Class 100

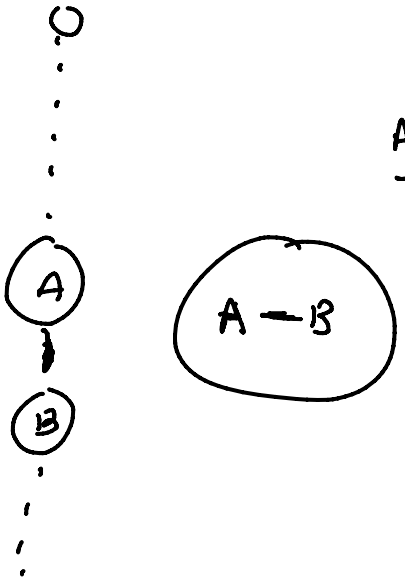
01 June 2023 21:09

Class 100

100 x 2 = 200 day

Graphs

Articulation Point, Bridges

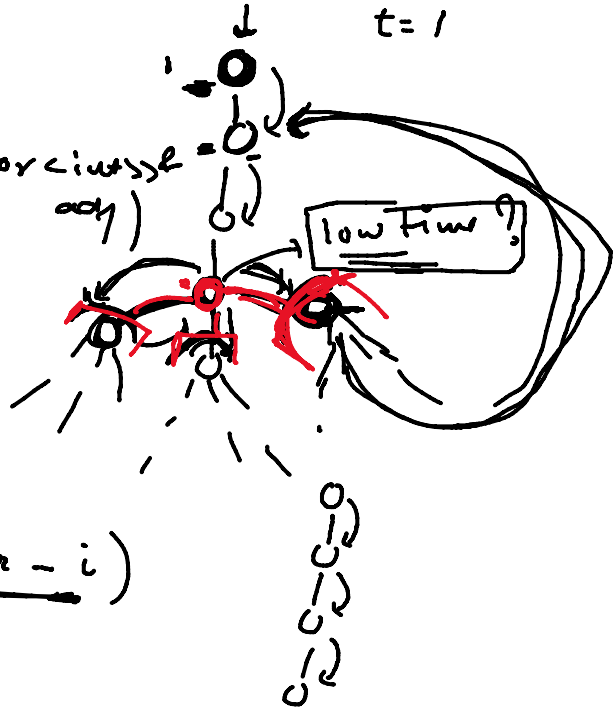


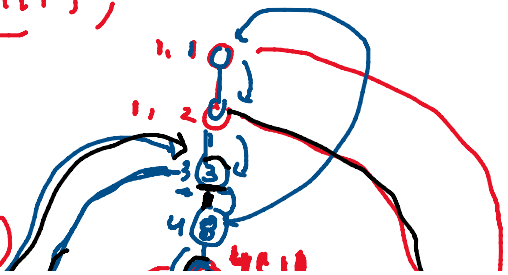
low[B] > dis[A]

low[B] ≥ dis[A]

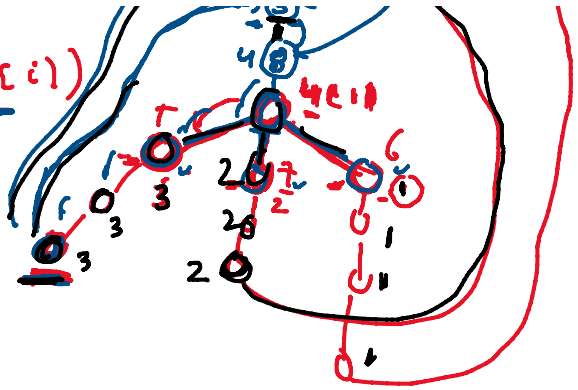
Root Node

```
int vis[N], dis[N], low[N], time=1

void dfs ( int cur, int par, vector<vector<int>> adj)
{   vis[cur] = 1;
    dis[cur] = time;
    low[cur] = time;
    time ++;
    int child=0;
    For (auto i : adj[cur])
    {   if ( ! vis[i] )
        {   dfs( i, cur, adj); child ++;
            low[cur] = min ( low[cur], low[i] );
        }
        else if ( i != par )
        {   low[cur] = min (low[cur], dis[i]);
```





