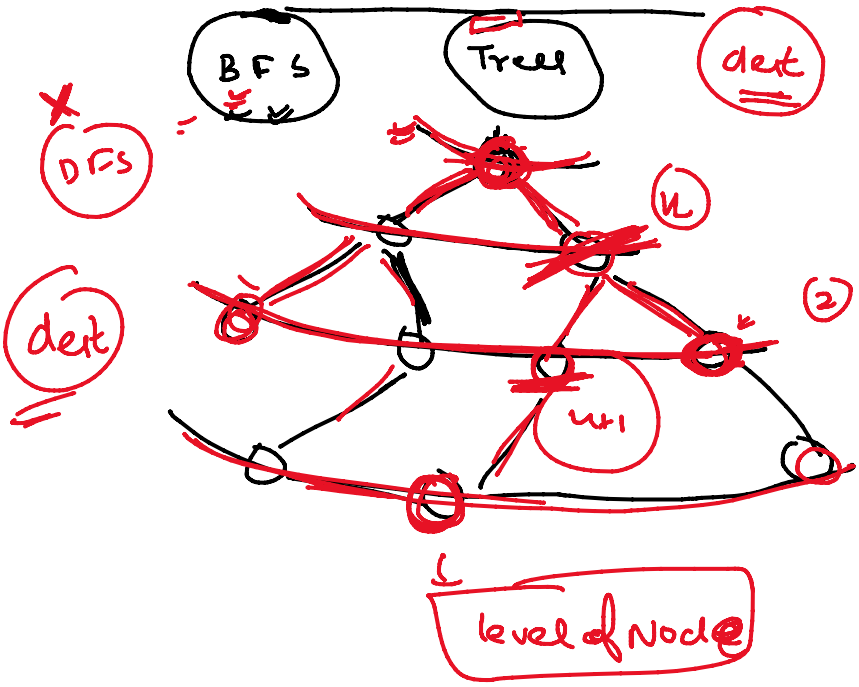
Class 96

27 May 2023 14:08

Class 96

Graphs

Unweighted Graph



layerwise visit

MinDistance → level of two Node from Given Source

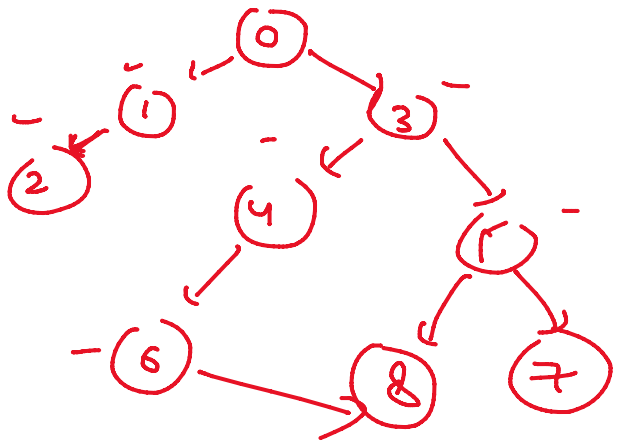
```
vector<int> bfs ( vector<vector<int>>& adj, int src )
{
    vector<int> vis(n, 0);
    queue<int> q;
    q.push(src);
    vis[src] = 1;
    vector<int> dis(n,0);
    dis[src] = 0;

    while( !q.empty() )
    {
        int tp = q.front();
        q.pop();

        for(auto i : adj[tp])
```

