POLYTECH MONTPELLIER

IG 4 - Algorithmique avancée

Contrôle sur la DP et les réductions : 1h30

- Durées recommandées :
  Exo 1 (mini question coupe) + Exo 2 (mini question réduction) : 20 min en tout,
  Exo 3 (DP) + Exo 4 (Reduction) : 1h10 en tout.
- Le barème indiqué à chaque exercice (sur 23) est approximatif.
- Pour gagnez du temps, ne lisez les rappels que si besoin!

1 Rappels

1.1 Flots

Un **réseau** R est un triplet (G, s, t) où:

- $G = (V, A)$ est un graphe orienté avec une fonction de capacité c sur les arcs (chaque arc $a \in A$ a une capacité $c(a) > 0$)
- s (la source) et t (le puit) sont deux sommets particuliers de V

Soit $R = (G, s, t)$ un réseau. Un **flot** f dans le réseau R est une fonction $f : A \mapsto \mathbb{R}^+$ vérifiant les propriétés suivantes:

- **contrainte de capacité**: pour tout $a \in A$, $f(a) \leq c(a)$
- **conservation du flot**: pour tout u sauf s et t on a $f^+(v) = f^-(v)$,
  avec $f^+(v) = \sum_{vw \in A} f(vw)$ et $f^-(v) = \sum_{wv \in A} f(wv)$
- **La valeur d'un flot** f est $|f| = f^+(s) - f^-(s)$

On définit $\delta^+(X) = \{uv \in A, u \in X, v \notin X\}$, et $\Delta(v) = f^+(v) - f^-(v)$ pour tout $v \in V$. Les notations sont étendues aux sous ensembles.
Problème du FLOT :

- entrée : un réseau R
- sortie : un flot f de R
- objectif : maximiser $|f|$

1.2 Coupes

- **Une coupe** S dans un réseau de $R = (G, s, t)$ est un sous ensemble de sommets tel que $s \in S$ et $t \notin S$.
- Rappel : $\delta^+(S)$ arcs sortants de S, $\delta^-(S)$ arcs entrants de S
- **La valeur d'une coupe** est $c(S) = c(\delta^+(S))$ (et rappel, $c(\delta^+(S)) = \sum_{a \in \delta^+(S)} c(a)$)

Problème de COUPE :

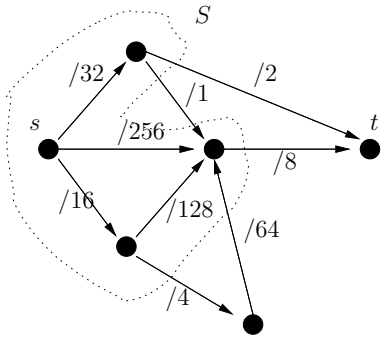
- entrée : un réseau R
- sortie : une coupe S de R
- objectif : minimiser $c(S)$

1.3 Réduction en problèmes d'optimisation

Soit Π_1 et Π_2 deux problèmes de maximisation (avec c_i leurs fonctions de coût). On dit que Π_1 se S-réduit à Π_2 (et on note $\Pi_1 \leq_S \Pi_2$) ssi il deux algorithmes poly f, g tel que

- à partir d'une instance I_1 de Π_1, f crée une instance I_2 de Π_2
- pour tout t, $\exists s_1$ solution de I_1 avec $c_1(s_1) \geq t \Leftrightarrow \exists s_2$ solution de I_2 avec $c_2(s_2) \geq t$
- pour toute solution s_2, g calcule une solution s_1 tq $c_1(s_1) \geq c_2(s_2)$

Exercice 1. *Coupe min ? (3 points)*
On considère la coupe S dans le réseau ci-dessous.



