ALGO1 - Terminal

21 décembre 2022

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` `algorithme non déterministe`

On considère le problème SUBSET SUM défini comme suit :

- entrée : un tableau $T[1..n]$ de nombres entiers, un entier α
- sortie : oui, s'il existe $J \subseteq \{1, \ldots, n\}$ tel que $\sum_{j \in J} T[j] = \alpha$.

1. Écrire un algorithme non-déterministe qui décide SUBSET SUM en temps polynomial.

```
subsetsum(T[1..n], α)
    J = ∅
    pour tout j = 1 à n
        choisir de manière non déterministe β ∈ {0, 1}
        si β = 1 alors J := J ∪ {j}
    si Σ_{j∈J} T[j] = α alors
        accepter
    sinon
        rejecter
```

La première boucle est en temps $O(n)$. Le calcul de la somme consiste à réaliser $O(n)$ sommes de nombres dont le nombre de bits est de l'ordre de grandeur des nombres écrits dans T. La comparaison $=$ est linéaire en le nombre de bits dans α.
L'algorithme est bien polynomial en la taille de l'entrée, i.e. le nombre de bits qu'il faut pour stocker T et α.

`Exercice 2` `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

$1+1+1+1+1+1+1+1+1+1+1+1+1+1$ $\rightsquigarrow$ 14 "1"

ou alors

$(1+1+1+1+1+1+1)\times(1+1)$ $\rightsquigarrow$ 9 "1"

ou alors

$(1+1+1)\times(1+1)\times(1+1)+(1+1)$ $\rightsquigarrow$ 9 "1"

ou alors

$\big((1+1+1)\times(1+1)+1\big)\times(1+1)$ $\rightsquigarrow$ 8 "1"

1. **Concevoir** un algorithme `MoinsDe1` 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 3 — Elimination dans un tournoi de baseball

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. (Remarque : $r_{\{i,j\}}$ et $r_{\{j,i\}}$ désignent le même objet)
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 $maxp_i$. On suppose que $p_j \leq maxp_i$ pour tout $j \in \mathcal{E} \setminus \{i\}$. 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' ;*
— $A = \{(\mathsf{s}, m) \mid m \in M_{\mathcal{E}\setminus\{i\}}\} \sqcup \{(m,j) \mid m \in M_{\mathcal{E}\setminus\{i\}} \text{ et } j \in m\} \sqcup \{(j, \mathsf{d}) \mid j \in \mathcal{E} \setminus \{i\}\}$;
— $c : A \to \mathbb{R}^+ \cup \{\infty\}$ *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.

On rappelle théorème max-flow-min-cut : une coupe minimale a comme capacité la valeur du flot maximal. Ainsi, nous allons démontrer i est éliminée ssi la capacité d'une coupe min dans G_i est $< \sum_{m \in M_{\mathcal{E}\setminus\{i\}}}$.

Lemme 1. *Il y a une coupe minimale qui est de la forme* $C(T) = (\{s\} \cup M_T \cup T, \textit{le reste})$ *pour un certain* $T \subseteq \mathcal{E} \setminus \{i\}$.

Démonstration. Considérons une coupe minimale (A, B). Montrons comment faire pour que A soit de la forme $\{s\} \cup M_T \cup T$. Pour cela, on pose T l'ensemble des équipes dans A. On sait que s est dans A (par définition d'une coupe).
Par l'absurde, supposons $jk \in A$ avec $j \notin T$. Alors l'arc $jk - j$ est de capacité infinie et va de A dans B. Bref la coupe est de capacité infinie et donc n'est pas minimal.
Maintenant, il est possible d'avoir $jk \in B$ mais $j, k \in T$. Mais alors déplacer jk dans A fait baisser la valeur de la coupe. Ainsi, on obtient une coupe minimale est de la forme $(\{s\} \cup M_T \cup T, \text{le reste})$. □

Lemme 2. *La capacité* $|C(T)|$ *de* $C(T)$ *est*

$$\sum_{m \in M_{\mathcal{E}\setminus\{i\}}} r_m + |T| max p_i - (\sum_{j \in T} p_j + \sum_{m \in M_T} r_m).$$

Démonstration. Le calcul donne que la capacité de $C(T)$ vaut

$$\sum_{j \in T} (max p_i - p_j) + \sum_{m \notin M_T} r_m.$$

En triturant, on a l'expression du lemme. □

Conclusion.

$$\begin{gathered} i \text{ est éliminée} \\ \text{ssi (par définition)} \\ \text{il existe } T \subseteq \mathcal{E} \setminus \{i\} \text{ tel que } \frac{\sum_{j \in T} p_j + \sum_{m \in M_T} r_m}{card(T)} > max p_i. \\ \text{ssi (par le lemme 2)} \\ \text{il existe } T \subseteq \mathcal{E} \setminus \{i\} \text{ tel que } |C(T)| < \sum_{m \in M_{\mathcal{E}\setminus\{i\}}} r_m \\ \text{ssi (par le lemme 1)} \\ \text{une coupe minimale est de capacité } < \sum_{m \in M_{\mathcal{E}\setminus\{i\}}} r_m. \end{gathered}$$