Class 97
29 May 2023 21:08

Class 97

Graphs

Shortest Path Algorithm

BFS → Dijkstra Algorithm

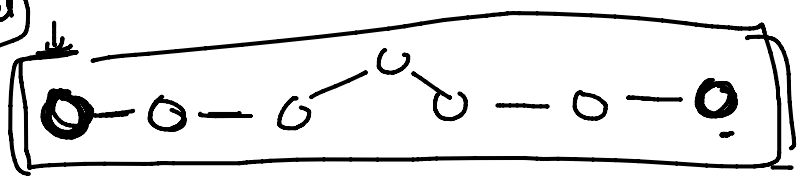
Bellman Ford Algo

Dijkstra
- weighted
- Greedy
- -ve weight edges ✗
- Elogv

Bellman Ford
- weighted
- DP
- -ve edges
- (V^2)
- Detect -ve wt cycle

Graph will always be connected

source



Max No of edges in the Final shortest Path b/w source and Node

Tree → edges (n-1)

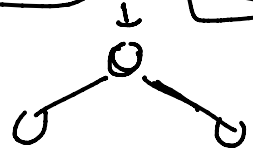
(n) nodes (n-1)

m edges
n nodes

→ backedges

m - (n-1)

Tree + Backedges → Graph = Tree

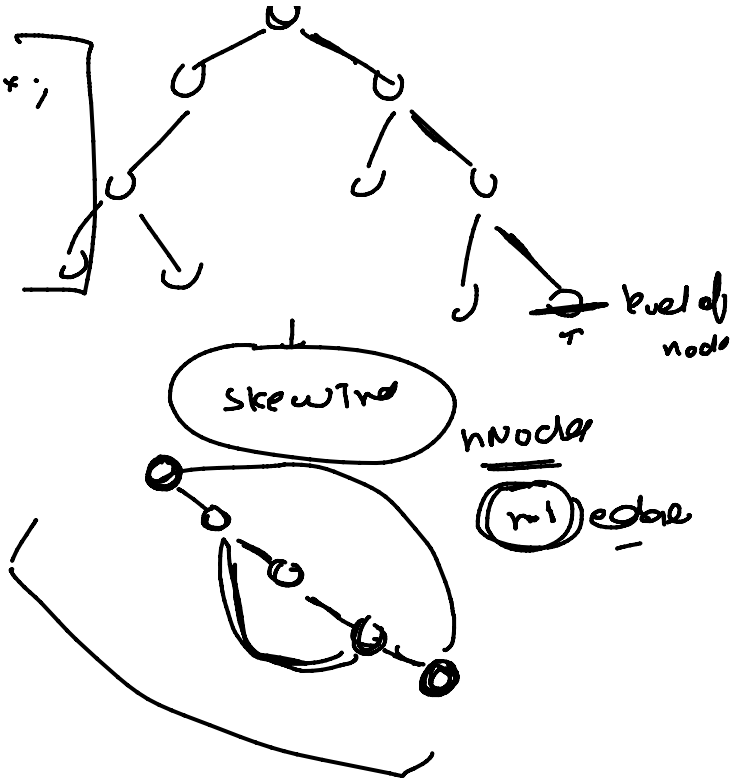


→ Initialize dist vector with INT_MAX;

→ Initialize dist vector with INT_MAX;

