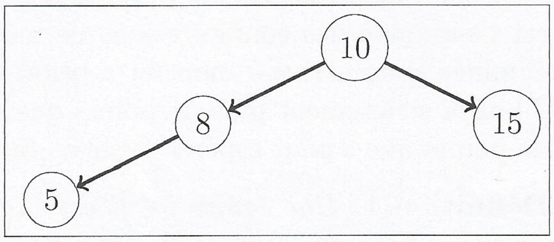
ALGO1 - Terminal

15 décembre 2015

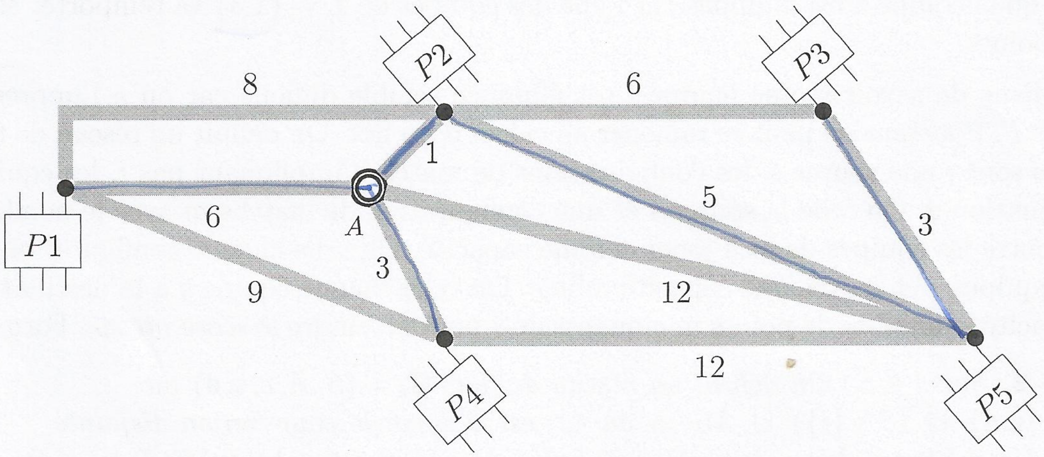
L'effort pédagogique dans les réponses est prise en compte dans l'évaluation, le résultat final n'étant pas suffisant. Écrivez assez grand. Rédigez soigneusement. Les exercices sont indépendants.

`Exercice 1`

1. **Expliquer** la technique de rééquilibrage AVL en ajoutant 2 à l'arbre binaire de recherche donné à droite.



`Exercice 2` `Connecter des broches de puces électroniques`



On souhaite connecter ensemble les broches des 5 puces électroniques $P1, \ldots, P5$ avec la prise A avec de l'étain. Le schéma indique les connexions possibles : pour chacune d'elle, le nombre indique la quantité d'étain à utiliser pour établir une connexion. Par exemple, pour connecter $P1$ à $P2$ il faut 8mg d'étain. On souhaite minimiser la quantité d'étain total utilisé. On pourrait tout connecter et ça coûterait $8+6+1+5+3+6+9+3+12+12 = 65mg$. Mais par exemple, si on décide de couler 9mg d'étain de $P1$ à $P4$ et $3mg$ de A à $P4$, il n'est pas utile de couler $6mg$ de $P1$ à A car $P1$ et A sont déjà (indirectement) connectés.

1. **Expliquer** quel problème il faut résoudre.
2. **Exécuter** un des algorithmes vus en cours pour résoudre le problème sur l'exemple.

Équipes	Nombre de matchs déjà gagnés = nombre de points déjà gagnés	Nombres de matchs restants			
		1	2	3	4
1	90	-	1	6	4
2	88	1	-	1	4
3	87	6	1	-	4
4	79	4	4	4	-

Une équipe gagne un point à chaque fois qu'elle gagne un match (il n'y a pas de matchs nuls). Soit $\mathcal{E}$ un ensemble fini d'équipes. On note p_i le nombre de points déjà gagnés de l'équipe i. Pour toutes équipes i et $j \neq i$, on note $r_{\{i,j\}}$ le nombre de matchs qu'il reste à jouer entre i et j.

Dans l'exemple, $\mathcal{E} = \{1, 2, 3, 4\}$, $p_2 = 88$ et $r_{\{1,3\}} = 6$.

