[Disjoint Set Union Masterclass]

Agenda → Introduction → DSU basics
DSU by size
DSU by Rank & Path compress
Time Complexity
Implement

Applicati → Basic
Problem →

Offline query with DSU
MST
≡

Pre requisite → Just understands

→ loops
→ Recursion
→ Introductory graphs

Disjoint Set Union

Cluster based problem → You will be having some elements & you need to add them to different clusters / groups. & sometimes you might need to get that which group any element belongs to.

Terminology

↳ To uniquely denote a group we will pick any one element of the group & name it as the leader / parent of the group.

↳ Union Operation → (a, b) → adds b to the group of a. or vice-a-versa

↳ Act/find operation (x) → to what group/cluster/set the element x belongs → relative parent of the group

How to implement Get and union ?? (n elements)

⇒ Arrays

group

A

0	1	2	3	4	5	6
6	6	6	6	6	6	6

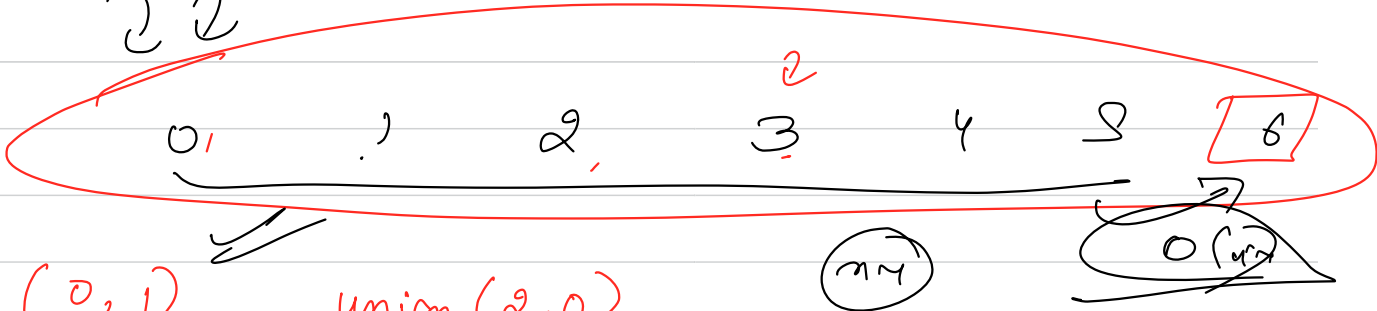
2

0, 1, 2, 3, 4, 5, 6 → elements

→ groups

→ found

They are all belongs to new one group of 2



get(3)

A[3]

union (0, 1)

union (2, 3)

union (2, 4)

union (2, 5)

union (2, 0)

↳ add 0 to the group of 2

→ union (6, 3) == union (6, 2)

```
int Get(int x) {                    // O(1)
    return p[x];
}

void union(a, b) {                  // O(n)
    a = Get(a);                     // for one query
    b = Get(b);
    for i in 1....n {
        if p[i] == b
            p[i] = a
    }
}
```

// we can store groups as group of vectors LL

Complexity

Ref

0	1	2	3	4	5	6	→ elements
0	5	2	3	4	5	6	→ parent

O(n) ~~XX~~

unia(3,4)
unia(3,5)
↳ union(2,3)

LL →

LL →

2→3→4⇄5

indegree

merge 2 linked list

get(4)

Singly

Doubly LL

get → O(n)
union → O(n)
} → using LL

We can update the parent array while merging 2 LL, to optimize get

```
get(x) {                                  O(1)
    return parent[x];
}
union(a, b) {                             O(n)
    a = get(a)
    b = get(b)
    for el in LL(b):
        parent[el] = a
    merge(LL(a), LL(b))
}
```

So what we can see if we add n elements then the operation of updating parent array for each element is $O(n)$.

So amortized complexity $\frac{n \times n}{n} \rightarrow O(n)$ for union

We do $O(n)$ loop k times for n union

amortized $\rightarrow \frac{n \times k}{n} \rightarrow O(k)$

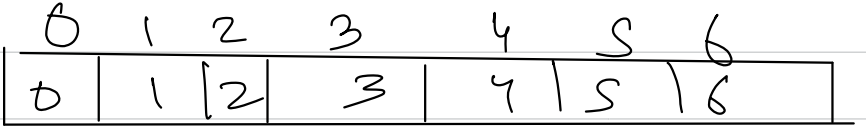
Now lets say we can do an optimization.