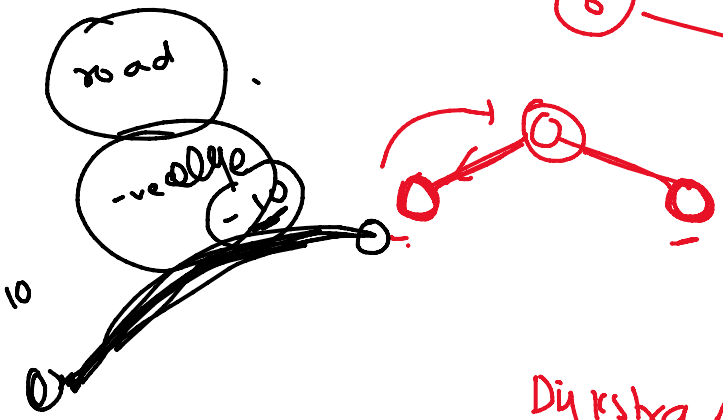
src → in node

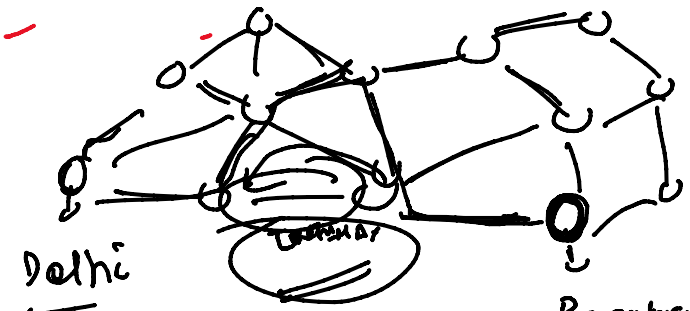
```
for(auto i: adj[tp])
{
    if(vis[i]==0)
    {
        vis[i]=1;
        av.push(i);
        dis[i]= dis[tp]+1;
    }
}
return dis;
```

if (i==dest) return dis[i];

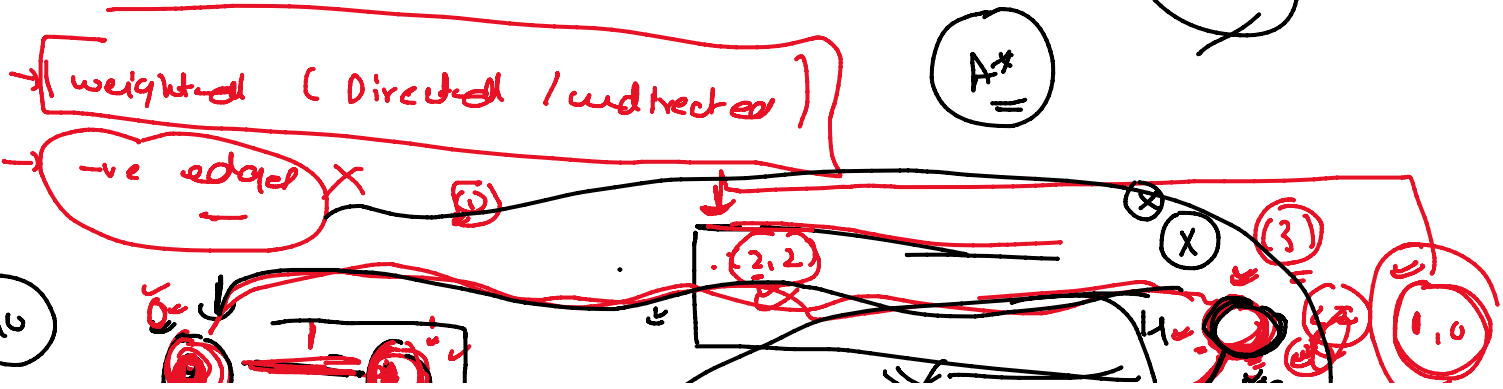
dis[node];

