GRAPHS...

video-33

"let's make it easy too"

HARD

Leetcode - 1489

If you have tried my Graph Concepts & Qns playlist, these Qns, will seem very easy. Do try it once :)

Find Critical and Pseudo-Critical Edges In Minimum Spanning Tree

Companies :- Meta

Pre-requisite for this video :-

Minimum Spanning Tree | Easy Theory Full Detail | Amazon, Microsoft | Graph Concepts & Qns - 33
codestorywithMIK
9:01

~~Prim's Algorithm | Minimum Spanning Tree | Full Dry Run | INTUITION | Graph Concepts & Qns - 34~~
~~codestorywithMIK~~
31:24

~~Min Cost to Connect All Points | Prim's Algorithm | META | Graph Concepts & Qns - 35 | Leetcode-1584~~
~~codestorywithMIK~~
19:04

Kruskal's Algorithm | Minimum Spanning Tree | Full Dry Run | INTUITION | Graph Concepts & Qns - 36
codestorywithMIK
24:01

DSU

Kruskal's , DSU

Even if you don't revise them, I am going to make this easy. let's go...

1489. Find Critical and Pseudo-Critical Edges in Minimum Spanning Tree

Hard 844 71 Add to List Share

Given a weighted undirected connected graph with `n` vertices numbered from `0` to `n - 1`, and an array `edges` where `edges[i] = [a_i, b_i, weight_i]` represents a bidirectional and weighted edge between nodes `a_i` and `b_i`. A minimum spanning tree (MST) is a subset of the graph's edges that connects all vertices without cycles and with the minimum possible total edge weight.

Find *all the critical and pseudo-critical edges in the given graph's minimum spanning tree (MST)*. An MST edge whose deletion from the graph would cause the MST weight to increase is called a *critical edge*. On the other hand, a pseudo-critical edge is that which can appear in some MSTs but not all.

Note that you can return the indices of the edges in any order.

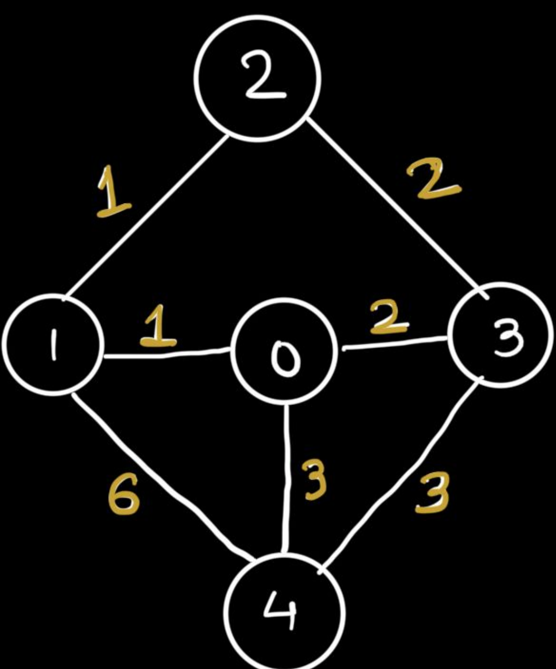
Example :- n = 5

edges = [{0,1,1}, {1,2,1}, {2,3,2}, {0,3,2}, {0,4,3}, {3,4,3}, {1,4,6}]