We will not union b to a for union(a, b)

Anytime

We will add the smaller size group to the bigger one group

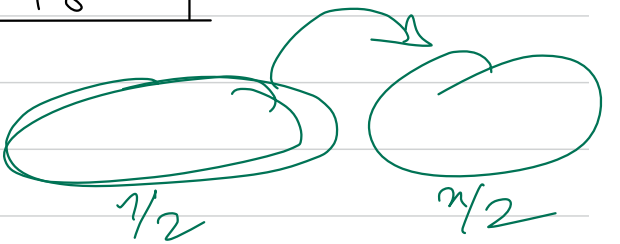
Union by size

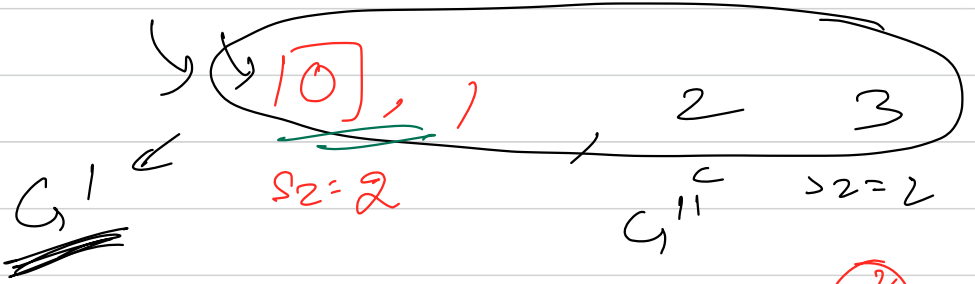


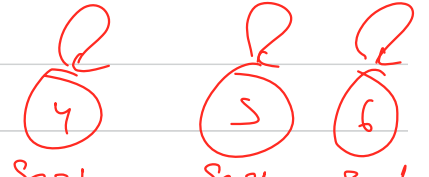
Size of group :: rank

union (0, 1)

union (0, 2)

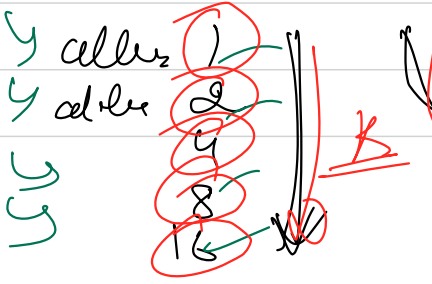


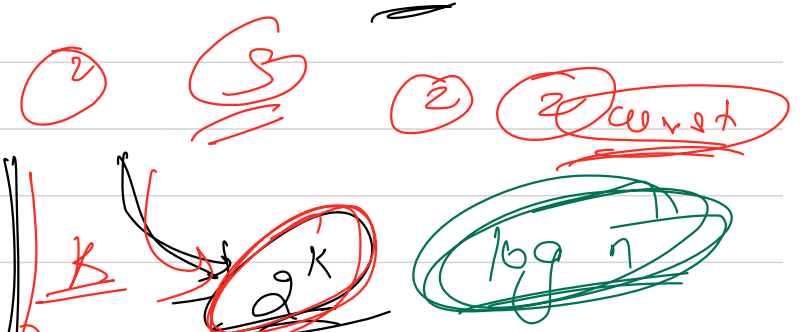




Size ≥ 4

1 2 4 8





when did you applied that loop, when you merge

Smaller into greater

another

$n \times (\log n)$

n

n unions

$\Rightarrow$ $O(\log n)$

union $(\log n)$

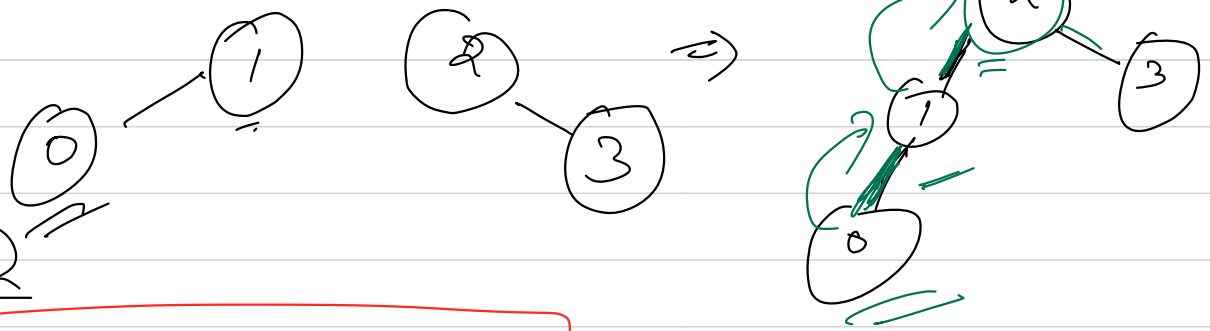
Cut

to optimize get instead of using LL how about trees

hierarchy

2

Cut



Union (4,4)

new link on new merge

How may merge we do ??

