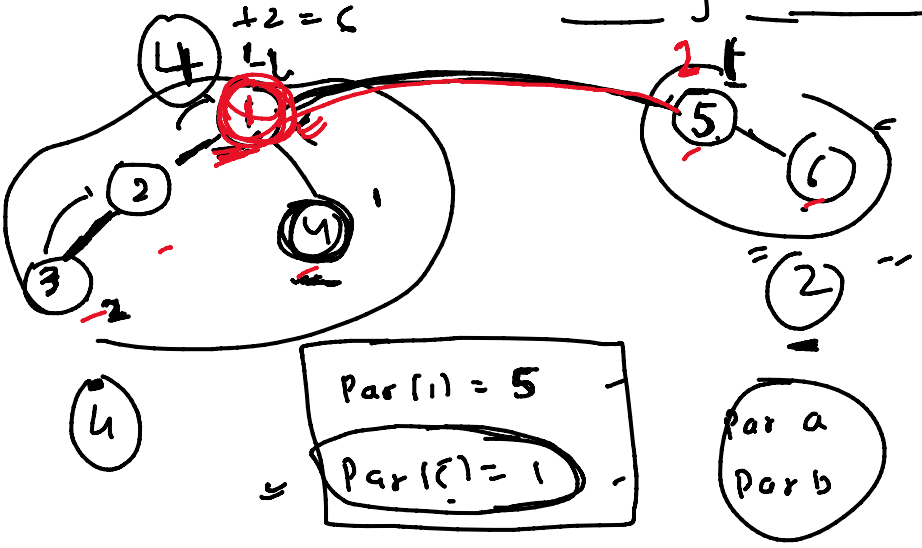
Class 99

31 May 2023 21:08

Class 99

Graphs

Union by Rank Method in DSU



find. Parent ( )

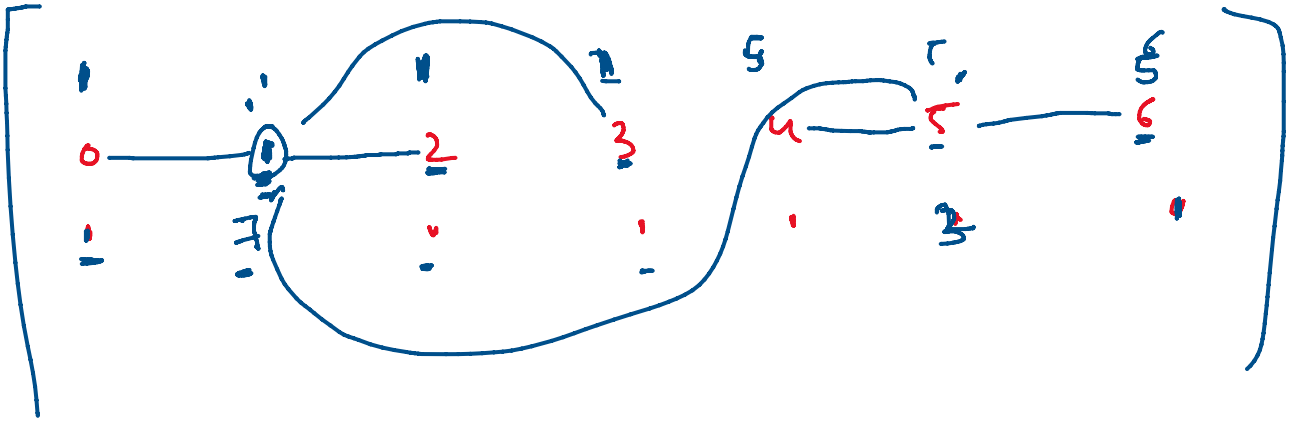
(omparent → Rank

Comp Size

Par a

Par b

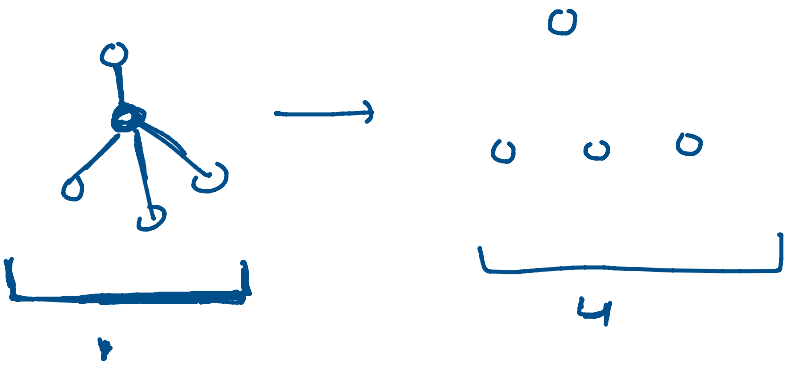
```
if (sz[para] > sz[parb])
{  parent[parb] = para;
   sz[para] += sz[parb];
}
else
{  parent[para] = parb;
   sz[parb] += sz[para];
}
```