5 6

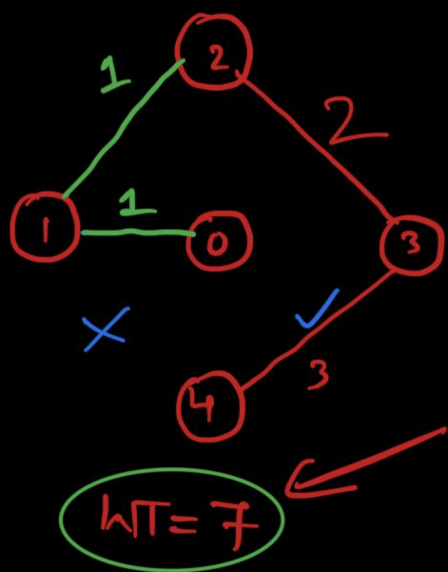
CE PC

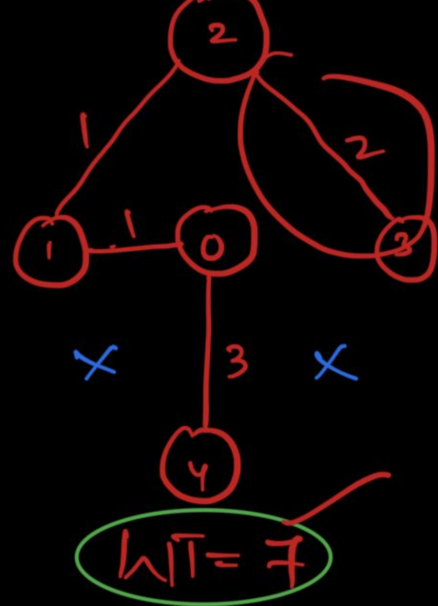
{1, 0} {2, 3, 4, 5,

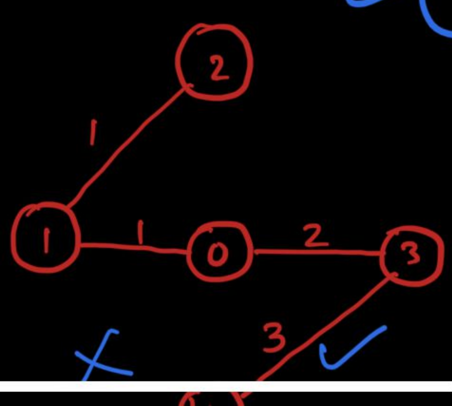


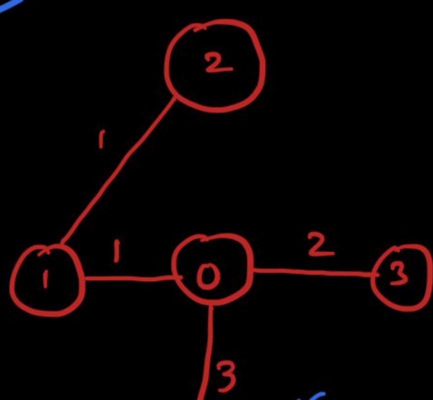
wt = 21

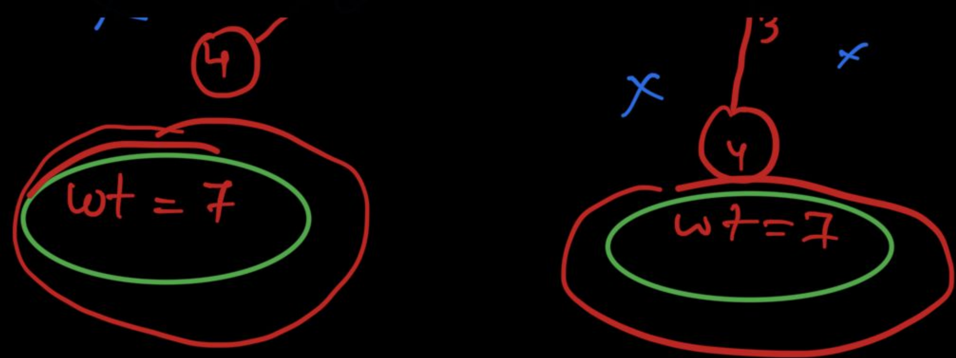
V = 5











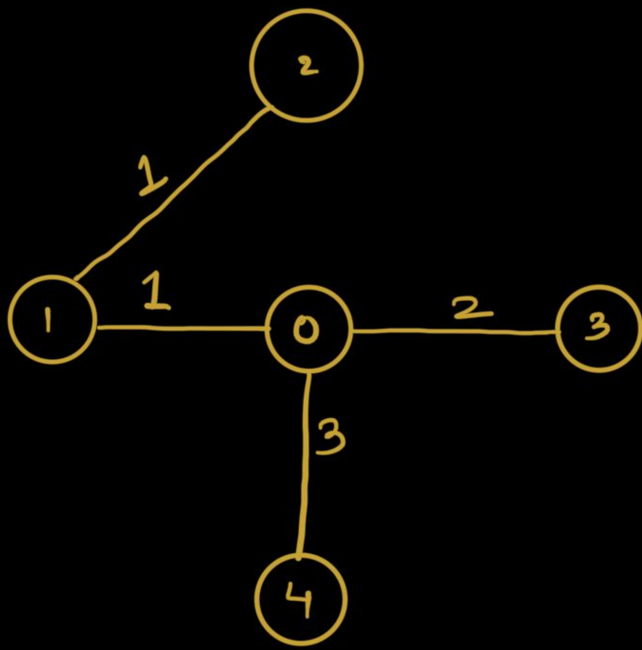
Quick Recap of Kruskal's :-

sort

Sorted (wt·) & we· added index.