→ Relax all edges (N-1) times

→ Optimize



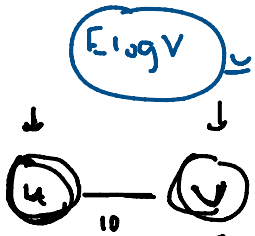

```
vector<vector<int>> edges;
```

edges[i] = { u, v, wt }

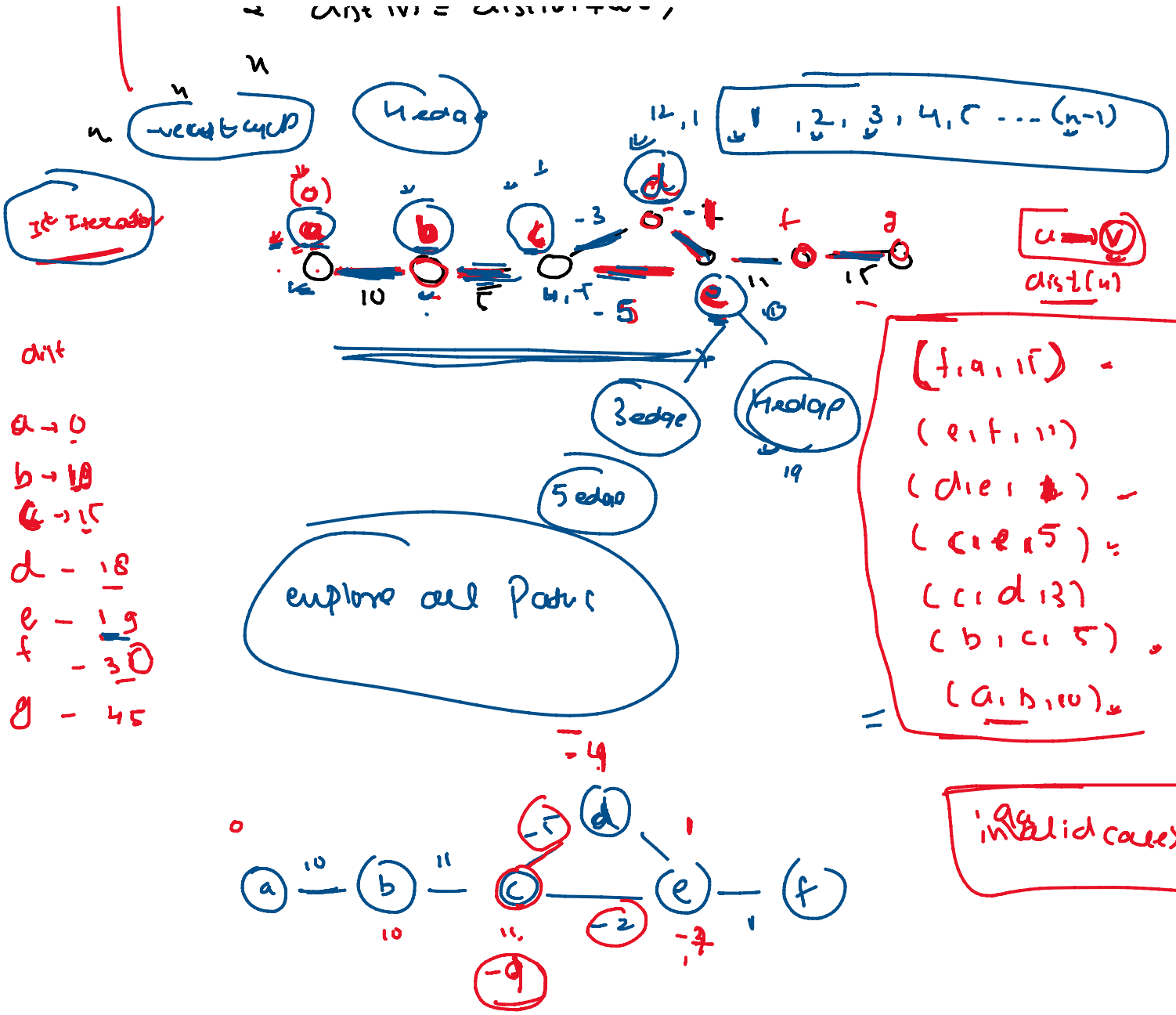
```
vector<int> dist(n+1, INT_MAX);
dist[src] = 0;
for(i=0; i<n-1; i++)
    for(auto edge : edges)
        int u = edge[0];
        int v = edge[1];
        int wt = edge[2];
        if(dist[u] != INT_MAX && dist[u]+wt < dist[v])
            dist[v] = dist[u]+wt;
```

V × E

E log V



dist[u] + edgewt

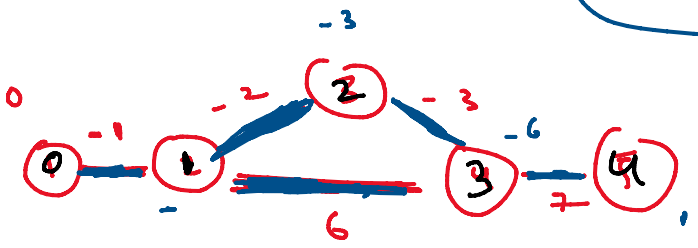


(n-1)

```
For (auto edge : edges)
{  int u = edge[0];
   int v = edge[1];
   int wt = edge[2];
if ( dist[u] != INT_MAX and dist[u] + wt < dist[v])
```

} return "Cycle Present";

Best case



1-2

(n-1)

1 - 2
2 - 3
3 - 4
2 - 4
4 - 5

Floyd warshall

All Pair shortest Path

Dijkstra
Bellman

Intermediate Node

(1) - (2) + (2 - 4)

1 - 2
1 - 3
1 - 4
1 - 5
2 - 3
3 - 4
2 - 5
3 - 4
4 - 5

Adjacency Matrix

1 - 3
+
3 - 4

	1	2	3	4	5
1	0	10	0	0	0
2	10	0	1	0	0
3	0	11	0	2	0
4	0	0	2	0	1
5	0	0	0	1	0

vector<vector<int>> dis(n, vector<int>(n,0));

dis[i][j] =

Bottom up DP

vector<vector<int>> dis(n, vector<int>(n,0));

dis[i][j] = Bottom up DP (V³)

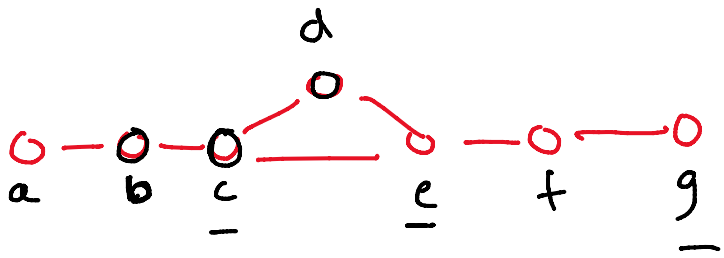
Fix Intermediate Node

```
for (int k = 0; k < V; k++)
    for (i=0; i<V; i++)
        for (j=0; j<V; j++)
            if (dist[i][k] == INT_MAX || dist[k][j] == INT_MAX)
                continue;
            else if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j] = dist[i][k] + dist[k][j];
```

⇒ Pairs generate

i - j

k



c - g

c - a + a - g

- [a - e]

[(a - b) + (b - e)
  (a - c) + (c - e)
  (a - d) + (d - e)]