Agenda

→ Refresher & solve basic DSU problem

→ Imp application of DSU → Minimum Spanning Tree in graph

→ Cool trick → calculation of bipartiteness using DSU

→ Problem solving → Offline query with DSU.

Q → You got n elements. You have 2 type of queries — 2

a) Type 1 → (x, y) → merge them in same group

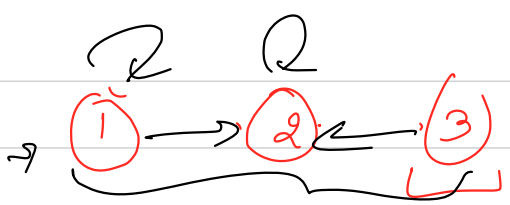
b) Type 2 → (x) → return the max, min, total elements of the group

$n \leq 10^5$ $Q \leq 10^5$

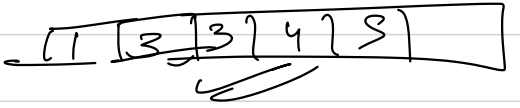
parent / rank

→ (find) → to which group x belongs to

Union By Rank



normal array



Type 1 (1, 2)

	min	max	total
Type 2 (3) →	3	3	1

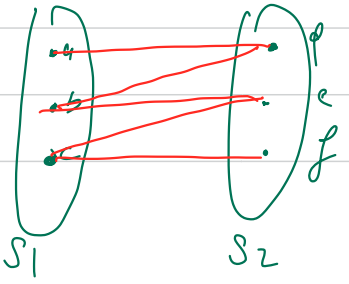
Type 1 (2, 3)

Type 2 (3) →	1	3	3

max[parent] ⇐ max(max[1], max[2])

* Bipartite graph →

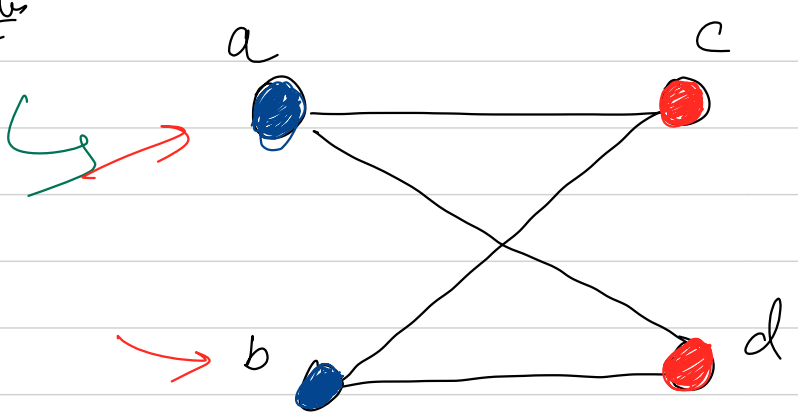
You have a graph G. If we can split the vertices of graph G into 2 sets S_1 and S_2 such that there is no edge between two vertices from the same set; then the graph is Bipartite



Q → 10 b

M

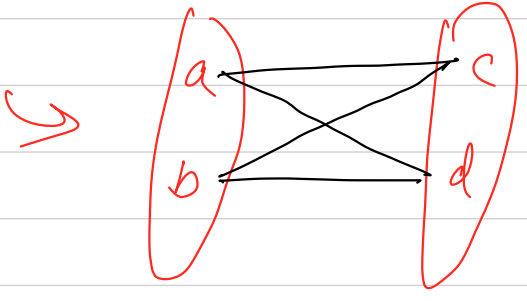
21



is this bipartite ??

This is a bipartite graph

V → { a, b, c, d }



→ Yes a bipartite

→ 2 colouring problem → Given a graph G, if you can color the vertices with 2 and only 2 colors such that no two neighbouring vertices have same color then we say a graph is 2-colourable

→ If a graph is bipartite then it is 2-colourable

Disjoint Set Union

Persalary types of query

While we are merging 2 groups, we can maintain an extra space minimal of a groups, maximal of groups & size of groups

↳ Q3 You have an undirected graph of n vertices.

