L 23: Graph Based Algorithms (L1)

Graphs: Review

- **Storage**: Adjacency Matrix and Adjacency Lists
- **Adjacency Lists**:

  Storage 1: node *graph[50];

  Storage 2: node graph[50];
- Do you know, which one is correct?
- How do these differ?

Graphs: Traversal & MST

- DFS: uses concept of backtracking, recursive
- Conventional: For DFS as well as BFS, visited[] is used to mark a vertex as visited;
- Try writing code for DFS, use visited[]

- Time Complexity?
- BFS: breadth wise: easy, iterative
- Try writing code for DFS, use visited[]

- Time Complexity?

Graphs: Traversal & MST

- MST: Kruskal, Prim's algo
- Kruskal Algo: Uses disjoint sets, we will discuss that in coming lectures and then we will cover Kruskal algo
- Prim's Algo: Each step adds an edge to set A such that the edges in set A always form a single tree

Prim's Implementation

- Prim's Algo: Each step adds an edge to set A such that the edges in set A always form a single tree
- Effectively it is a greedy algo.
- Start with a start vertex S, add the most light edge, safe for A till all vertices get covered:
- Safe: To a tree A, add the lightest edge that connects A to an isolated vertex
- How to implement:
- Efficient implementation should be planned

Prim's Implementation

- Let A be initial MST having start vertex S. Each time, we need a vertex with min-wt edge from A's vertices.
- So Min-Heap of vertices should be used.
- MH should be prioritized such that root gets selected in next iteration,
- So key for a vertex v of MH should be min-weight out of all edges connecting v to any vertex of A.
- Use Extract-Min and then _____

- then use decrease-key for all ***relevant*** vertices in adj list of U

```
for(int i = 0; i < V; i++)
{       Node U = extractMin(PQ, heapsize);
        start=graph[U];
        while (start !=NULL)
        {       v=start->vertex; new_wt=v->wt;
                if(v belongs to PQ && new_wt < dist of v stored in PQ)
                {       Parent of v in PQ= v;
                        decreaseKey( );
                }       }       }
```

How to check whether v belongs to PQ: use visited[ ]

How to compare new_wt with min_dist of v ?

Some students: all unvisited adj. of v pushed in PQ STL. Is it good?

Size of PQ increased. Time of PQ operations: > O(logV); each time PQ size should decrease by 1.

Soln: Operate/decrease-key directly on v's node in PQ. How to find where is v in PQ?

Soln: store indexinPQ[0…n-1] & update whenever PQ operations done

Prim's Algo: Analysis

- Build-min-heap: O (V)
- Extract-min: O (log V) called V times: O(V Log V)
- Decrease-key will happen many times: O(_____)

Tomorrow's Topics

- **Disjoint Sets & Kruskal's Algo**
- **Strongly Connected Components**
- Read & Revise these topics