Union Find data structure

$m$ finds and $n$ union can be done in $O(m + n \log n)$ time
using an array based data structure

Tree based representation of sets

```
S1
  \  /
 x1 x2
  \/
 x3
```

```
S2
  \  /
 x4 x10
  \/
 x0
```

Star-like structure

```
\  /
 x
```

Find ($x$) reports the label $l$ which $x$ is pointing. $O(1)$ time

Union: We can link the root of one tree to the other or create a new root and link both roots to this
Consequence: no longer stars
Distance to root increases
Find becomes more expensive
in the distance to the root

How do we have any bound on
the distance of an element to the
root?

We will maintain some kind of
depth/rank information with every
tree. The rank function is defined
as 0 for a single ton element
when we union two trees with
ranks \( r_1 \) and \( r_2 \) where \( r_1 < r_2 \),
then we make the root with rank
\( r_1 \), point to the root with rank \( r_2 \)
else if \( r_1 = r_2 \) we can choose
to link the root arbitrarily.
Then rank increases by 1.

- rank = 0

\[ n_1 < n_2 \]

\[ n_1 = n_2 \]

Union by rank heuristic

Claim: The minimum no. of nodes in a tree with rank \( r \geq 2^r \)

\[ \Rightarrow \text{ The max rank } \leq \log_2 n \]

\[ \Rightarrow \text{ Find will be done in } O(\log_2 n) \]

Proof (by induction on rank)

\[ \text{rank} = 0 \quad \text{by definition} \]
Suppose it is true for all ranks $< i$

$\Rightarrow$ when we union two trees, $T_1, T_2$ with ranks $l, k \quad l, k < i$

then $T_1$ has at least $2^l$ nodes

$T_2$ has at least $2^k$ nodes

Rank after union

Case 1 \quad $l < k$ \quad Total in $T$

$2^l + 2^k$ \quad and rank in $R$

Case 2 \quad $l = k$ \quad rank in $R+1$

$\geq 2^k + 2^k = 2^{k+1}$

Using these rules with union heaps the cost of $m$ finds and $n$ unions:

$O(m \log n + n)$

And every individual operation cost is bounded by $O(\log n) \quad (O(1) \text{ for array based})$
We can improve the performance of the tree-based data structure by using an heuristic called "path compression".

By path compression heuristic we will decrease the distance of many nodes to the root node.
The cost of \( m \) unions and \( n \) finds by using rank and path compression heuristic is \( \mathcal{O}\left((m+n)\log^* n\right) \).

\[ \log^* n : \log \text{ star } n \text{ function} \]

\[ \log^* 2 = 1 \]

\[ \log^* k \quad \text{where } k \leq 2 \]

\[ = \log^* i + 1 \]

The \( \log^* \) function is related to an inverse Ackermann function.