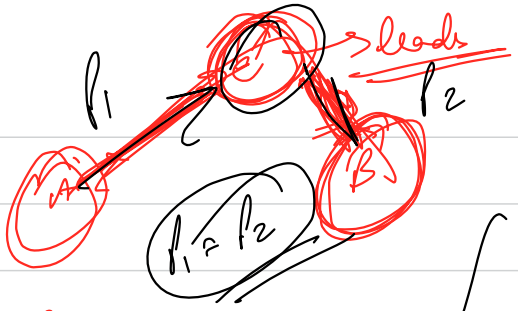
Each vertex is colored by Red or Blue.

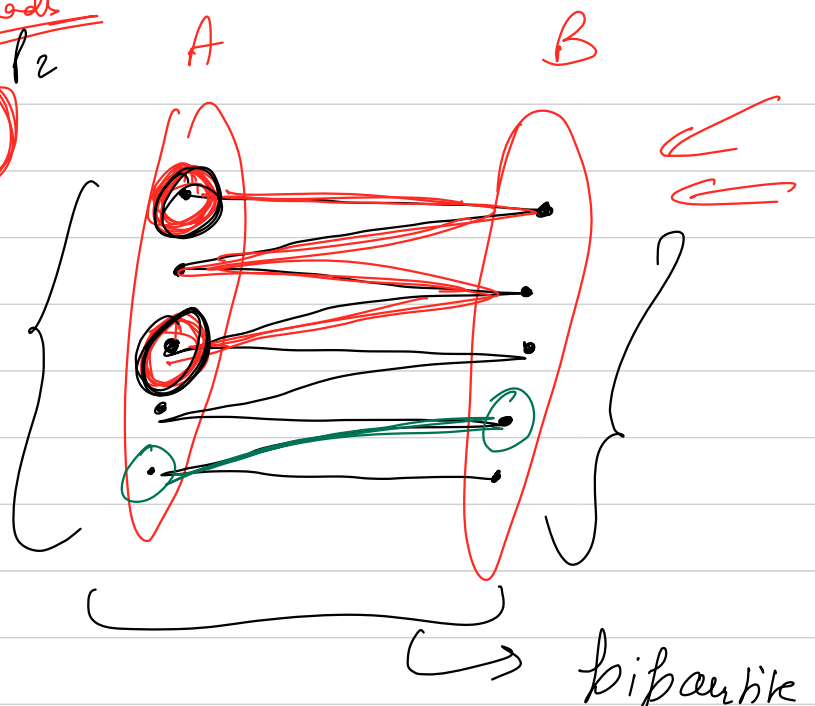
Type 1 → (x, y) → diff component → add edge between them, edge always connects 2 diff color vertex

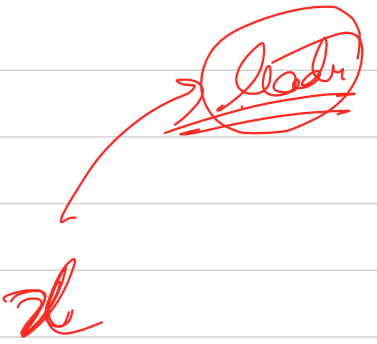
Type 2 → (x, y) → if x & y are from same component, whether they have same color or not ✓

$n \leq 10^5$

$Q \leq 10^5$







Len(a,b) = even

len(a,c) , len(c,b)

odd odd

len(a,c) len(c,b)

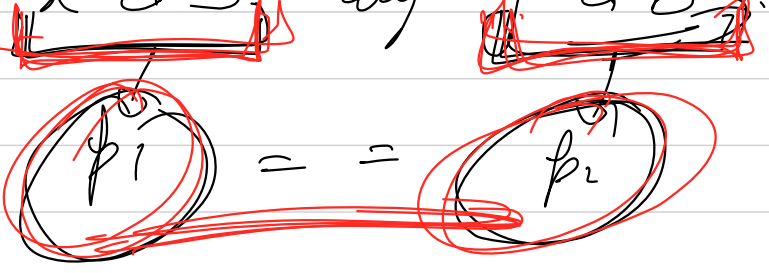
even even

bipartite combinat

if a graph is bipartite, then 2 vertices of the same set then they will be have even cycle dist btw them

Let's say b is the leader of the component in which x, y belongs.