1. Combien vaut $c(S)$? (voir si besoin dans les rappels pour la définition de $c(S)$)

2. Cette coupe est-elle optimale (OUI ou NON sans justification) ?

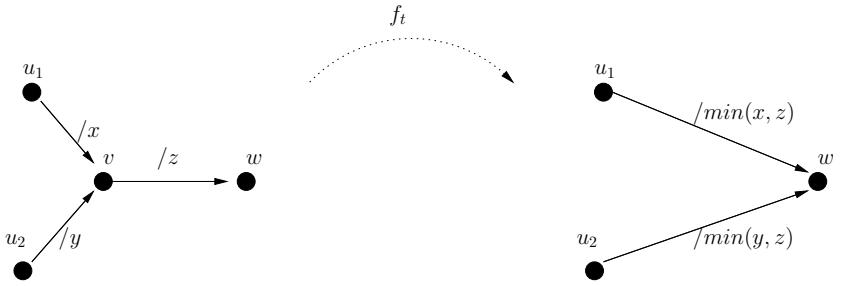
Exercice 2. *Réduction correcte ? (5 points)*
On rappelle les notations suivantes: dans un graphe orienté $G = (A, V)$, pour tout sommet u, $d^+(u) = |\{v \text{ tels que } uv \in A\}|$ est le degré sortant de u, et $d^-(u) = |\{v \text{ tels que } vu \in A\}$ est le degré entrant u.
On souhaite écrire une réduction du problème FLOT (cf rappel pour la définition du problème FLOT) vers FLOT qui supprime certain types de sommets. Pour ce faire, on commence déjà par définir la fonction suivante de traduction d'instances , notée f_t (pour fonction de Traduction, pour ne pas la confondre avec des flots). Soit $R = (G, s, t)$ un réseau, où chaque arc a de G à une capacité $c(a)$. Si on trouve un sommet v (différent de s et t) tel que (voir la figure ci-dessous) :

- $d^-(v) = 2$ et $d^+(v) = 1$, avec u_1 et u_2 les deux voisins entrants de v, et w le voisin sortant de v
- $d^+(u_1) = d^+(u_2) = 1$ (et donc le seul voisin sortant de u_1 et de u_2 est v)

alors f_t définit $R' = (G', s, t)$ obtenu à partir de R en

- supprimant le sommet v
- supprimant les arcs u_1v, u_2v, et vw
- ajoutant les arcs u_1w et u_2w
- définissant les capacités ainsi :
  - $c'(a) = c(a)$ pour tous les arcs a différents de u_1v, u_2v, et vw
  - $c'(u_1w) = \min(x, z)$, $c'(u_2w) = \min(y, z)$ où $x = c(u_1v)$, $y = c(u_2v)$, et $z = c(vw)$

(Si il n'y a pas de tel sommet v alors f_t ne fait rien et laisse $R' = R$).



1. Est ce que le sens $\Rightarrow$ de l'équivalence (cf section rappels) que devrait vérifier f_t est respecté ? (répondre soit OUI sans justifier, soit NON avec un contre exemple).

2. Est ce que le sens $\Leftarrow$ de l'équivalence (cf section rappels) que devrait vérifier f_t est respecté ? (répondre soit OUI sans justifier, soit NON avec un contre exemple).

Exercice 3. *Stable maximum dans les graphes d'intervalle (9 points)*

On considère le problème suivant noté STABLE-INTERV du stable maximum dans les graphes d'intervalle :

- entrée : n intervalles d'entiers $I_i = [a_i, b_i]$ (avec a_i et b_i entiers positifs, $a_i \leq b_i$) et n poids p_i, pour $i \in \{0, \ldots, n-1\}$. On supposera que tous les a_i et b_i sont distincts (donc les a_i et b_i sont en tout $2n$ valeurs différentes)
- sortie : un sous ensemble $S \subseteq \{0, \ldots, n-1\}$ d'indices tel que pour tout entiers distincts i, j dans S, on ait $I_i \cap I_j = \emptyset$ (c'est à dire que I_i et I_j ne doivent avoir aucun élément en commun)
- objectif : maximiser $\sum_{i \in S} p_i$, correspondant à la somme des poids des intervalles sélectionnés

Informellement, on veut donc sélectionner des intervalles ne s'intersectant pas, et de poids total maximum. Par exemple, avec $I_0 = [1, 3]$, $I_1 = [2, 4]$, $I_2 = [6, 7]$, et $p_0 = 10$, $p_1 = 100$ et $p_2 = 1000$, le sous ensemble $S = \{0, 1\}$ n'est pas une solution valide car $I_0 \cap I_1 = \{2, 3\}$ (et n'est donc pas vide), mais $S' = \{0, 2\}$ est une solution valide, de valeur 1010.
Nous allons écrire une programmation dynamique pour résoudre ce problème. Etant donné un intervalle $I_i = [a_i, b_i]$ et un entier v, on dit que I_i *est à droite de* v si $v < a_i$. Fixons une entrée du problème d'origine STABLE-INTERV), et définissons le problème auxiliaire suivant noté STABLE-INTERV-AUX :

- entrée : un entier $v \in \{0, \ldots, \Delta\}$ avec $\Delta = \max_{0 \leq i \leq n-1} b_i$
- sortie : un sous ensemble $S \subseteq \{0, \ldots, n-1\}$ d'indices d'intervalles tel que pour tout entiers distincts i, j dans S, on ait $I_i \cap I_j = \emptyset$ (c'est à dire que I_i et I_j ne doivent avoir aucun élément en commun), *et tel que pour tout* $i \in S$, I_i *est à droite de* v
- objectif : maximiser $\sum_{i \in S} p_i$, correspondant à la somme des poids des intervalles sélectionnés

1. Ecrire récursivement $aux(I_1, \ldots, I_n, p_1, \ldots, p_n, v)$ (sans pour l'instant la transformer en programmation dynamique) qui, étant donné les n intervalles et leur poids p_i "fixés", et une entrée v de STABLE-INTERV-AUX, calcule $opt(v)$ (la valeur optimale pour l'entrée v de STABLE-INTERV-AUX).

