nombre total de participants dans le réseau à la fin de la campagne.

Le nombre total de participants (n) est la somme des nœuds internes (i) et des feuilles (l).

$$n = i + l$$

$$n = 200 + 601 = \mathbf{801}$$

Le nombre total de participants est de **801**.

Feuille supplémentaire