To check bipartiteness we can check what is the parity of cycle between x & b, say p_1 and y & b, say p_2.



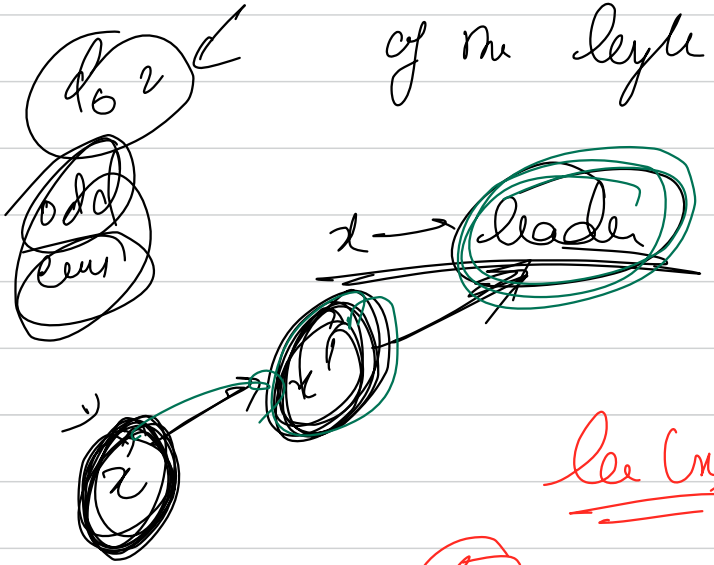
$(x, y) \rightarrow$ even

(odd + odd)

(even + even)

