DATA STRUCTURES AND ALGORITHMS

LECTURE 26

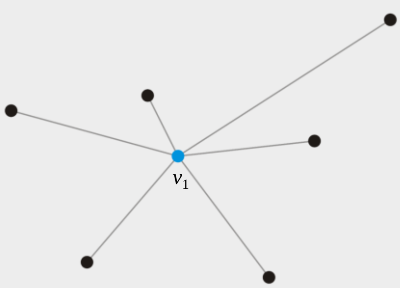
PRIM'S ALGORITHM

IMRAN IHSAN
ASSISTANT PROFESSOR
AIR UNIVERSITY, ISLAMABAD

STRATEGY

Suppose we take a vertex

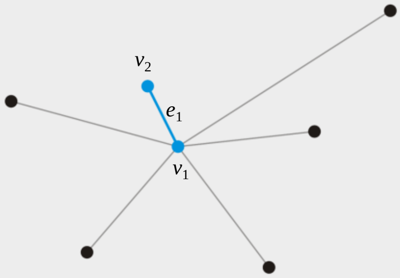
- Given a single vertex v_1, it forms a minimum spanning tree on one vertex



STRATEGY

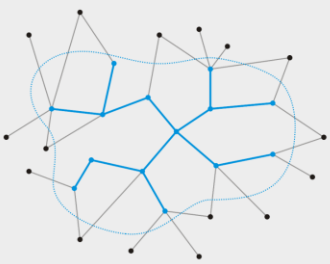
Add that adjacent vertex v_2 that has a connecting edge e_1 of minimum weight

- This forms a minimum spanning tree on our two vertices and e_1 must be in any minimum spanning tree containing the vertices v_1 and v_2



STRATEGY

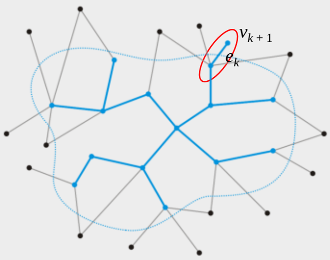
- Suppose we have a known minimum spanning tree on $k < n$ vertices
- How could we extend this minimum spanning tree?



STRATEGY

Add that edge e_k with least weight that connects this minimum spanning tree to a new vertex v_{k+1}

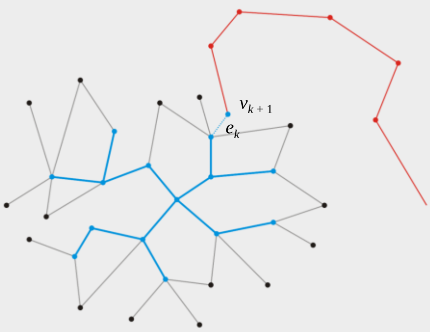
- This does create a minimum spanning tree on $k + 1$ nodes—there is no other edge we could add that would connect this vertex
- Does the new edge, however, belong to the minimum spanning tree on all n vertices?



STRATEGY

Suppose it does not

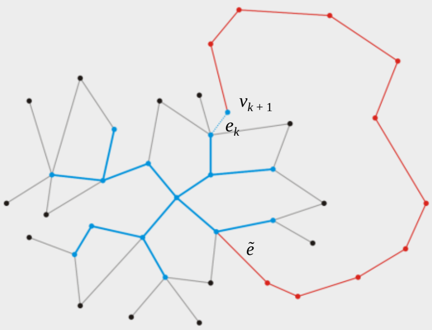
- Thus, vertex v_{k+1} is connected to the minimum spanning tree via another sequence of edges



STRATEGY

Because a minimum spanning tree is connected, there must be a path from vertex v_{k+1} back to our existing minimum spanning tree

- It must be connected along some edge $\tilde{e}$

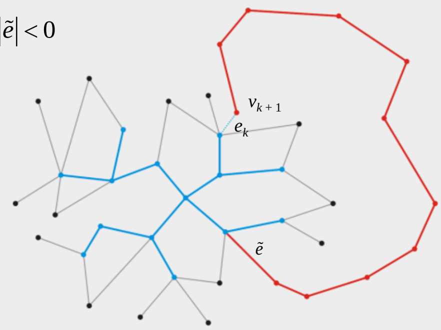


STRATEGY

Let w be the weight of this minimum spanning tree

- Recall, however, that when we chose to add v_{k+1}, it was because e_k was the edge connecting an adjacent vertex with least weight
- Therefore $|\tilde{e}| > |e_k|$ where $|e|$ represents the weight of the edge e

$$|e_k| - |\tilde{e}| < 0$$



MINIMUM SPANNING TREES

Prim's algorithm for finding the minimum spanning tree states:

- Start with an arbitrary vertex to form a minimum spanning tree on one vertex
- At each step, add that vertex v not yet in the minimum spanning tree that has an edge with least weight that connects v to the existing minimum spanning sub-tree
- Continue until we have $n - 1$ edges and n vertices

