Université Savoie Mont Blanc,
UFR SceM
INFO601_CMI, L3 INFO

November 27, 2025

TD 3, Structures de données pour ensembles disjoints (Union-Find)

Petites relations utiles :

- $\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$,
- $\sum_{i=1}^{n} 2^i = 2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$,
- $2^k = n$ si et seulement si $k = \log_2 n$
- si $k = \log_2 n$, $n + \frac{n}{2} + \frac{n}{2^2} + \cdots + \frac{n}{2^k} = n\left(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k}\right) \leq 2n$.

Exercice 1. Rang maximum d'un ensemble à 8 éléments

Essayez de déterminer un ensemble à 8 éléments et une séquence d'union qui maximise le rang d'un élément. Quel est le rang maximum que l'on peut atteindre ?

Exercice 2. Hauteur des arbres dans la forêt

Peut-on dire que la hauteur des arbres est toujours limité par $\alpha(n)$? Si non, donnez un majorant.

Exercice 3. Rang maximum

Peut-on dire que le rang d'un sommet de l'arbre est toujours limité par $\alpha(n)$? Si non, donnez un majorant.

Exercice 4. Forêt après 8 opérations

Soit l'ensemble $S = \{a, b, c, d, e, f, g, h\}$. Dessinez la forêt d'ensembles disjoints après:

1. Union(b, a), Union(d, c), Union(c, a),
2. Union(f, e), Union(h, g), Union(e, g),
3. Union(g, a),
4. Trouver-Ensemble(h)

Exercice 5. Pire cas appel à Trouver-Ensemble

Peut-on dire que tout appel à Trouver-Ensemble prend en pire cas $\alpha(n)$?

Exercice 6. Changements des profondeurs par Trouver-Ensemble

Si jamais un appel à Trouver-Ensemble est fait sur un sommet de profondeur k, combien de sommets voient leur profondeur réduit à 2 ?

Si aucun Trouver-Ensemble n'a été appelé avant, montrez que la somme des profondeurs de l'arbre diminue d'au moins $2^{k-2} - 1$.

Si il y a eu des Trouver-Ensemble avant, trouvez cette borne inférieure.

Exercice 7. Complexité du calcul de l'arbre couvrant de poids minimal

Soit $G = (V, E)$ un graphe avec n sommets et e arêtes, avec un poids w_i sur chaque arête e_i. Trouvez la complexité de l'algorithme de Kruskal, qui calcule un arbre couvrant minimal.

```
Kruskal(G) :
1   A := {}
2   pour chaque sommet v de G :
3      CréerEnsemble(v)
4   trier les arêtes de G par poids croissant
5   pour chaque arête (u, v) de G prise par poids croissant :
6      si Trouver-Ensemble(u) != Trouver-Ensemble(v) :
7         ajouter l'arête (u, v) à l'ensemble A
8         Union(u, v)
9   renvoyer A
```