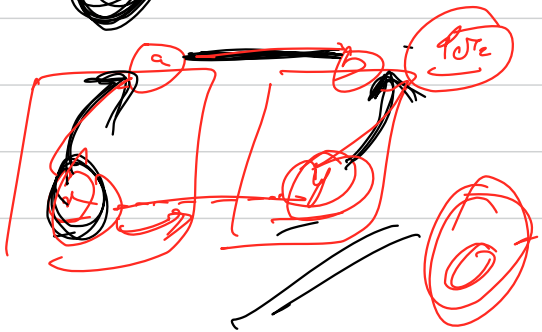
DSU

Get(x) → return leader of x group & parity of x to leader where x & leader

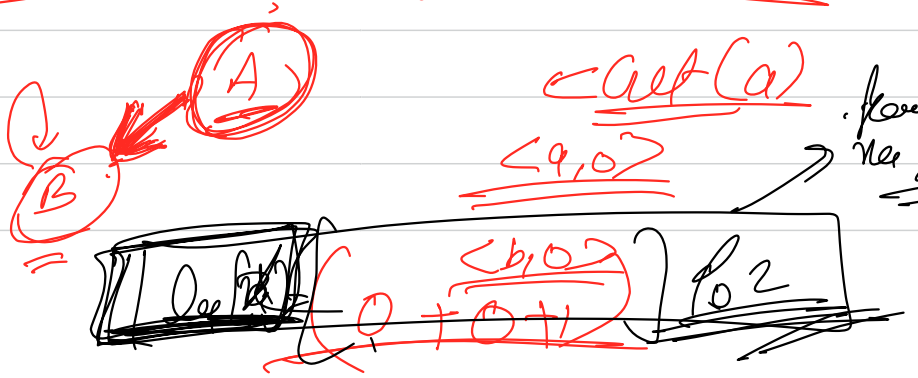


(a+b) % c

((a % c + b % c) % c

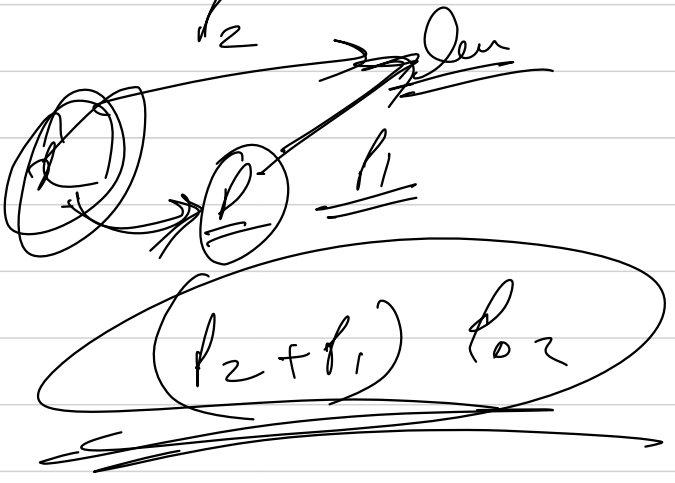


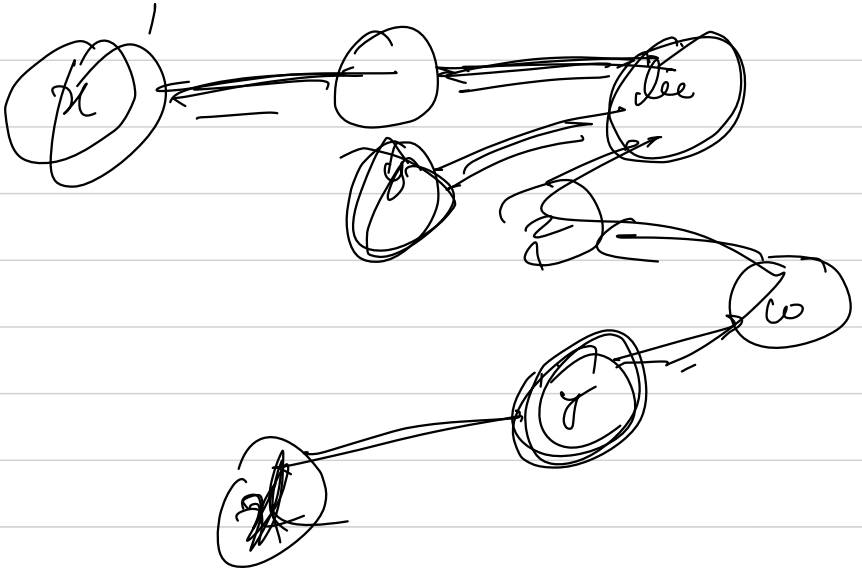
lca(x) → dist to the leader



Go and recursively calc the leader of the group & parity of your part with the leader

if the leader is odd or even





addedy → unv.

val x cut(x)
val y cut(y)

val x. party == val y. party

then

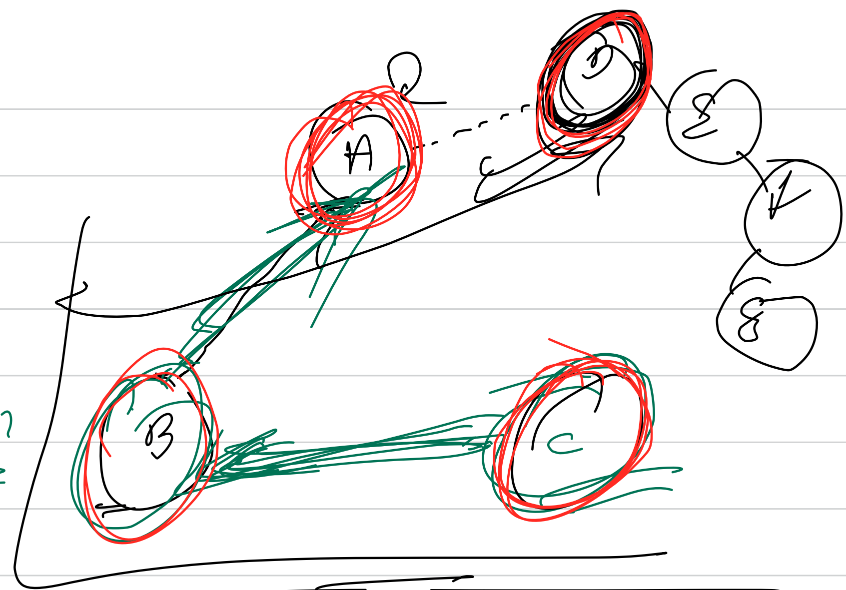
else

no

<A, 1>

<C, 0>

(0 + 1 + 1) %2



par	A	A	A
lev	0	1	0

1	1	2
1	2	3
2	1	2
2	1	3

add edge

un(1, 2)

un(2, 3)

the parent's distance will be changed, now it should be the dist to the root which is equal to the distance to the old parent plus the distance from old parent to the root

