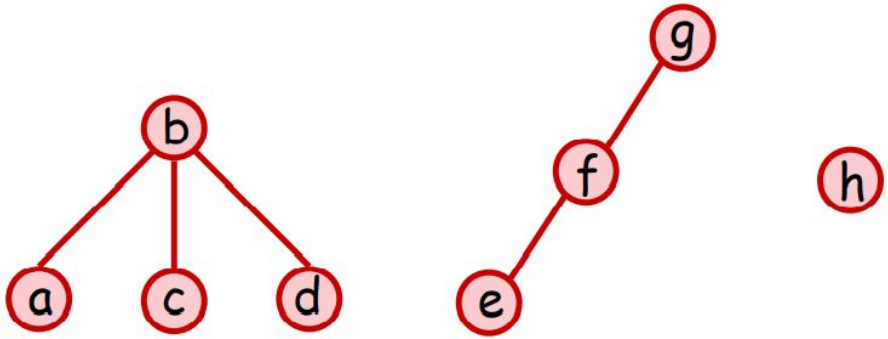
Disjoint Set

https://shorturl.at/NlJbJ

DSU

- maintain disjoint set by a forest
  - each element has its parent
  - each set is a seperate rooted tree
  - the representative of each set is the root of tree
- example: { {a, b, c, d}, {e, f, g}, {h} }



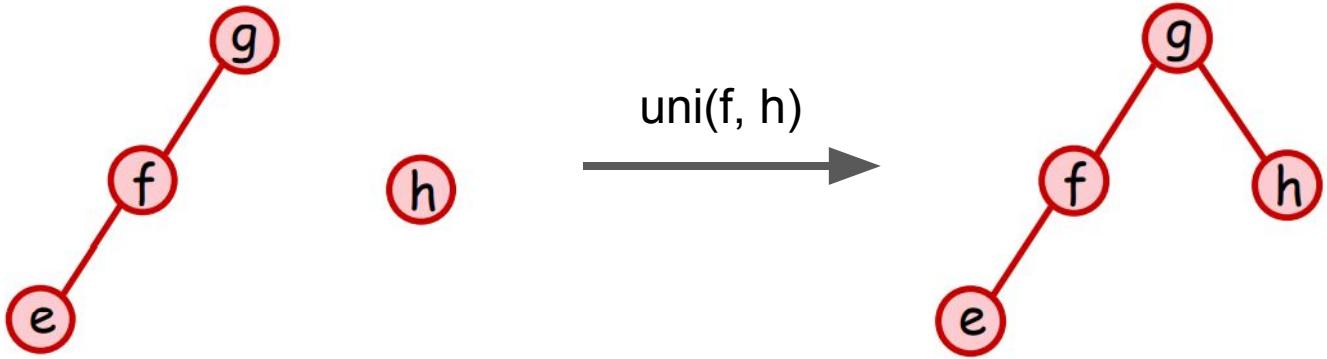
DSU- struct declaration

```
struct DisjointSet {
    int n;
    vector<int> parent, size;

    DisjointSet(int _n) : n(_n), parent(n), size(n, 1) {
        iota(parent.begin(), parent.end(), 0);
    }

    int find_root(int x);
    bool same(int x, int y);
    void uni(int x, int y);
};
```

index	0	1	2	3	4
parent	0	1	2	3	4
size	1	1	1	1	1

DSU- union()

```
void DisjointSet::uni(int x, int y) {
    parent[find_root(x)] = find_root(y);
}
```



DSU- optimization: union by size

```
void DisjointSet::uni(int x, int y) {
    parent[find_root(x)] = find_root(y);
}
```

```
void DisjointSet::uni(int x, int y) {
    int rx = find_root(x), ry = find_root(y);
    if (rx == ry) return;
    if (size[rx] > size[ry]) swap(rx, ry);
    parent[rx] = ry;
    size[ry] += size[rx];
}
```

Would the size of each element correct?

DSU- struct declaration

```
struct DisjointSet {
    int n;
    vector<int> parent, size;

    DisjointSet(int _n) : n(_n), parent(n), size(n, 1) {
        iota(parent.begin(), parent.end(), 0);
    }

    int find_root(int x);
    bool same(int x, int y);
    void uni(int x, int y);
};
```

Only the size of each rooted tree's root is correct, but it is enough.

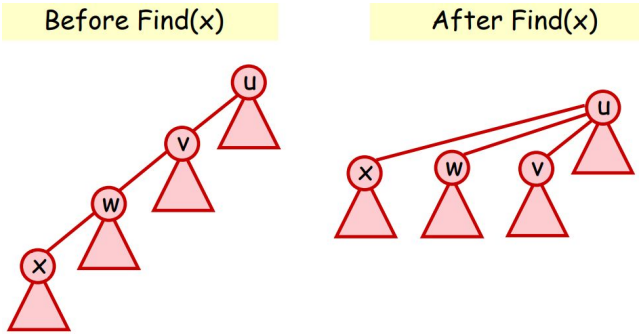
index	0	1	2	3	4
parent	0	1	2	3	4
size	1	1	1	1	1

DSU- find_root()

```
int DisjointSet::find_root(int x) {
    if (x == parent[x]) return x;
    else return find_root(parent[x]);
}
```

DSU- optimization: path compression

```
int DisjointSet::find_root(int x) {
    if (x == parent[x]) return x;
    else return find_root(parent[x]);
}
```




```
int DisjointSet::find_root(int x) {
    if (x == parent[x]) return x;
    else return parent[x] = find_root(parent[x]);
}
```

DSU- same()

```
bool DisjointSet::same(int x, int y) {
    return find_root(x) == find_root(y);
}
```

Time Complexity

- m operations: $\Theta(m\,\alpha(n))$ (with both)
- m operations: $\Theta(m \log n)$ (with Union by size Only) (why?)
- m operations: $\Theta(m \log n)$ (with Path Compression Only)

Sample Code:

Path Compression + Union By Size:
https://pastebin.com/ff2yg6mM

What Disjoint Set can do?

- Use Disjoint Set to count the number of connected components
- Use Disjoint Set to check whether a graph contains cycles
- Use Disjoint Set to check whether a graph is bipartite
- Use Disjoint Set to find bridge
- Some offline RMQ problem
- …