CentraleSupélec

Algorithmique et complexité

TD 3/7 – Graphes à flots

Ce TD est l'occasion de s'exercer à la résolution de problèmes à base de graphes et d'approfondir la compréhension des problèmes traités en cours. Il adresse notamment le problème du flot maximum.

Exercice 1 : Pompe à eau

Une société de distribution d'eau dispose de 3 pompes A, B et C, pour alimenter en eau 4 villages D, E, F et G. La table 1 donne les capacités de production des pompes et la table 2 renseigne la demande des villages (en m^3/s).

Pompe	Capacité
A	45
B	25
C	30

Table 1 – Capacité de production des pompes

Village	Demande
D	30
E	10
F	20
G	30

Table 2 – Demande des villages

La situation géographique des lieux et la configuration du réseau limitent les possibilités d'approvisionnement. En effet, le réseau d'acheminement se compose de canaux directs entre pompes et villages. Un canal relie une pompe à un village avec un débit maximum à ne pas dépasser. La table 3 donne les débits maximum des canaux existants.

Pompe / Village	D	E	F	G
A	10	15	10	25
B	20		15	5
C		10	5	10

Table 3 – Capacité d'acheminement du réseau

Le problème que l'on souhaite résoudre est double :

- Peut-on satisfaire la demande ?
- Comment choisir les débits que chaque pompe doit délivrer dans chaque canal ?

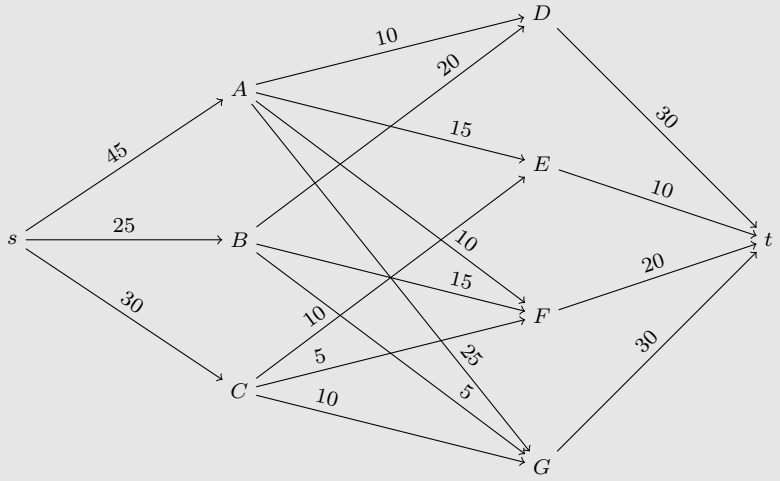
Question 1

Modéliser le problème sous forme d'un problème de flot.

Élements de correction :

Pour cela, on crée un graphe orienté avec un somment source s, un sommet terminal t, 3 sommets représentant les pompes (A, B, C) et enfin 4 sommets pour représenter les villages (D, E, F G).

Les arcs du graphe sont : de la source vers les sommets pompes, la capacité de ces arcs est la capacité de production des pompes. Les sommets pompes sont reliés aux sommets villages avec la capacité associée au réseau d'acheminement. Enfin chaque sommet village est relié au sommet terminal avec la capacité demandée.



Question 2

Appliquer l'algorithme de Ford-Fulkerson pour calculer une solution au problème.

Élements de correction :

1. on peur appliquer un **parcours en profondeur** à chaque itération de l'algo et on trouve les chaînes augmentantes suivantes :
   — $s \to A \to D \to t$: 10
   — $s \to A \to E \to t$: 10
   — $s \to A \to F \to t$: 10
   — $s \to A \to G \to t$: 15
   — $s \to B \to D \to t$: 20
   — $s \to B \to F \to t$: 5
   — $s \to C \to E \leftarrow A \to G \to t$: 10 (ici on déleste du flot entre E et A on prend l'arête à contre-sens)
   — $s \to C \to F \to t$: 5
   — $s \to C \to G \to t$: 5
2. l'algo **s'arrête** et trouve dans l'ensemble des noeuds visités la coupe de capacité min la plus proche de la source (ici il trouve la coupe : $\{s, A, C, E, G\} \cup \{B, D, F, t\}$).

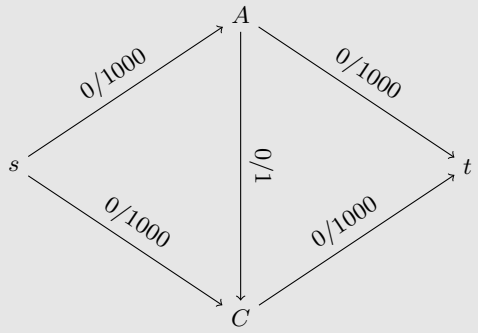
Question 3

Quelle est la complexité de l'algorithme de Ford-Fulkerson ? Comment pourrait-on la réduire ?