Sum of wt of edg → min

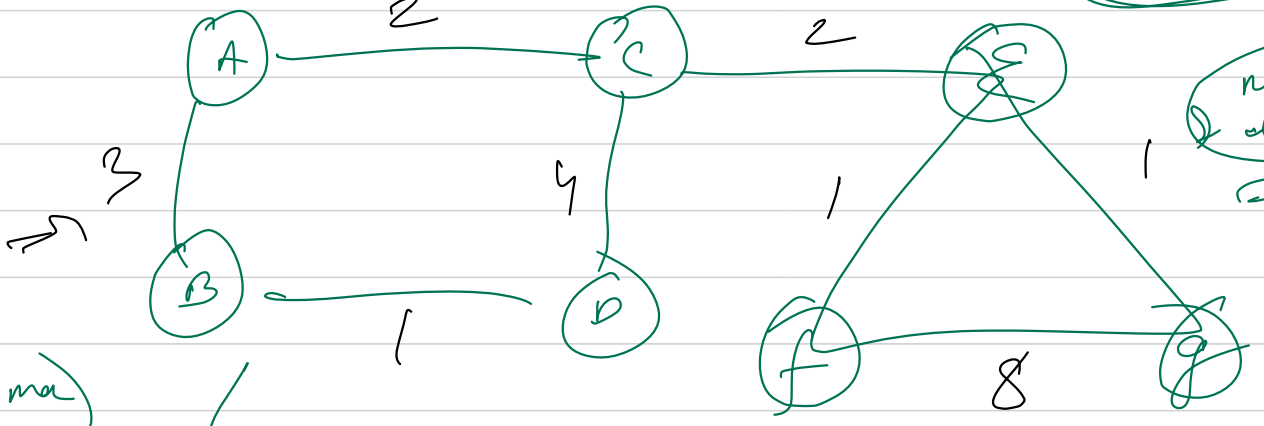
Minimum Spanning Tree

greedy

Prims

Kruskals

make BFS algorithm

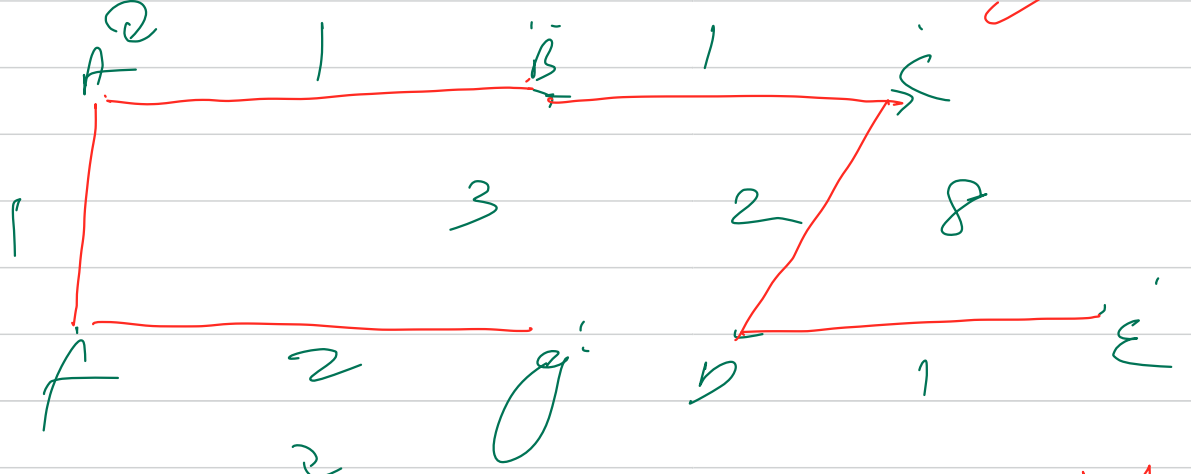


min → max

Remove cycles → one by one choose the edges of min wt. If adding that edge creates a cycle then don't add it

Kruskals → DSU

cdj cubrst



Sort the edj → wt → Af, AB, BC, DE, DC, fg, Bg, CE

SANKET 10