log n

no of operations you need to traverse to root of tree will be equal to no. of operation to get the fault

= No of link = No of goodby → $\log n$

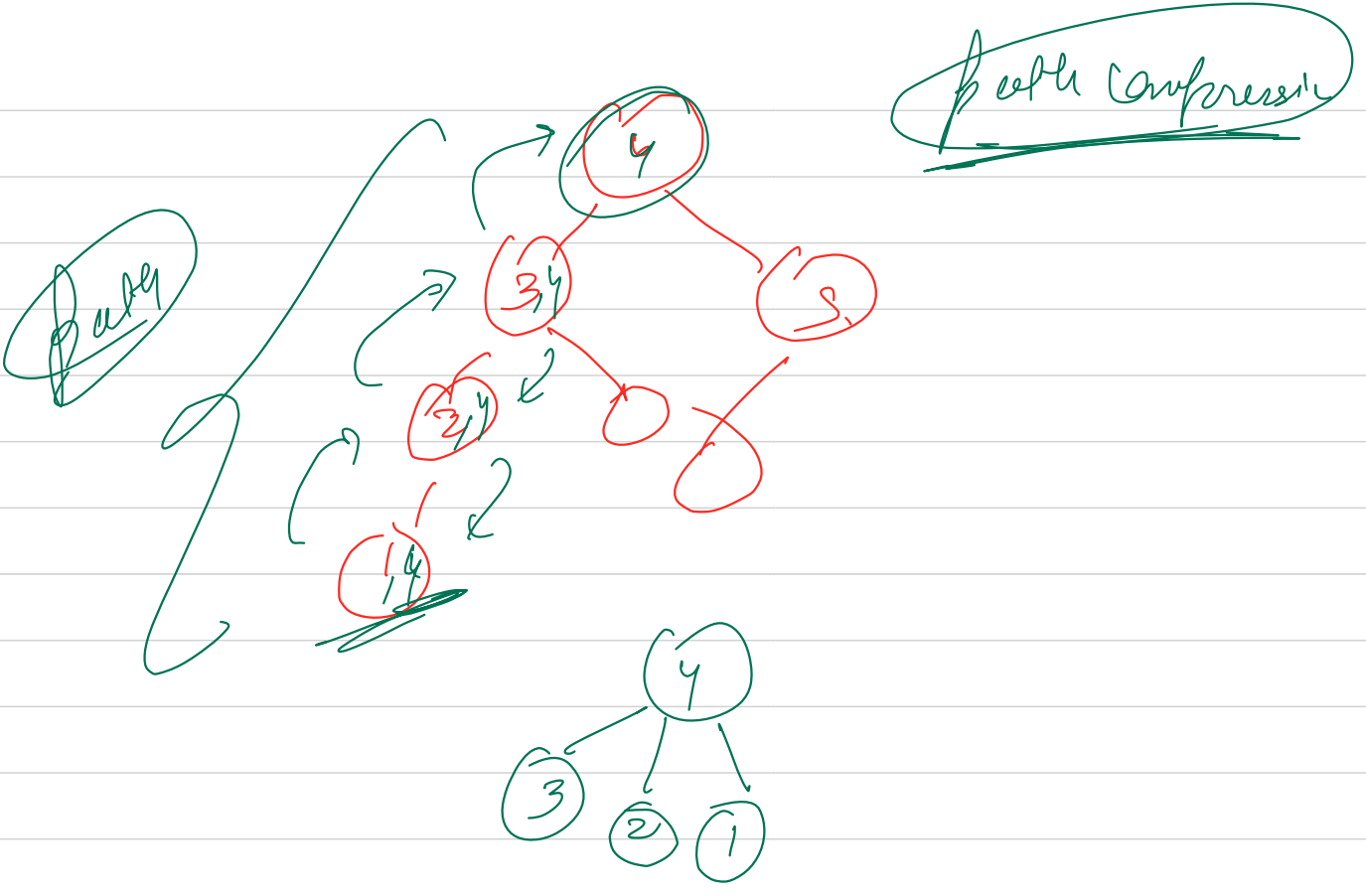
Get → $O(\log n)$

Union → $O(\log n)$

Union by rank → In union by size, size is updated at most log n times. Count of how many merges have been done

↳ # path compression (path-y-am)

recursively


Path Compression
Path
4
3
5
2
1
4
3
2
1

Time Complexity ↔ Inverse Ackermann function

Path compression

Union, get → $O(\log^* n)$

What is this $\log^* n$ → very very slow growth function

↳ how many times we should take binary log of n to get a value smaller than one.

