Disjoint Set

Union Find Set

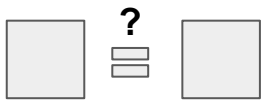
Previous Problem

Kruskal’s algorithm

Previous Problem

Kruskal’s algorithm

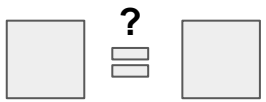
- Find out (quickly) if two nodes were in the same group

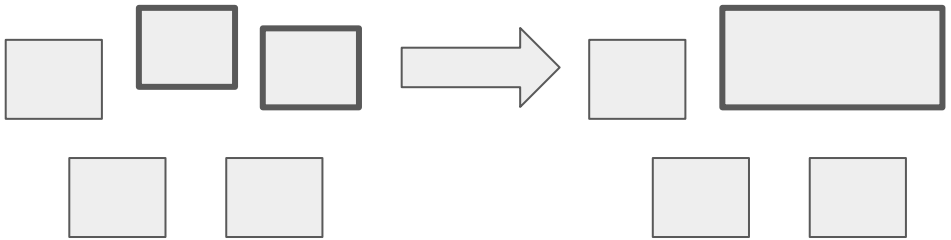


Previous Problem

Kruskal's algorithm

- Find out (quickly) if two nodes were in the same group
- Merge (quickly) two groups





Simple Idea

Let 1 node represent each the group to which they belong.

Simple Idea

Let 1 node represent each the group to which they belong.

Give nodes a way to find their representative.

Simple Idea

Let 1 node represent each the group to which they belong.

Give nodes a way to find their representative.

When merging two groups update the representative for one of the groups.

Simple Merge

```
Merge( node_a, node_b )
    Let rep_a be representative of node_a
    Let rep_b be representative of node_a
    For each Node curNode in nodeSet
        If rep_b is the representative of curNode
            Set representative of curNode to rep_a
```

Simple Merge

```
Merge( node_a, node_b )
    Let rep_a be representative of node_a
    Let rep_b be representative of node_a
    For each Node curNode in nodeSet
        If rep_b is the representative of curNode
            Set representative of curNode to rep_a
```

SLOW

“Forest” Idea

Store groups as rooted Trees

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

Find would be worst case O(N)

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

Find would be worst case O(N)

BALANCE the trees

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

Find would be worst case O(N)

BALANCE the trees

- Like the AVL

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

Find would be worst case O(N)

BALANCE the trees

- Like the AVL
- Only change the root of the shorter tree

“Forest” Idea

Store groups as rooted Trees

- Each node stores a parent
- The representative would be the root
- To merge just connect the root of one tree to the root of the other tree

Find would be worst case O(N)

BALANCE the trees

- Like the AVL
- Only change the root of the shorter tree

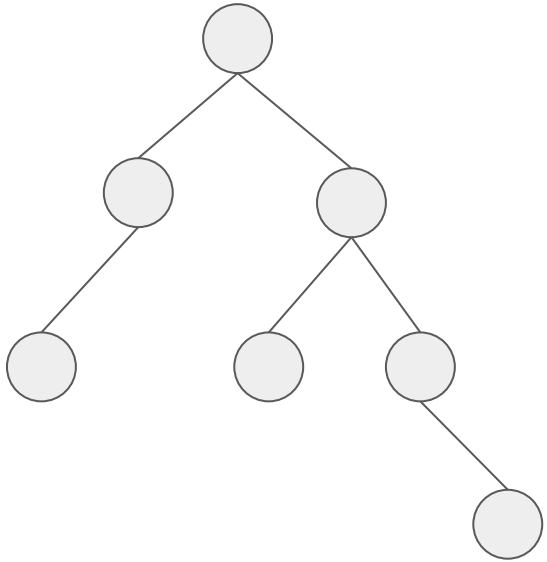
Ensures log(N) depth

Path Compression

When finding the representative (root) compress the path.

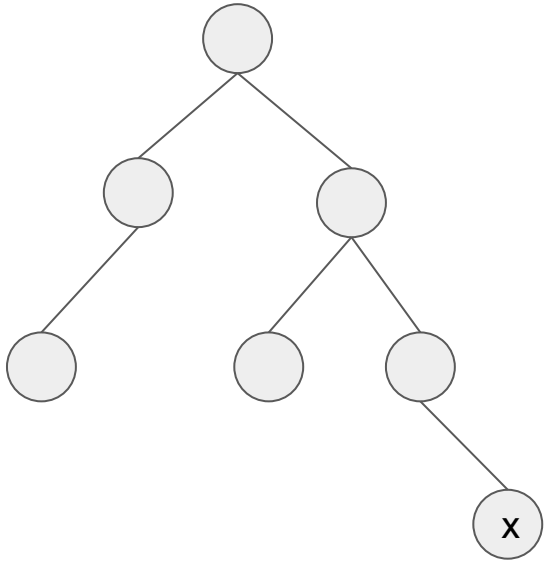
Path Compression

When finding the representative (root) compress the path.



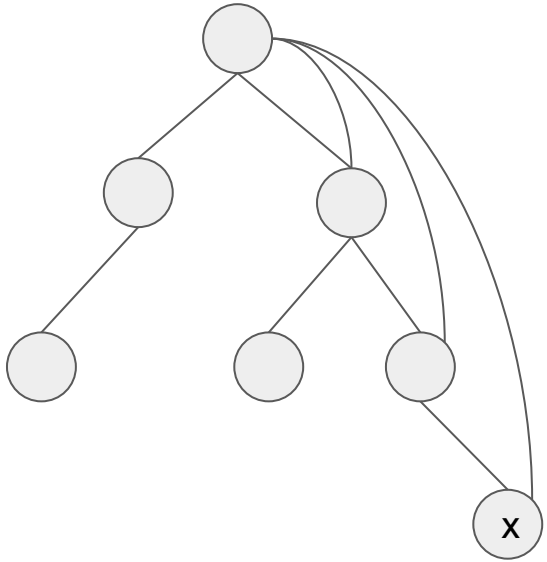
Path Compression

When finding the representative (root) compress the path.



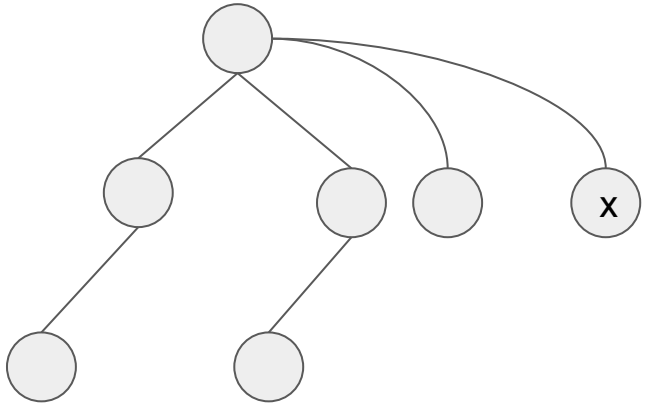
Path Compression

When finding the representative (root) compress the path.



Path Compression

When finding the representative (root) compress the path.



Path Compression

When finding the representative (root) compress the path.

Runtime becomes O(Ackerman's^{-1}(N))

Path Compression

When finding the representative (root) compress the path.

Runtime becomes O(Ackerman's^{-1}(N))

Note this is not 1/Ackerman's(N)