2. Ajouter (dans une autre couleur, ou recopiez ailleurs) les instructions pour transformer aux en une programmation dynamique *auxDp*, et ajouter aussi une méthode "cliente" *stableInt* qui résout le problème principal. Plus précisément, on souhaite donc avoir :

- une méthode int $auxDp(I_1, \ldots, I_n, p_1, \ldots, p_n, v, t)$ où est t est un tableau dont vous préciserez le type
- une méthode int $stableInt(I_1, \ldots, I_n, p_1, \ldots, p_n)$ qui, étant donné une instance de STABLE-INTERV donnée en paramètre, calcule la valeur optimale de l'instance de STABLE-INTERV donnée en paramètre

(Remarquez que la méthode aux de la première question n'est donc pas utilisée, elle a juste servi de version préliminaire de la méthode aux-dp.)

3. Déterminez la complexité (en temps) de la programmation dynamique de la question précédente en fonction de n et Δ.

4. Quelles modifications apporter au problème STABLE-INTERV-AUX et à l'algorithme *aux* pour que la complexité ne dépende plus que de n ? Quelle serait alors la nouvelle complexité ?

Exercice 4. *Couplage coloré (6 points)*

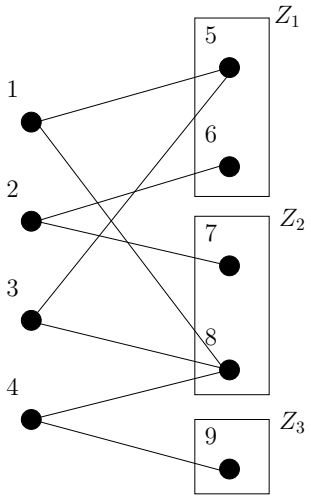
Un graphe $G = (V, E)$ est biparti ssi on peut partionner $V = V_1 \cup V_2$ tel que les V_i soient des stables (c'est à dire qu'il n'y a aucune arête à l'intérieur d'un V_i). Ainsi, toutes les arêtes sont de la forme $e = \{v_1, v_2\}$ avec $v_i \in V_i$. Etant donné un graphe, un couplage est un sous ensemble d'arêtes $E' \subseteq E$ n'ayant aucun sommet en commun ($\forall e_1, e_2 \in E', e_1 \cap e_2 = \emptyset$). On rappelle le problème suivant dit du COUPLAGE :

- entrée : un graphe biparti $G = (V, E)$ avec $V = V_1 \cup V_2$ comme ci-dessus.
- sortie : un couplage E'
- objectif : maximiser $|E'|$

On considère ici la variante suivante du problème appelée problème du COUPLAGE-COLORE :

- entrée :
  - un graphe biparti $G = (V, E)$ avec $V = V_1 \cup V_2$ comme ci-dessus.
  - une partition de V_2 en c ensembles $Z_1 \cup \cdots \cup Z_c$ (on dit que les sommets de Z_i sont coloriés de la couleur i)
- sortie : un couplage coloré E': c'est à dire un couplage $E' \subseteq E$ tel pour tout $i \in [1, c]$, $|E' \cap Z_i| \leq 1$ (chaque classe de couleur Z_i n'est touchée que par au plus une arête)
- objectif : maximiser $|E'|$

Par exemple sur la figure suivante, l'ensemble $E' = \{\{1, 8\}, \{2, 7\}, \{4, 9\}\}$ n'est pas une solution valide car il utilise deux fois la couleur Z_2, mais l'ensemble $E' = \{\{1, 8\}, \{2, 6\}, \{4, 9\}\}$ est une solution de valeur 3.



1. Montrer que COUPLAGE-COLORE $\leq_S$ COUPLAGE. Pour cela, vous détaillerez les points suivants :

- définition de votre fonction f
- définition de votre fonction g
- preuve du sens $\Rightarrow$ de l'équivalence de la définition de $\leq_S$
- preuve de la propriété requise de la fonction g (et donc entraînant le sens $\Leftarrow$ de l'équivalence)