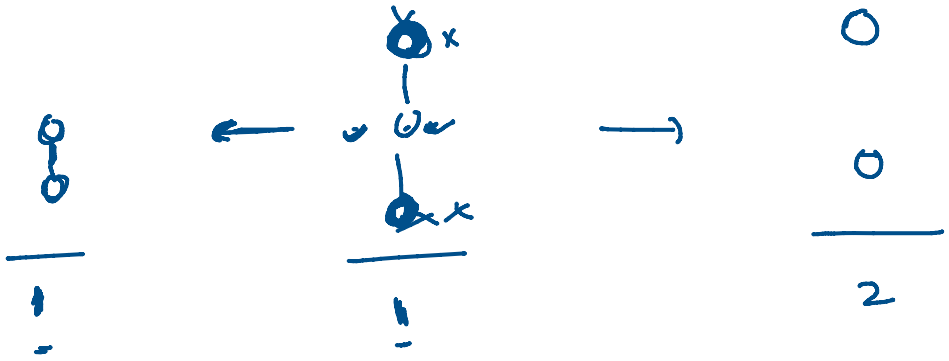


Articulation point and Bridges in Graph

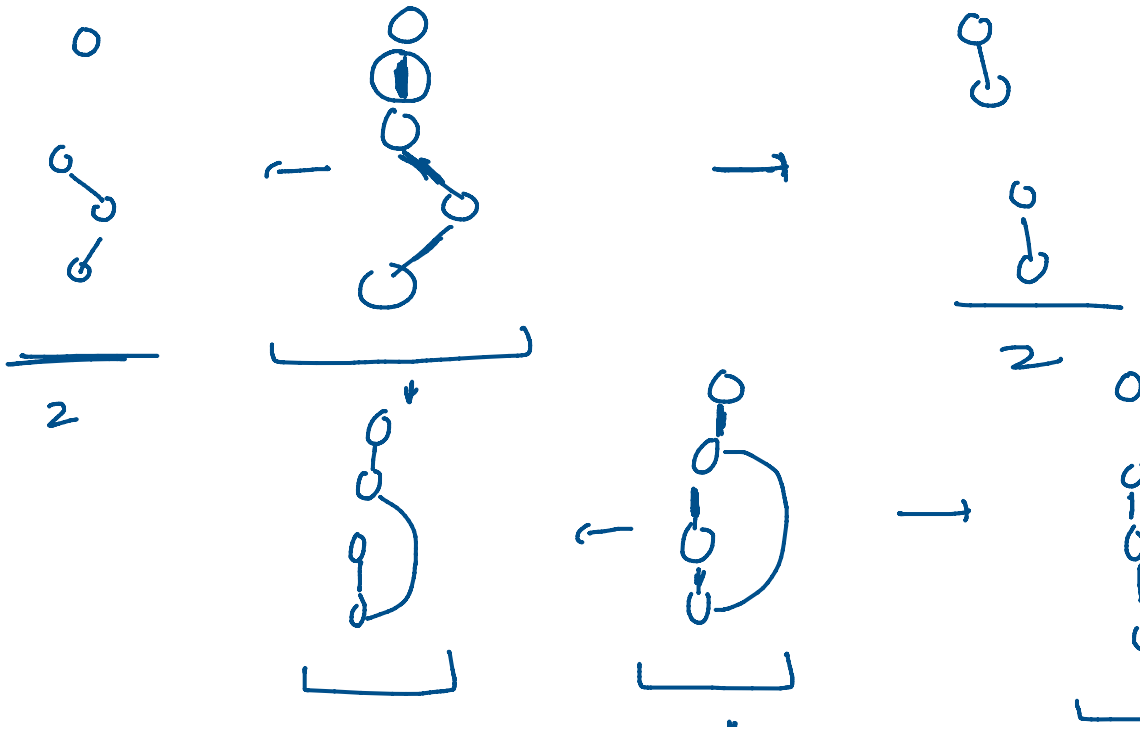
[Articulation point and Bridges in Graph]

