Classes

A universe U is often divided into classes $C_1 \ldots C_k$ ($k \geq 1$) such that

each $x \in U$ belongs to C_i for exactly one $i \in 1..k$

In other words:

- $C_1 \cup \ldots \cup C_k = U$
- $C_i \cap C_j = \emptyset$ for $i \neq j$

Example:

- connected graph components (Kruskal)

Goal: representation that allows us to

- test if two elements are in the same class
- merge two classes

Representatives

Approach: for each class assign a representative

two elements are in the same class
iff they have the same representative

We need operations that

- given an element finds its representative
- given two elements form the union of their classes (by assigning them the same representative)

A union-find structure implements those operations.

Naive Implementation of Union-Find

Let a table associate each element with its representative

- Find will run in constant time, but
- Union could take time linear in number of nodes

We shall aim at something better!

Tree Implementation of Union-Find

Maintain a forest with a tree for each class

- the nodes are the class members
- the root is the representative
- edges point from child to parent

To implement Find:

just follow pointers

To take the Union of two representatives:

let one be the child of the other

Then Find runs in time $O(h)$ with h the height of a tree

- h can be linear in the number of nodes
- to prevent this, we restrict the freedom of Union

Union-by-Rank

Union-by-Rank lets Union be guided by rank (aka height):

make root of shorter tree a child of other root

Then, with H the height and N the node count,

$$H(T) \leq \lg(N(T)) \text{ for all trees } T$$

as can be proved by induction:

- if T has one node: $0 \leq \lg(1)$
- if T is formed by the union of T_1 and T_2: inductively we have

$$\begin{aligned} H(T_1) &\leq \lg(N(T_1)) \\ H(T_2) &\leq \lg(N(T_2)) \end{aligned}$$

  and we can do a case analysis:

  - if $H(T_1) > H(T_2)$ then $H(T) = H(T_1)$
  - if $H(T_1) < H(T_2)$ then $H(T) = H(T_2)$
  - if $H(T_1) = H(T_2)$ then $H(T) = H(T_1) + 1$

Hence all operations run in time logarithmic in the number of nodes.

Union-by-Rank with Path Compression

When traversing edges during Find

let each node point directly to root

which will result in a "flatter" tree

- it's not feasible to maintain the exact height
- instead we keep an upper estimate, called the rank

The cost of an operation

- may still be proportional to the logarithm of the number of nodes
- but that happens very rarely

Running m operations on tree with n nodes takes time in

$$O(m\,\alpha(n))$$

with α a function that grows extremely slowly.