0 1 2 3 4 5 6

[{0, 1, 1, 0}, {1, 2, 1, 1}, {0, 3, 2, 3}, {2, 3, 2, 2},
{0, 4, 3, 4}, {3, 4, 3, 5}, {1, 4, 6, 6}]



MST-WEIGHT
= 7

Steps :-

1. Push original index also
2. Sort based on weight
3. Find MST_WEIGHT → Kruskal's Algo
4. Check each edge → 1. Skip (Krusk).
   2. Add (Krusk).

KR
DSU

MST-WEIGHT = 7

[{0, 1, 1, 0}, {1, 2, 1, 1}, {0, 3, 2, 3}, {2, 3, 2, 2},
{0, 4, 3, 4}, {3, 4, 3, 5}, {1, 4, 6, 6}]

0 1 2 3 4 5 6

i

u = 0
v = 1
wt = 1
idx = 0

skipwt = Kruskal(skip = i);

if (skipwt > MST-WEIGHT) {
Critical.push_back(idx);
} else if

include add

== MST-wt.

Pseudo_Critical :-

Kruskal (add_edge) {

wt == MST-wt-

Pseudo_Critical :-

Kruskull (add_edge) {

wt == MST wt.

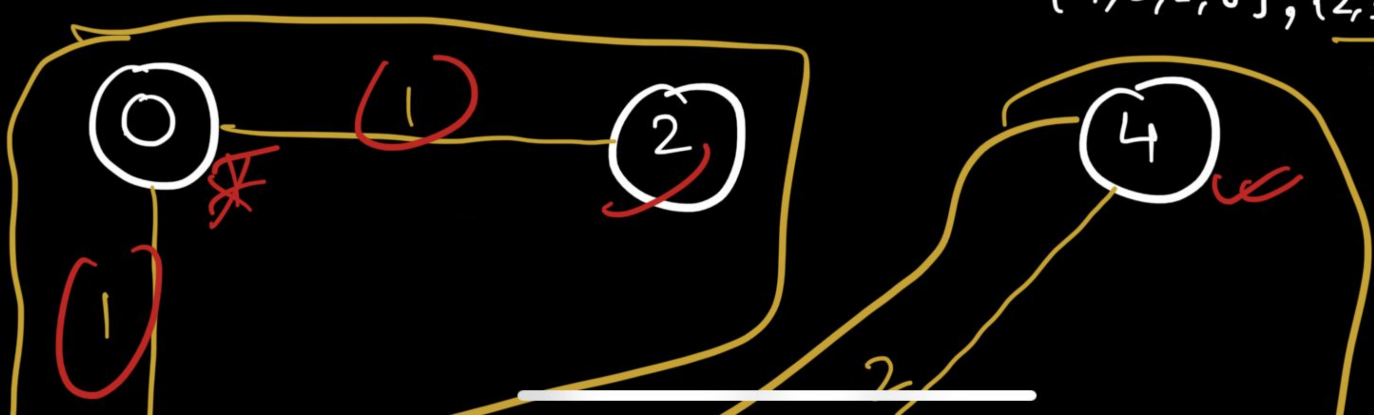
}

MST_1 MST_4

MST-WEIGHT = 10

{0,1,1,0}, {0,2,1,2}, {1,2,1,1}, {3,4,2,4}, {3,5,2,5},
{4,5,2,6}, {2,3,4,3}}

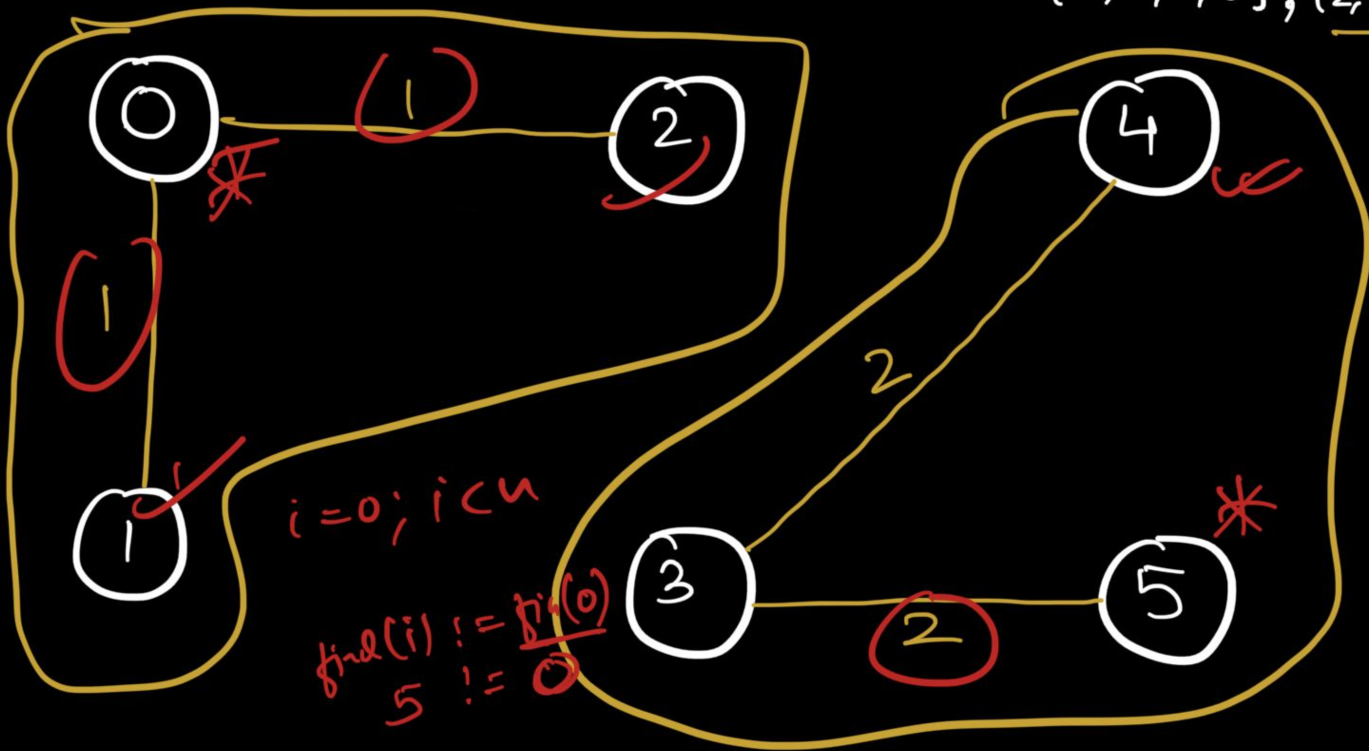
skip



MST-WEIGHT = 10

{0,1,1,0}, {0,2,1,2}, {1,2,1,1}, {3,4,2,4}, {3,5,2,5},
{4,5,2,6}, {2,3,4,3}}

skip



V-1

V=6

6

if (Kruskal(i, -1) > MSTw).

(Mi..