Art Point





Bridge

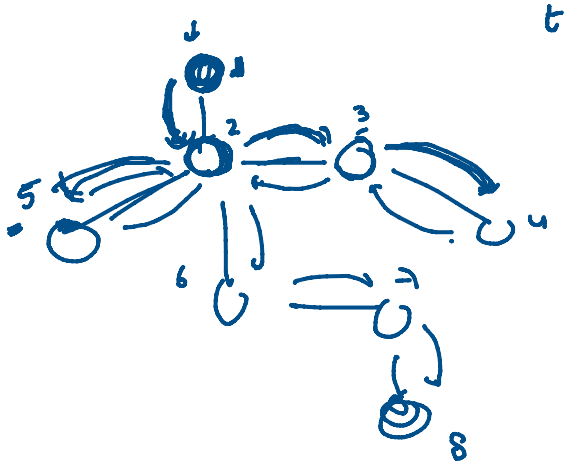


Discover Time :- Time at which current node is discovered

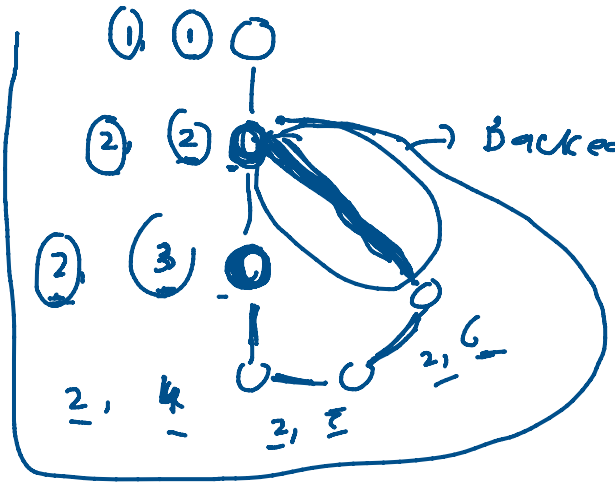
during DFS

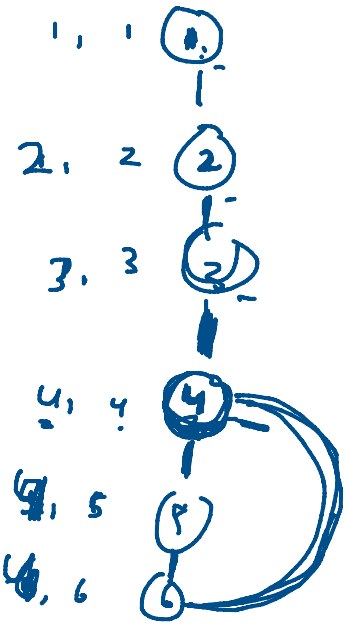
t = 2

BFS

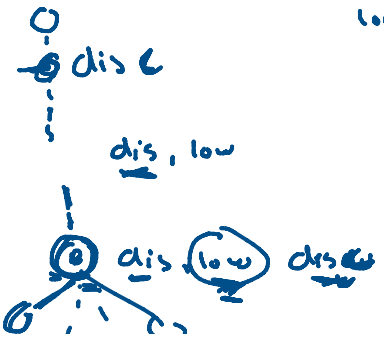


Lowest Time





Discover Time, lowest Time ✓

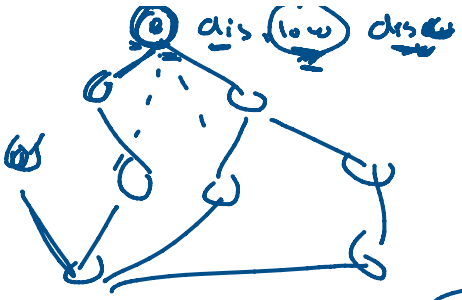


low[B] == dis A

A - B → Bridge!

low[B] ≤ dis[A]

low[B] ≤ dis[A]



Node A

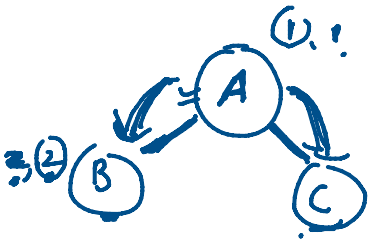
Articulation Point

Bridge x

low[B] > dis[A]