Pour tout sous-ensemble d'équipes $T \subseteq \mathcal{E}$, on note $M_T = \{\{j, k\} \mid j, k \in T \text{ and } j \neq k\}$, qui est l'ensemble des configurations de matchs possibles entre les équipes dans T. Une équipe i est éliminée si on arrive à montrer a priori qu'une autre équipe j (parmi un sous-ensemble d'équipes T) aura strictement plus de points que i à la fin du tournoi. On note $maxp_i$ le nombre maximal de points que i peut espérer avoir en fin de tournoi : $maxp_i = p_i + \sum_{k\in\mathcal{E},k\neq i} r_{\{i,k\}}$. Formellement :

Définition 1. *Une équipe $i \in \mathcal{E}$ est éliminée s'il existe un sous-ensemble $T \subseteq \mathcal{E}$ tel que*

$$\frac{\sum_{j\in T} p_j + \sum_{m\in M_T} r_m}{card(T)} > maxp_i.$$

1. Montrer que l'équipe 4 est éliminée car l'une des équipes de $T = \{1,3\}$ va remporter strictement plus de points.

Le problème de savoir si une équipe i est éliminée semble difficile car on a l'impression qu'il faut deviner T. Pourtant on peut se ramener au calcul d'un flot. On définit un réseau de flots G_i où les sommets sont : une source s, les configurations de match n'impliquant pas i, les équipes sauf i et une destination d. On relie la source s et une configuration de matchs m avec le nombre restant de matchs entre les équipes de m à jouer comme capacité. On relie chaque configuration de match $\{j, k\}$ aux équipes j et k avec une capacité infinie. On relie chaque équipe j à la destination d avec comme capacité le nombre de points manquants de j pour atteindre le score $maxp_i$. Formellement :

Définition 2. *Soit $i \in \mathcal{E}$. On définit un réseau de flots $G_i = (S, A, c, \mathsf{s}, \mathsf{d})$ où*

- $S = \{\mathsf{s}, \mathsf{d}\} \sqcup (\mathcal{E} \setminus \{i\}) \sqcup M_{\mathcal{E}\setminus\{i\}}$ *où* $\sqcup$ *est le symbole pour 'union disjointe';*
- $A = \{(\mathsf{s}, m) \mid m \in M_{\mathcal{E}\setminus\{i\}}\} \sqcup \{(m, j) \mid m \in M_{\mathcal{E}\setminus\{i\}}$ *et* $j \in m\} \sqcup \{(j, \mathsf{d}) \mid j \in \mathcal{E} \setminus \{i\}\}$;
- $c : A \to \mathbb{R}^+ \cup \{\infty\}$ *définie par :*
  1. $c(\mathsf{s}, m) = r_m$ *pour tout* $m \in M_{\mathcal{E}\setminus\{i\}}$;
  2. $c(m, j) = +\infty$ *pour tout* $m \in M_{\mathcal{E}\setminus\{i\}}$ *et* $j \in m$;
  3. $c(j, \mathsf{d}) = maxp_i - p_j$ *pour tout* $j \in \mathcal{E} \setminus \{i\}$.

2. Donner un dessin du réseau pour l'exemple avec $i = 4$.

3. Exécuter l'algorithme de Ford-Fulkerson sur le réseau de flot de la question précédente.

4. Démontrer que i est éliminée ssi $|f| < \sum_{m\in M_{\mathcal{E}\setminus\{i\}}} r_m$ où $|f|$ est la valeur d'un flot maximal f de G_i.

Rédigez une démonstration structurée où les objectifs sont clairs. N'hésitez pas à indiquer les endroits où vous bloquez si vous bloquez.

`Exercice 4` `le moins de 1 possibles`

On considère des expressions arithmétiques bien formées contenant des 1 , des $+$, des $\times$ et des parenthèses. Un nombre n est représenté par plusieurs expressions. Par exemple, 14 s'écrit

$$1+1+1+1+1+1+1+1+1+1+1+1+1+1 \qquad \rightsquigarrow 14 \text{ ``}1\text{''}$$

ou alors

$$(1+1+1+1+1+1+1)\times(1+1) \qquad \rightsquigarrow 9 \text{ ``}1\text{''}$$

ou alors

$$(1+1+1)\times(1+1)\times(1+1)+(1+1) \qquad \rightsquigarrow 9 \text{ ``}1\text{''}$$

ou alors

$$((1+1+1)\times(1+1)+1))\times(1+1) \qquad \rightsquigarrow 8 \text{ ``}1\text{''}$$

1. **Concevoir** un algorithme qui donne le nombre minimal de 1 qu'une telle expression doit contenir pour représenter un nombre n.

   Dans votre réponse, on attend le nom du paradigme (diviser pour régner / glouton / programmation dynamique / réduction etc.) et une explication pédagogique sur la conception de l'algorithme. On peut utiliser le test "a divise b" qui teste si a est un diviseur de b en $O(1)$.

2. **Écrire** le pseudo-code de l'algorithme que vous avez conçu et donner sa complexité temporelle.

3. **Exécuter** votre algorithme pour calculer le nombre minimal de 1 qu'une telle expression doit contenir pour représenter $n = 14$.

`Exercice 5` `Algorithme non-déterministe`

On s'intéresse au problème de décision suivant, appelé **3-coloration** :

- Entrée : un graphe fini non orienté $G = (S, A)$;
- Sortie :
  - oui s'il existe une fonction $c : S \to \{0, 1, 2\}$ telle que $(s, t) \in A$ implique $c(s) \neq c(t)$;
  - non, sinon.

1. Afin de montrer que **3-coloration** est dans NP, **écrire** un algorithme non-déterministe en temps polynomial qui accepte exactement les instances positives du problème de **3-coloration**.