elseif. (____

6

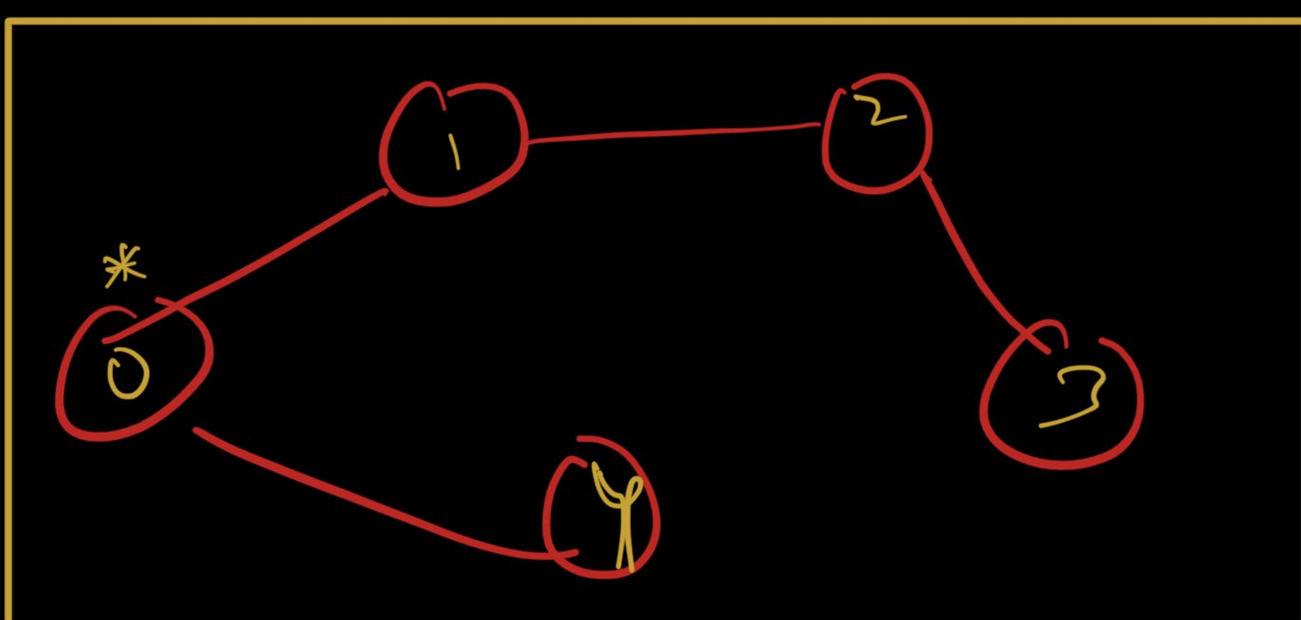
if (Kruskal(i, -1) > MST_c).

(mi..

elseif (______

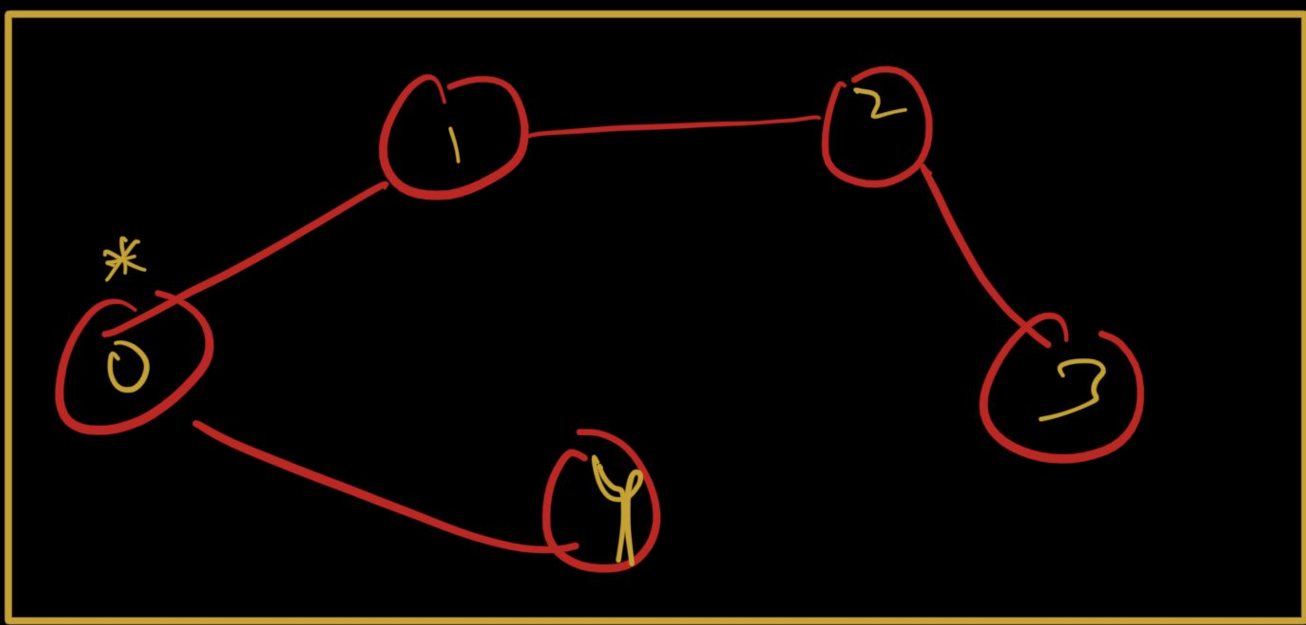
Krus!:

MST: (Connected).



n.

MST (Connected).



n.

```
for (i=0 ; i<n ; i++) {
    if ( find(i) != find(0))
                  return ∞
}
```