# DM9

L'algorithme *Union-Find* a pour but de gérer dynamiquement une partition d'un ensemble $\{1, 2, \dots, n\}$ fixé. Initialement, on part de la partition maximale $\{\{1\}, \{2\}, \dots, \{n\}\}$. En pratique, on représente les partitions de $\{1, \dots, n\}$ comme suit :

- chaque entier $k$ possède un (seul) parent $p_k \leqslant n$, ce que l'on notera $k \to p_k$ ; on peut avoir $k = p_k$ ;
- chaque entier $k$ possède un poids $w_k \in \mathbb{N}$ ;
- deux entiers distincts $k$ et $\ell$ ne peuvent être ancêtres l'un de l'autre, et appartiennent au même sous-ensemble si et seulement s'ils ont un ancêtre commun ;
- si l'entier $k$ appartient à un singleton, $w_k = 0$.

**Question 1.** On dit que $m$ est le *chef* de $k$ si $m$ est un ancêtre de $k$ tel que $m = p_m$. Démontrer que tout entier a un unique chef et que deux entiers $k$ et $\ell$ appartiennent au même sous-ensemble si et seulement s'ils ont le même chef.

L'algorithme est censé gérer trois types de requêtes, en procédant comme suit :

1. La requête auxiliaire Chef$(k)$ : on recherche le chef de $k$. Si $k = p_k$, il s'agit de $k$ lui-même. Sinon, il s'agit du chef de $p_k$, et ce chef devient le nouveau parent de $k$.
2. La requête Test$(k, \ell)$ : on se demande si $k$ et $\ell$ appartiennent au même sous-ensemble. Cela revient à identifier les entiers $k' = \text{Chef}(k)$ et $\ell' = \text{Chef}(\ell)$ puis à vérifier si $k' = \ell'$.
3. La requête Union$(k, \ell)$ : on souhaite réunir les sous-ensembles auxquels appartiennent $k$ et $\ell$. Cela revient à identifier les entiers $k' = \text{Chef}(k)$ et $\ell' = \text{Chef}(\ell)$, puis :
   - si $w_{k'} > w_{\ell'}$, l'entier $k'$ devient le nouveau parent de $\ell'$ ;
   - si $w_{\ell'} > w_{k'}$, l'entier $\ell'$ devient le nouveau parent de $k'$ ;
   - si $k' \neq \ell'$ et $w_{k'} = w_{\ell'}$, l'entier $k'$ devient le nouveau parent de $\ell'$ et son poids augmente de 1 .

On souhaite démontrer que répondre à $m$ de ces requêtes peut se faire en temps $\mathcal{O}\left(m \log^\star(m)\right)$, où $\log^\star$ est la fonction définie par $\log^\star(m) = 1$ lorsque $m \leqslant 1$ et $\log^\star(m) = 1 + \log^\star\left(\log_2(m)\right)$ lorsque $m > 1$.

**Question 2.** En partant de l'ensemble $\{\{1\}, \{2\}, \{3\}, \{4\}\}$, on effectue successivement les requêtes Union $(1, 2)$, Union $(3, 4)$, Union $(2, 4)$ et Test $(2, 4)$. Indiquer le parent et le poids de chaque entier $k \leqslant 4$.

**Question 3.** On note $w_k^\infty$ le poids d'un entier $k$ à la fin de l'algorithme. Démontrer que $w_u^\infty < w_v^\infty$ pour tous les entiers $u \neq v$ tels que $v$ a été le parent de $u$.

**Question 4.** Démontrer, pour tout entier $\ell \geqslant 1$, qu'il y a au plus $m/2^{\ell-1}$ entiers $k \leqslant n$ pour lesquels $w_k^\infty = \ell$.

**Question 5.** Démontrer que toute requête Test ou Union effectuée au cours de l'algorithme a une complexité majorée par $\mathcal{O}\left(\log_2(\min\{m, n\})\right)$.

**Question 6.** Soit $\mathcal{G} = (V, E)$ le graphe dont les sommets sont les entiers de 1 à $n$ et les arêtes sont les paires $(u, v)$ pour lesquelles $u \neq v$ et $v$ a été le parent de $u$ au cours de l'algorithme, mais pas quand celui-ci se termine. Démontrer que la complexité totale de nos $m$ requêtes est majorée par $\mathcal{O}(m + |E|)$.

**Question 7.** Soit $(a_\ell)_{\ell \geqslant 0}$ la suite définie par $a_0 = 0$ et $a_{\ell+1} = 2^{a_\ell}$. Pour tout entier $\ell \geqslant 0$, on note $V_\ell$ l'ensemble des entiers $k$ tels que $a_\ell \leqslant w_k^\infty < a_{\ell+1}$. Démontrer que $\mathcal{G}$ contient au plus $4m$ arêtes reliant deux sommets de $V_\ell$, et au plus $2m$ arêtes allant d'un sommet de $V_\ell$ à un sommet en dehors de $V_\ell$.

**Question 8.** Que conclure sur la complexité de l'algorithme *Union-Find* ?