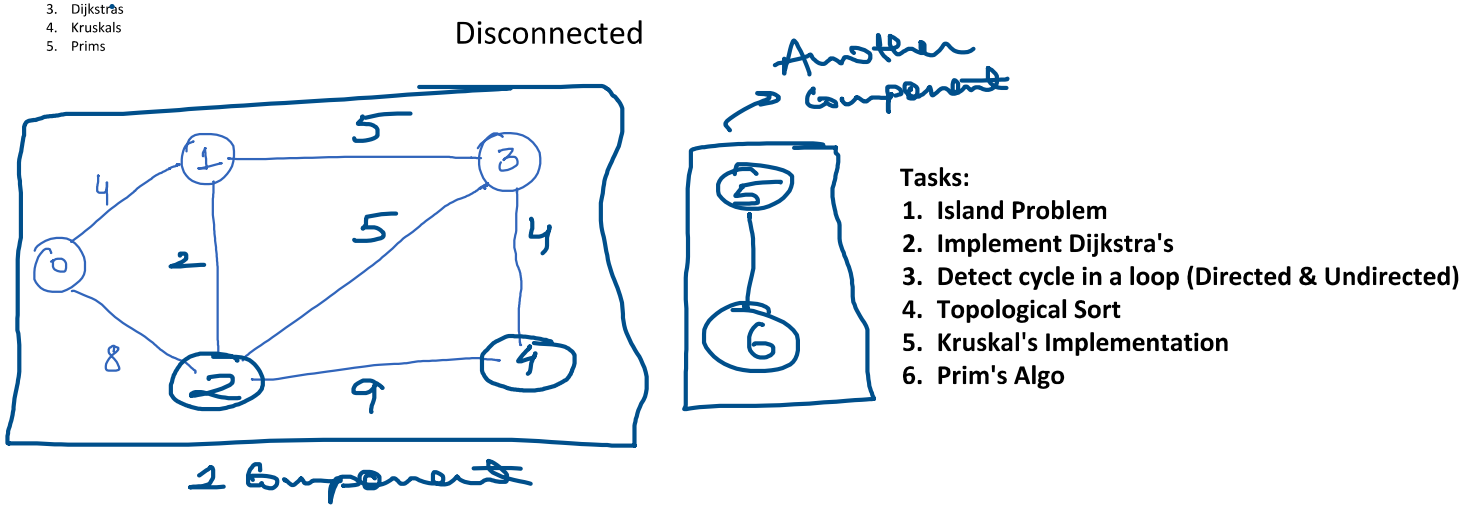
this notebook

June 22, Graphs

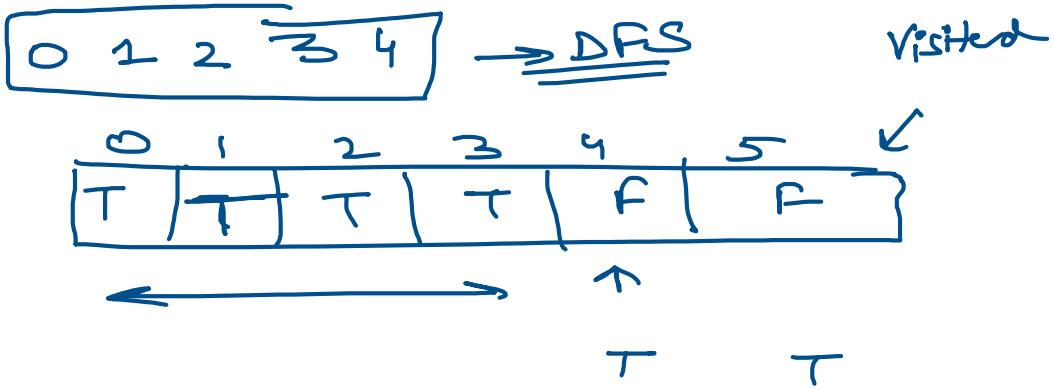
22 June 2020 19:58

1. Connected Components //Done
2. Greedy Algorithms
3. Dijkstras
4. Kruskals
5. Prims



Tasks:

1. **Island Problem**
2. **Implement Dijkstra's**
3. **Detect cycle in a loop (Directed & Undirected)**
4. **Topological Sort**
5. **Kruskal's Implementation**
6. **Prim's Algo**



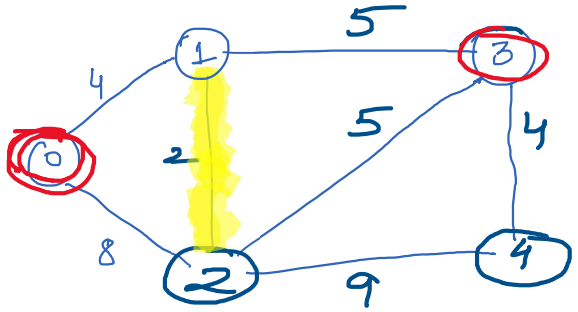
Try the famous "Island Problem" - Based on Connected Components