Élements de correction :

La complexité de Ford-Fulkerson est $\mathcal{O}((|V| + |E|) \times f_{max})$ (si on s'amuse à augmenter le flot unité par unité et donc f_{max} fois un parcours $\mathcal{O}(|V| + |E|)$)

L'algo de Ford-Fulkerson cherche une chaîne augmentante quelconque sans imposer un algo de parcours. L'exemple suivant peut nécessiter 2000 étapes (1000 chaînes augmentantes $s \to A \to C \to t$ + 1000 chaînes $s \to C \to A \to t$) ou seulement 2 étapes ($s \to A \to t$ et $s \to C \to t$) !



Une variante est d'utiliser un parcours en largeur et il s'agit de l'algo de Edmonds–Karp. La chaîne augmentante trouvée ainsi est le chemin le **plus court** en nombre d'arêtes qui possède du flot disponible. ce qui résout l'exemple précédent en 2 étapes. La complexité de l'algo de Edmonds–Karp est étonnamment $\mathcal{O}(|V| \times |E|^2)$. Cet algo ne dépend pas de la quantité du flot et présente une complexité polynomiale.

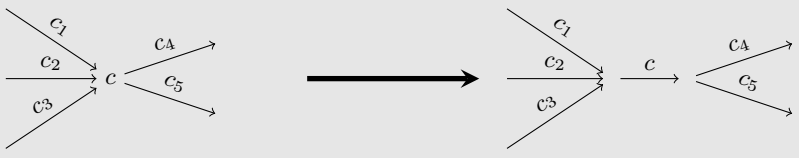
Exercice 2 : Problème de l'évasion

Question 1

Considérons une variante au problème de flot maximum où les arêtes ainsi que les sommets ont des capacités à ne pas dépasser. Proposez une transformation pour réduire ce problème au problème ordinaire du flot maximum.

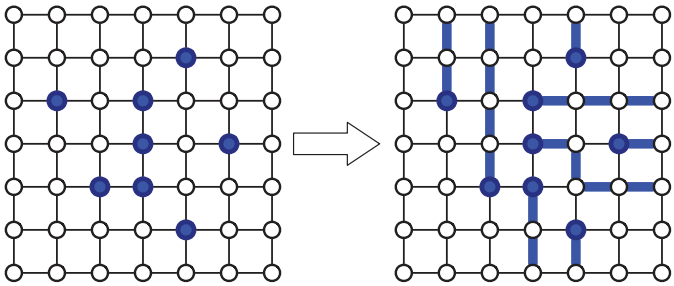
Élements de correction :

On applique la transformation illustrée ci-dessous :



Formellement, on part d'un graphe $G = (V, E)$ avec des capacités sur les sommets $C_V : V \to \mathbb{R}$ et le arêtes $C_E : E \to \mathbb{R}$, puis on crée un graphe $G' = (V', E')$ et sa fonction de capacités $C : E' \to \mathbb{R}$:

- $V' = \{v_{in} \mid v \in V\} \cup \{v_{out} \mid v \in V\}$
- $E' = \{(u_{out}, v_{in}) \mid (u, v) \in E\} \cup \{(v_{in}, v_{out}) \mid v \in V\}$
- $\forall (u, v) \in E, \quad C(u_{out}, v_{in}) = C_E(u, v)$
- $\forall v \in V, \qquad C(v_{in}, v_{out}) = C_V(v)$



Question 2

Considérons une grille carrée de $n \times n$ points. On a m prisonniers dangereux dont les positions sont m points distincts de la grille ($m < n^2$). Le problème de l'évasion consiste à savoir s'il existe m chemins disjoints (n'ayant aucun point en commun) menant ces prisonniers à la sortie de la grille sans se croiser. La figure ci-dessous illustre un tel plan d'évasion.

Donnez une solution au problème de l'évasion.

Élements de correction :

On peut le transformer à un problème de flot (à sa variante) ainsi :

1. Ajouter un sommet source s et le connecter aux m positions de départ.
2. Ajouter un sommet terminal t et le connecter aux $4n - 4$ points de sortie.
3. Chaque arête non-orientée $\{u, v\}$ de la grille est transformée en deux arêtes orientées (u, v) et (v, u).
4. Chaque arête ainsi que chaque sommet a une capacité égale à 1.
5. Il suffit de calculer le flot max et vérifier s'il vaut m ?