low[cur] = min {low[cur], dis[i]}



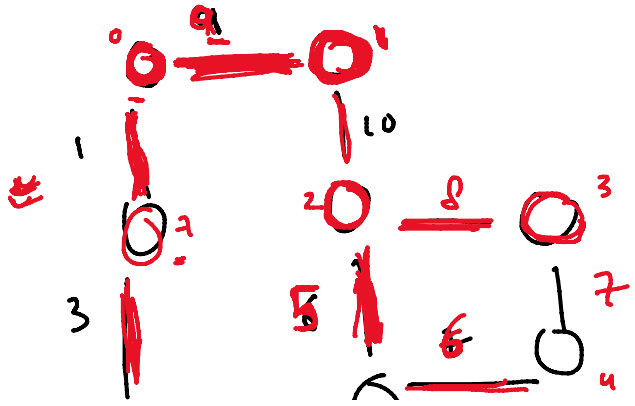
// Bridge and articulation point

```
if ( low[i] > dis[cur] )
    bridge.ph (cur, i);

if ( low[i] ≥ dis[cur]  and  cur != root )
    Articulation Point.ph (cur);
```

```
if ( cur == root  &&  child > 1 )
    arti_point.ph (cur);
```

MST — Kruskal Algorithm using DSU

Prims Algorithm



[5, 3, 2]

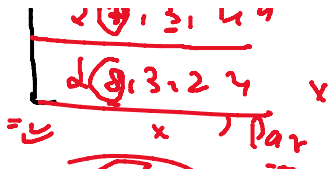
[6, 5, 2]

[7, 4, 5]

[8] 4, 3 ✗

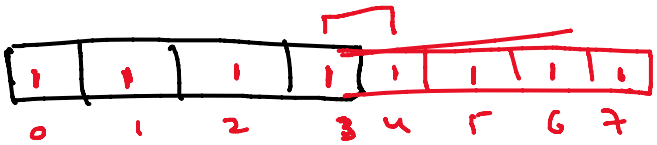
[3, 3, 4]

[3, 3, 2] ✗



BFS → O(V+E)

PQ



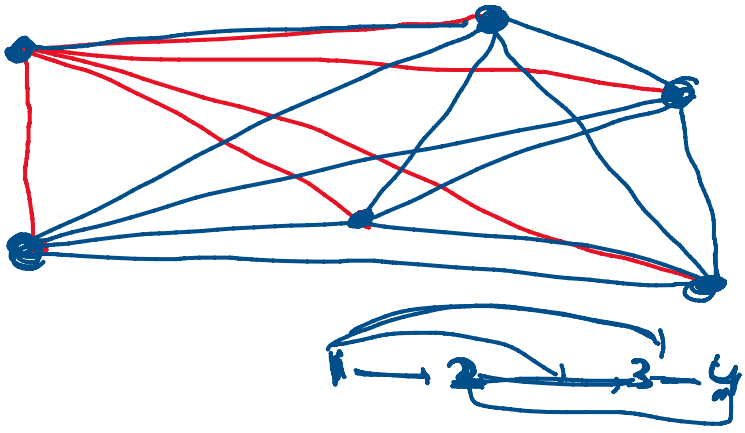
→ (V log V) (n log n)

```
int Parent[N];
int key[N]; // store distance of Nodes
bool mstSet[N];

For (i=0; i<n; i++)
  { key[i] = INT_MAX;
    mstSet[i] = 0;
  }                          wt node

priority_queue < pair<int,int>, vector < pair <int,int> > ,
                                  greater < pair <int,int> > > pq;

key[0] = 0;

pq.push( { 0, 0 } );

Parent[0] = -1;

while ( !pq.empty() )

{ int u = pq.top().second;
    pq.pop();
    key[u] = pq.top().first;
    mstSet[u] = true;
```

```
For (auto it: adj[u])
{ int v = it.first;
  int wt = it.second;
  if (mstSet[v] == false && wt < key[v])
  {
    parent[v] = u;
    key[v] = wt;
    pq.push( { key[v], v } );
  }
}
```

key[3] = 5

Concept



MST

Tree

Min

Min spanning Tree

Kruskal's, Prim

```
vector<vector<int>> edgelist;
For (i=0; i<Points.size(); i++)
{ int x = Points[i][0];
  int y = Points[i][1];
```

```
For ( j = i+1; j < point.size(); j++)
{ edgelist.Push_back( { abs(Point[i][0]-x) +
                        abs(Point[j][1] - y),
                        {i,{j,1}} );
}
}

sort(edge_list.begin(), edgelist.end());
DSU d(Point.size());
int ans=0;

For ( i=0; i<edgelist.size(); i++)
{   if ( d.are_same ( edge[i][1], edge[i][2] )
            continue;

    else
    { ans += edge list[i][0];
      d.union( edgelist[i][1], edgelist[i][2]);
    }
}

return ans;
```