Greedy Algorithms

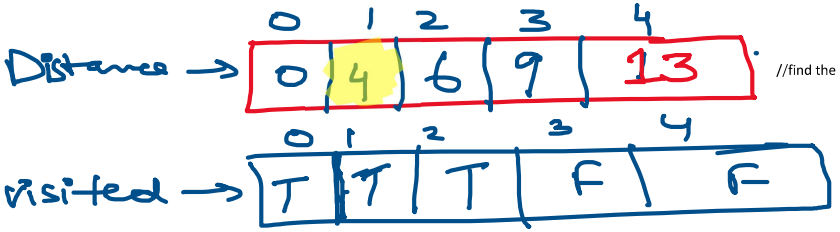
DIJKSTRA's Algorithm

Single source shortest distance algorithm

0 1 4
0 2 8
1 3 5
1 2 2
2 3 5
2 4 9
3 4 4



1st -> Dist[src] = 0



//find the shortest node that is not visited

Distance[1] + (1-2) edge weight

If newDistance < Distance[neighbour] then update value in distance array

Summarize

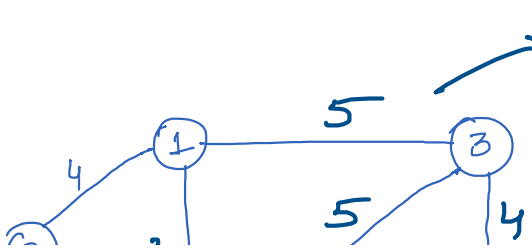
visited [ ] → [F, F — —]

Distance [ ] → [∞, ∞, ∞ — — —]

Distance[src] =

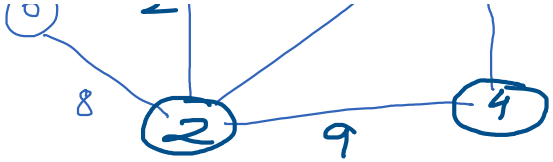
```
for(int i=0; i<V-1; i++)
{
    find minVertex i.e. not visited
    visited[minVertex] = true;
    for(every non visited neighbour of it)
    {
        newDistance = Distance[minVertex] + (minVertex to neighbour);
        if(newDistance < Distance[neighbour])
        {
            update neighbours distance value
        }
    }
}
```

MST → Minimum Spanning Tree



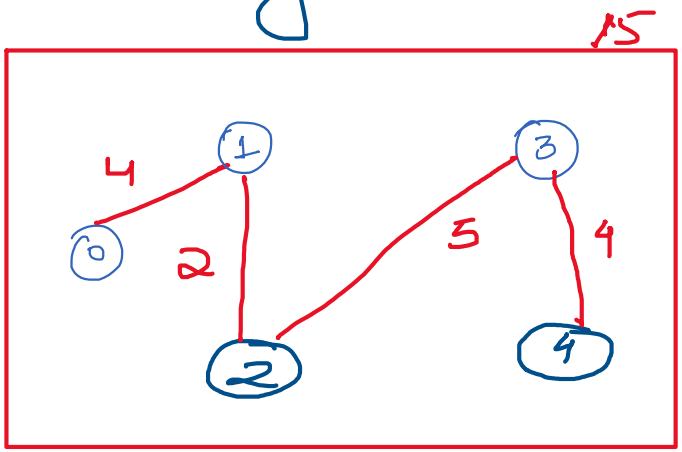
→ Graph

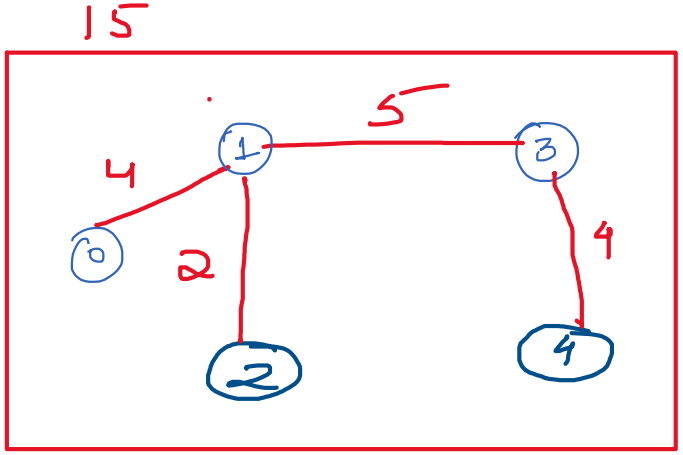
→ 5 vertices

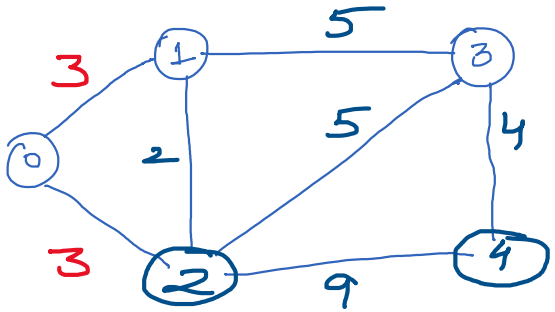


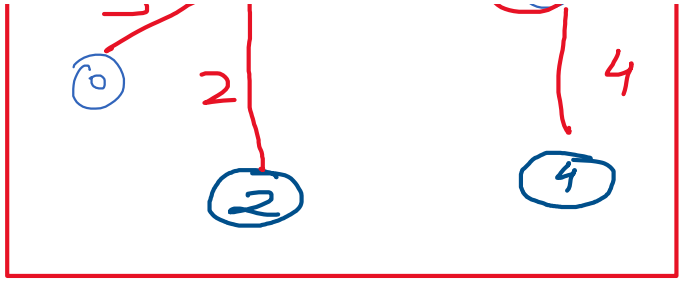
If we make a tree

Edges ⇒ n-1 [4]









Kruskal's Algorithm

1. Edge list
2. Sort in ascending order on basis of weight
3. Start picking edges from sorted list

→ Add to MST if it doesn't create a cycle

→ Stop when v-1 edges added.