Another possibility is Kruskal's algorithm

PRIM'S ALGORITHM

Associate with each vertex three items of data:

- A Boolean flag indicating if the vertex has been visited,
- The minimum distance to the partially constructed tree, and
- A pointer to that vertex which will form the parent node in the resulting tree

For example:

- Add three member variables to the vertex class
- Track three tables

PRIM'S ALGORITHM

Initialization:

- Select a root node and set its distance as 0
- Set the distance to all other vertices as ∞
- Set all vertices to being unvisited
- Set the parent pointer of all vertices to 0

PRIM'S ALGORITHM

Iterate while there exists an unvisited vertex with distance < ∞

- Select that unvisited vertex with minimum distance
- Mark that vertex as having been visited
- For each adjacent vertex, if the weight of the connecting edge is less than the current distance to that vertex:
  - Update the distance to equal the weight of the edge
  - Set the current vertex as the parent of the adjacent vertex

PRIM'S ALGORITHM

Halting Conditions:

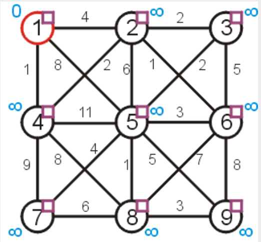
- There are no unvisited vertices which have a distance < ∞

If all vertices have been visited, we have a spanning tree of the entire graph

If there are vertices with distance ∞, then the graph is not connected and we only have a minimum spanning tree of the connected sub-graph containing the root

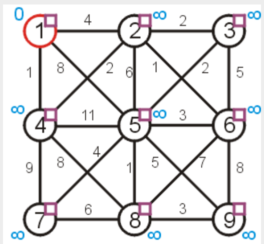
PRIM'S ALGORITHM

- Let us find the minimum spanning tree for the following undirected weighted graph



PRIM'S ALGORITHM

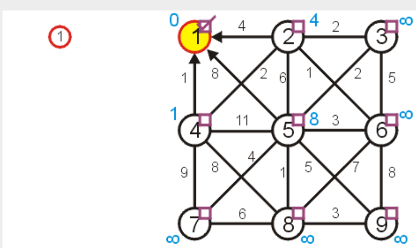
- First we set up the appropriate table and initialize it



		Distance	Parent
1	F	0	0
2	F	∞	0
3	F	∞	0
4	F	∞	0
5	F	∞	0
6	F	∞	0
7	F	∞	0
8	F	∞	0
9	F	∞	0

PRIM'S ALGORITHM

- Visiting vertex 1, we update vertices 2, 4, and 5

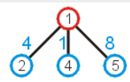


		Distance	Parent
1	T	0	0
2	F	4	1
3	F	∞	0
4	F	1	1
5	F	8	1
6	F	∞	0
7	F	∞	0
8	F	∞	0
9	F	∞	0

www.au.edu.pk Air University, Islamabad www.imranihsan.com

PRIM'S ALGORITHM

What these numbers really mean is that at this point, we could extend the trivial tree containing just the root node by one of the three possible children:

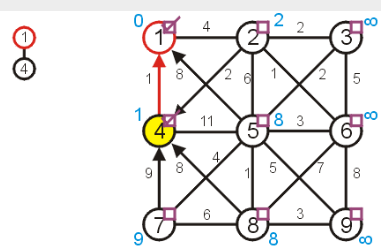


As we wish to find a *minimum* spanning tree, it makes sense we add that vertex with a connecting edge with least weight

PRIM'S ALGORITHM

The next unvisited vertex with minimum distance is vertex 4

- Update vertices 2, 7, 8
- Don't update vertex 5



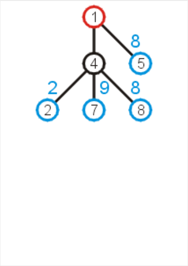
		Distance	Parent
1	T	0	0
2	F	2	4
3	F	∞	0
4	T	1	1
5	F	8	1
6	F	∞	0
7	F	9	4
8	F	8	4
9	F	∞	0

PRIM'S ALGORITHM

Now that we have updated all vertices adjacent to vertex 4, we can extend the tree by adding one of the edges

(1, 5), (4, 2), (4, 7), or (4, 8)

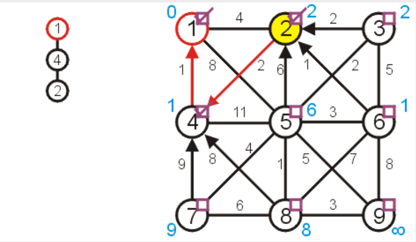
We add that edge with the least weight: (4, 2)