Take a big value

$\log^*(2^{65536})$ → In how many steps we can reduce 2^{65536} to less than 1 using binary log ($\log_2$)

$(\log_2 2^{65536}) \rightarrow (65536 \rightarrow \log_2 2^{16}) = (16 \rightarrow \log_2 2^4)$

$\rightarrow (4 = \log_2 2^2) \rightarrow 2 \rightarrow 1 \rightarrow 0$

6 steps

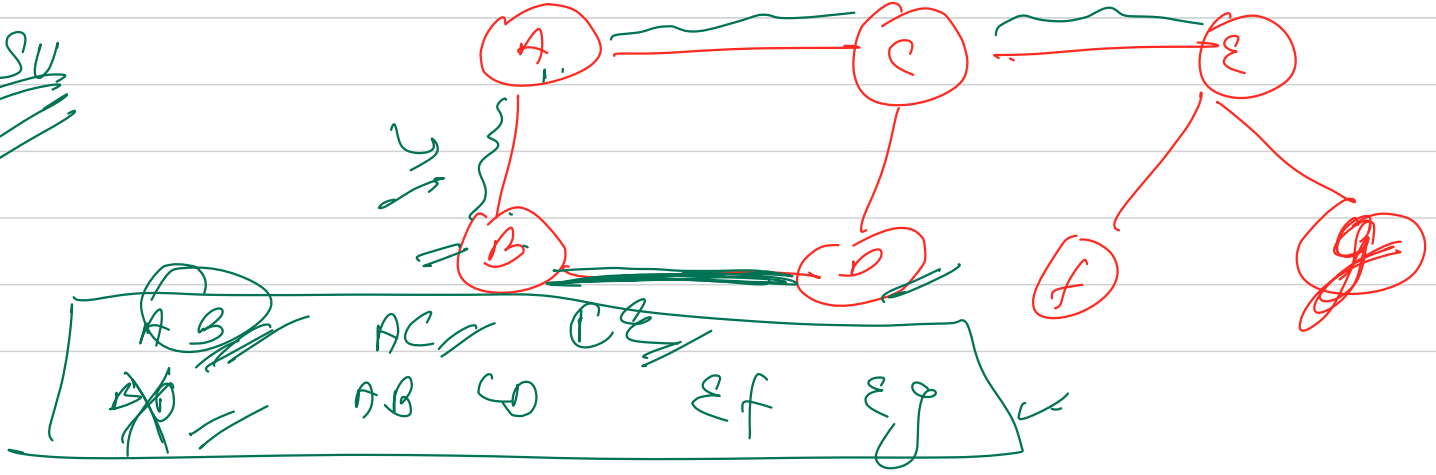
Graph algorithms

- Cycle finding
- Minimum Spanning Trees (Kruskals)
- Range Query
- connected components
- Bipartition
- ⋮

Tree is a special case of graph where the graph is acyclic

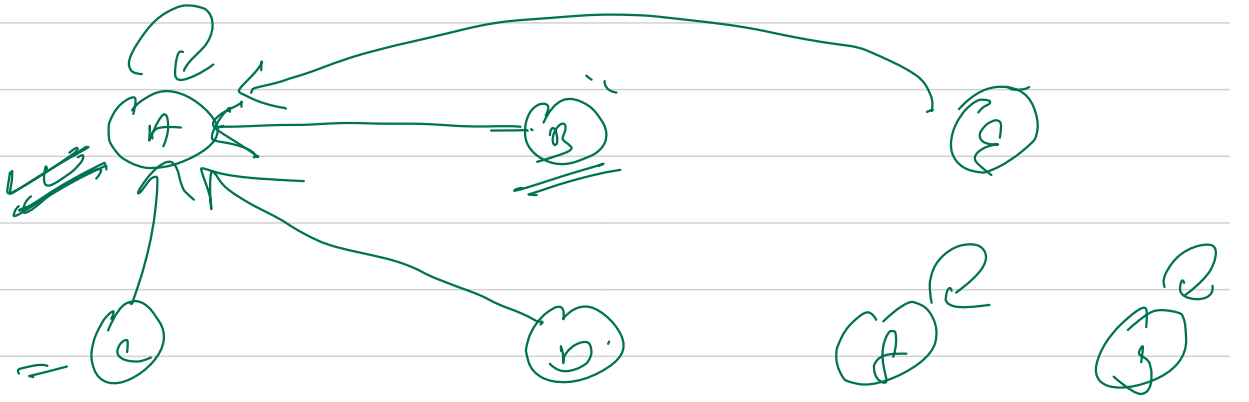
↳ Given a graph, you need to find if it has cycle or not

DSU



Union of vertices by edges

cycle ka test



union (A, B)

union (A, C)

union (C, E)

union (B, D)

union (A, B)

→ parent (A) → A → if 2 vertices
→ parent (B) → A belong to same
group then we have a cycle