CS122 Algorithms and Data Structures

MW 11:00 am - 12:15 pm, MSEC 101

Instructor: Xiao Qin

Lecture 15: Graphs (2)

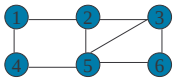
Depth first search

- Starting from vertex v
- Mark v as marked
- Select u as an unmarked node adjacent to v
- If no u, quit
- If u, begin depth first search from u
- When search from u quits, select another node from v
- Similar to preorder tree traversal

Breadth first search

- Starting from node v
- Identify all nodes adjacent to v
- Add these to the set
- Determine set of unvisited nodes which are adjacent to this set
- Add these to the set
- Continue until no new nodes are encountered

An Example



What would the visit orders for DFS(1), DFS(5), BFS(1), BFS(5) look like?

Unweighted Shortest Path

- Find the shortest path (measured by number of edges) from a designated vertex S to every vertex
- Simplified case of weighted shortest path

Algorithm

- Starting from node S
- Distance from S to S is 0, so label as 0
- Find all nodes which are distance 1 from S
- Label as distance 1
- Find all nodes which are distance 2 from S
- These are 1 step from those labeled 1
- This is precisely a breadth first search

Positive Weighted Shortest Path

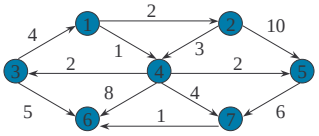
- Length is sum of the edges costs on the path
- All edges have nonnegative cost
- Find shortest paths from some start vertex to all vertices
- similar process to unweighted case
- Dijkstra's Algorithm

7

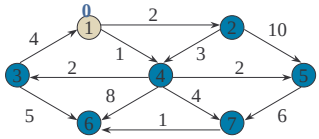
- Distance at each node v is shorted path distance from s to v using only known vertices as intermediates
- An example of a Greedy Algorithm
- Solve problem in stages by doing what appears to be the best thing at each stage
- *Decision* in one stage is not changed later

8

start at v1, all distances are infinity



mark v1 is removed from the toBeChecked set, with distance 0



9

Ford Algorithm

```
FordAlgorithm(G, s)
   for all vertices v do
      dist[v] = ∞; p[v] = NULL;
   dist[s] = 0;
   while there is (v,u) that dist[u]>dist[v] + weight(v,u) do
        dist[u] = dist[v] + weight(v,u);
        p[u] = v;
    endwhile
```

10

Spanning tree

- subgraph of G
- contains all vertices of G
- connected graph with no cycles

11

Minimum spanning tree

- spanning tree with minimum cost
- only exists if G is connected
- number of edges is |V|-1
- two greedy methods
  - Kruskal's algorithm
  - Prim's algorithm
- differ in how next edge is selected

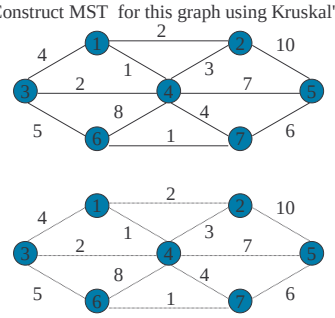
12

Kruskal's algorithm

- select edge with smallest weight as accept the edge if it does not cause a cycle
- determining if it causes a cycle: essentially the equivalence class (union/find) problem
- two vertices belong to the same set iff they are connected in the current spanning forest

13

Construct MST for this graph using Kruskal's algorithm



14

Prim's algorithm

- grow the tree in successive stages
- in each stage, one node is picked as the root, we add an edge, and thus a vertex is added to the tree
- have a set on vertices in the tree and a set that is not in the tree

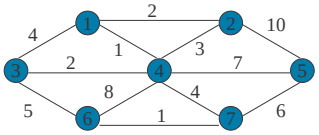
15

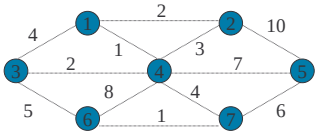
Prim's algorithm (cont.)

- at each stage, a new vertex to add to the tree is selected by choosing edge (u, v) such that the cost of (u,v) is the smallest among all edges where u is in the tree and v is not
- Build spanning tree starting from v1
- Result in the same spanning tree as that given by the Kruskal algorithm

16

Construct MST from v1 for this graph using Prim's algorithm





17