# TD 04 – Union-Find

**Exercice 1.** *Union-Find structure*

On s'intéresse à la question de représenter de manière efficace des partitions d'un ensemble, de manière dynamique (c'est à dire que la partition ou l'ensemble peuvent changer au cours du temps, par exemple parce qu'on fusionne deux classes d'équivalence ou qu'on rajoute un élément à l'ensemble). Plus précisément, on veut disposer d'un type de données dans lequel les classes d'équivalence sont identifiées par des représentants canoniques et qui soit muni de trois opérations :

- MAKE-SET : étant donné un élément $x$ de l'ensemble, MAKE-SET$(x)$ renvoie la classe d'équivalence réduite à $x$ ;
- UNION : étant donné deux éléments $x$ et $y$, UNION$(x, y)$ renvoie la classe d'équivalence résultant de la fusion de la classe d'équivalence de $x$ et de celle de $y$. N'importe quel élément de ces deux classes peut être pris comme représentant canonique ;
- FIND-SET : étant donné un élément $x$, FIND-SET$(x)$ renvoie un pointeur vers le représentant de la classe de $x$.

1. Montrer qu'on peut représenter chaque classe d'équivalence par une liste chaînée intelligemment de manière à ce que MAKE-SET$(x)$ et FIND-SET$(x)$ soient en $O(1)$ et UNION$(x, y)$ soit en $O(n)$, où $n$ est le nombre d'éléments de l'ensemble.
2. On choisit de représenter les ensembles par des arbres plutôt que par des listes. Chaque ensemble est représenté par un arbre, dont la racine est le représentant canonique. Chaque élément ne pointe que sur son parent. La racine pointe sur elle-même (c'est un détail qui facilite l'écriture des algorithmes).

   Donner les complexités des opérations pour cette représentation.
3. On utilise une heuristique, l'*union-by-rank*. On associe à chaque nœud $x$ du graphe un entier $\texttt{rk}(x)$ qui est une borne supérieure de la hauteur de l'arbre enraciné en ce nœud. Écrire les trois opérations.
4. Montrer que le rang d'un nœud est strictement inférieur à celui de son père (sauf si $\texttt{p}(x) = x$). De plus, le rang d'un nœud est inférieur ou égal à $\lfloor \log(m+1) \rfloor$, où $m$ est le nombre d'opérations UNION effectuées.
5. En déduire qu'une séquence de $m$ opérations dont $n$ MAKE-SET se fait en temps $O(m \log n)$.
6. On utilise maintenant en plus une optimisation : pour chaque nœud visité par une opération FIND-SET, on le re-branche directement sur sa racine.

   Écrire les opérations. Montrer que $\texttt{rk}(\texttt{p}(x))$ ne décroît pas au cours de l'exécution de ces opérations.

Pour tout $k \geqslant 0$, on définit la *k-ième fonction d'Ackermann* $A_k$, par pour tout $j \geqslant 1$,

$$A_k(j) = \begin{cases} j+1 & \text{si } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{si } k \geqslant 1. \end{cases}$$

où

$$A_k^{(0)}(j) = j$$
$$A_k^{(i+1)}(j) = A_k(A_k^{(i)}(j))$$

7. Montrer que pour tout $j \geqslant 1$, $A_1(j) = 2j + 1$ et $A_2(j) = 2^{j+1}(j+1) - 1$.

On définit l'inverse $\alpha$ de la fonction d'Ackermann par :

$$\forall n \geqslant 0, \alpha(n) = \min\{k \mid A_k(1) \geqslant n\}.$$

On a : $\forall n \leqslant 16^{512}, \alpha(n) \leqslant 4$.
On pose, par ailleurs

$$\texttt{lvl}(x) = \max\{k \in \mathbf{N} \mid \texttt{rk}(\texttt{p}(x)) \geqslant A_k(\texttt{rk}(x))\}.$$

**8.** Montrer que

$$0 \leqslant \texttt{lvl}(x) < \alpha(n).$$

où $n$ est le nombre d'élements de l'ensemble.
Enfin, on pose, pour $x$ de rang supérieur ou égal à 1,

$$\texttt{iter}(x) = \max\{i \in \mathbf{N} \mid \texttt{rk}(\texttt{p}(x)) \geqslant A^{(i)}_{\texttt{lvl}(x)}(\texttt{rk}(x))\}.$$

**9.** Montrer que

$$1 \leqslant \texttt{iter}(x) \leqslant \texttt{rk}(x).$$

On définit le potentiel d'un nœud $x$ après la suite d'opérations $q$ par :

$$\phi_q(x) = \begin{cases} \alpha(n) \cdot \texttt{rk}(x) & \text{si } x \text{ est une racine ou } \texttt{rk}(x) = 0 \\ (\alpha(n) - \texttt{lvl}(x)) \cdot \texttt{rk}(x) - \texttt{iter}(x) & \text{si } x \text{ n'est pas une racine et } \texttt{rk}(x) \geqslant 1. \end{cases}$$

**10.** Montrer que pour tout nœud $x$ et liste d'opérations $q$, on a :

$$0 \leqslant \phi_q(x) \leqslant \alpha(n) \cdot \texttt{rk}(x).$$

**11.** Soit $x$ un nœud qui n'est pas racine et $q$ une suite d'opération, A une opération soit LINK soit FIND-SET. Montrer que $\phi_{q\text{A}}(x) \leqslant \phi_q(x)$. De plus, si $\texttt{rk}(x) \geqslant 1$ et que $\texttt{lvl}(x)$ ou $\texttt{iter}(x)$ est modifié, $\phi_{q\text{A}}(x) \leqslant \phi_q(x) - 1$.

On suppose désormais qu'on travaille sur une séquence de $m$ opérations dont $n$ MAKE-SET. On appelle *coût amorti* d'une opération son coût plus l'augmentation (ou la diminution) de potentiel qui s'en suit. En particulier, puisque le potentiel reste toujours positif, cela donne une borne supérieure du coût moyen d'une opération.

**12.** Montrer que le coût amorti de chaque MAKE-SET est $O(1)$.

**13.** Montrer que le coût amorti de chaque LINK est $O(\alpha(n))$.

**14.** Montrer que le coût amorti de chaque FIND-SET est $O(\alpha(n))$.

**15.** Conclure.