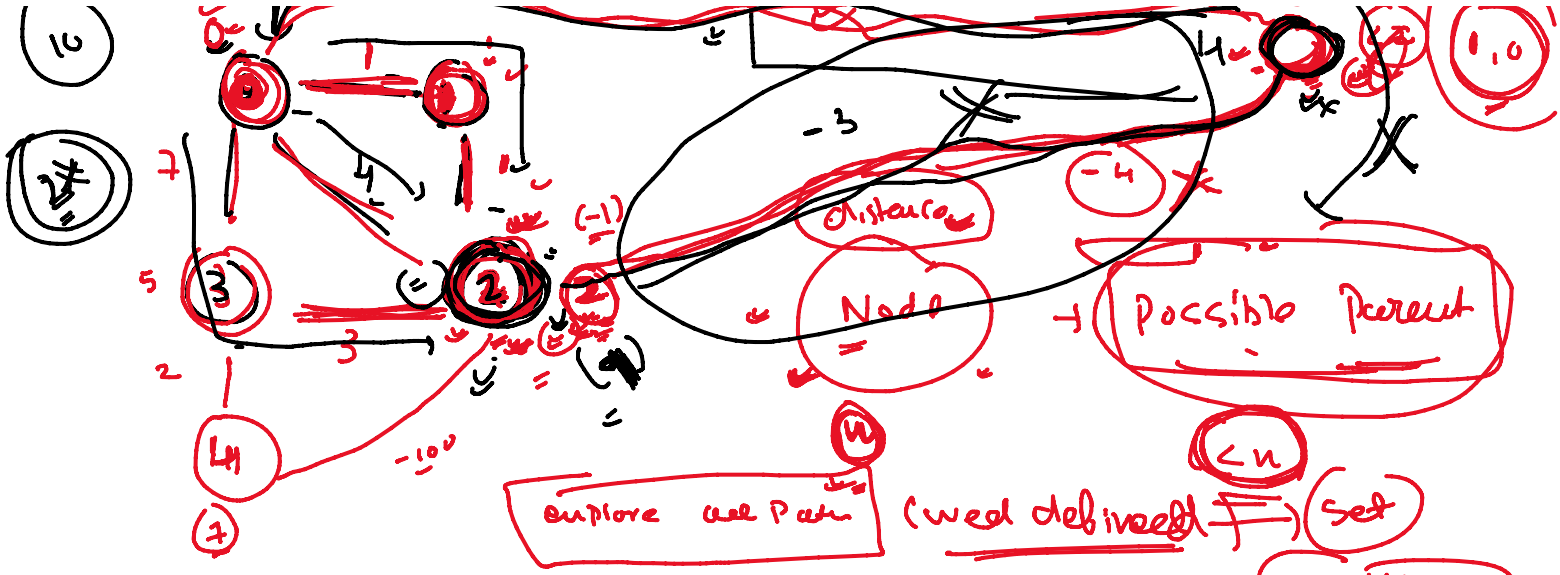






Dijkstra Algorithm



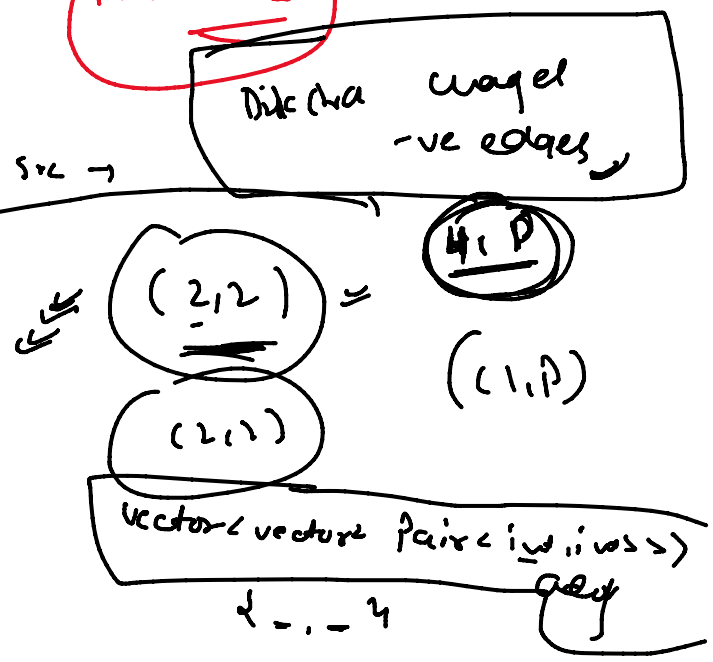



```
vector<int> dis(v, INT_MAX);
set<pair<int,int>> s;
dist[src] = 0;
s.insert({0, src});
while (!s.size())
{
    auto it = s.begin();
    int node = it->second;
    int dist = it->first;
    s.erase(it);
```

Priority-queue

Dijkstra weighted -ve edges

src →




```
    for (auto nbrpair : adj[node])
    {
        int nbr = nbrpair.second;
        int edge = nbrpair.first;
        if (dist + edge < dist[nbr])
        {
            if (s.find({dist[nbr], nbr}) != s.end())
                s.erase({dist[nbr], nbr});
```

if (s.find({dist[nbr], nbr}) != s.end())
s.erase({dist[nbr], nbr})
}
dist[nbr] = dist + edge;
s.insert({dist[nbr], nbr});