Bridge

low[B] ≥ dis[A]



unconnected ckt

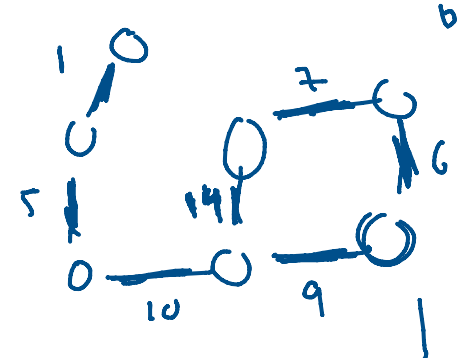
No of Rec > 1

Code

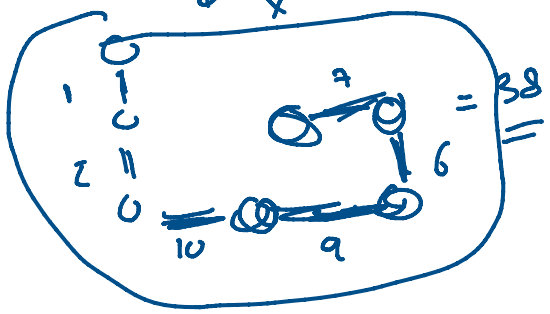
Spanning Tree

backedge

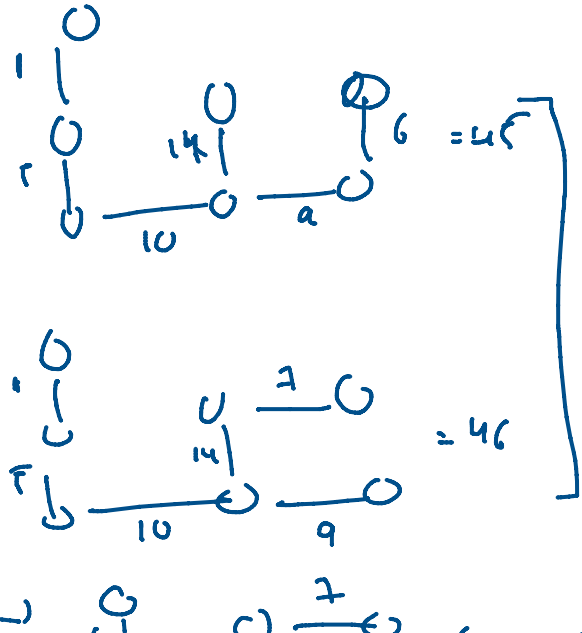
1, 5, 6, 7, 9, 10, 14



x



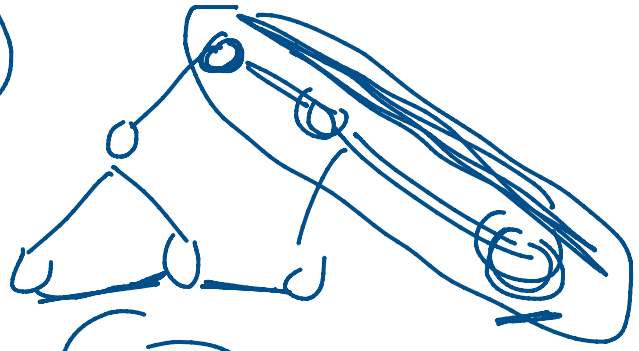
= 38



cycle

Minimum spanning Tree

Find MST



1, 5, 10

Kruskal Algorithm using DSU

→ Sort edges

→ Pick edge one by one, if edge is making cycle then leave it

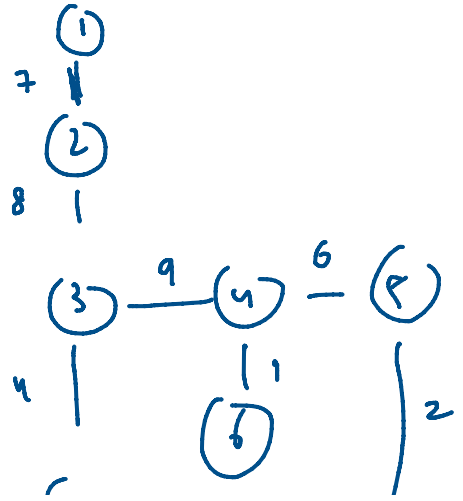
```
void kruskal ( vector < vector <int> > edges , int n)

{   sort ( edges.begin(), edges.end(), cmp );

    vector <int> Parent (n);

    for (i=0; i<n; i++)
    { Parent[i] = i;
    }

  int cost = 0;

vector < vector<int> > ans ;

for (auto it : edges)

{  int u = it[0];
```

(A) — (B)

```
{ int u = it[0];
  int v = it[1];
  int wt = it[2];
  if( find_par(u, parent) != find_par(v, parent) )
  { cost += wt;
    [ans.push_back(it);]
    union(u, v, parent);
  }
}
return ans
```

```
bool cmp(vector<int>a, vector<int>b)
{ if(a[2] < b[2])
    return true;
  return false;
}
```

4 - 6 (1)
5 - 8 (2)
3 - 7 (4)
4 - 5 (6)
- 2 (7)



cost = 37