PRIM'S ALGORITHM

Next visit vertex 2

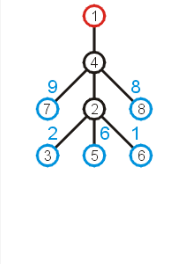
- Update 3, 5, and 6



		Distance	Parent
1	T	0	0
2	T	2	4
3	F	2	2
4	T	1	1
5	F	6	2
6	F	1	2
7	F	9	4
8	F	8	4
9	F	∞	0

PRIM'S ALGORITHM

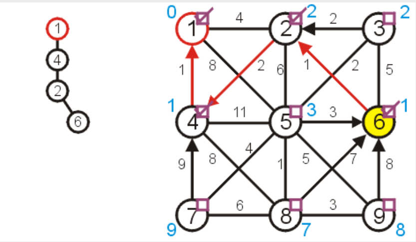
- Again looking at the shortest edges to each of the vertices adjacent to the current tree, we note that we can add (2, 6) with the least increase in weight



PRIM'S ALGORITHM

Next, we visit vertex 6:

- update vertices 5, 8, and 9

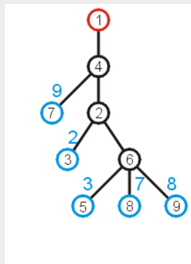


		Distance	Parent
1	T	0	0
2	T	2	4
3	F	2	2
4	T	1	1
5	F	3	6
6	T	1	2
7	F	9	4
8	F	7	6
9	F	8	6

PRIM'S ALGORITHM

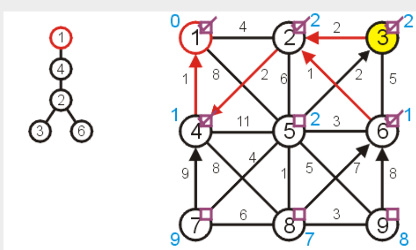
The edge with least weight is (2, 3)

- This adds the weight of 2 to the weight minimum spanning tree



PRIM'S ALGORITHM

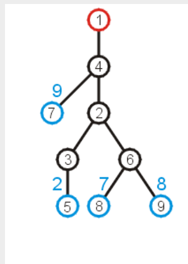
- Next, we visit vertex 3 and update 5



		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	F	2	3
6	T	1	2
7	F	9	4
8	F	7	6
9	F	8	6

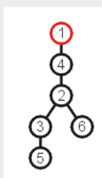
PRIM'S ALGORITHM

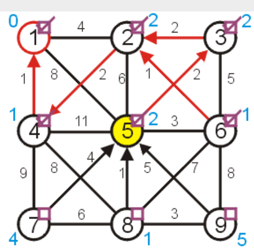
- At this point, we can extend the tree by adding the edge (3, 5)



PRIM'S ALGORITHM

- Visiting vertex 5, we update 7, 8, 9





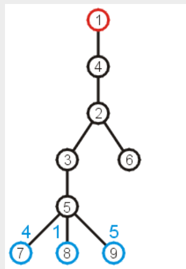
		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	T	2	3
6	T	1	2
7	F	4	5
8	F	1	5
9	F	5	5

www.au.edu.pk Air University, Islamabad www.imranihsan.com

PRIM'S ALGORITHM

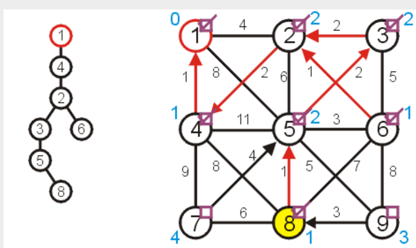
At this point, there are three possible edges which we could include which will extend the tree

The edge to 8 has the least weight



PRIM'S ALGORITHM

- Visiting vertex 8, we only update vertex 9

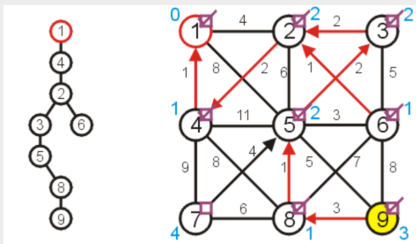


		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	T	2	3
6	T	1	2
7	F	4	5
8	T	1	5
9	F	3	8

www.au.edu.pk Air University, Islamabad www.imranihsan.com

PRIM'S ALGORITHM

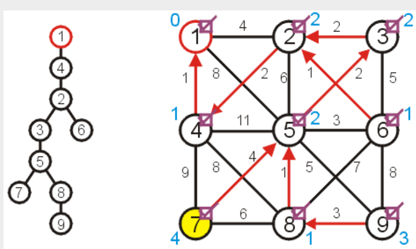
- There are no other vertices to update while visiting vertex 9



		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	T	2	3
6	T	1	2
7	F	4	5
8	T	1	5
9	T	3	8

PRIM'S ALGORITHM

- And neither are there any vertices to update when visiting vertex 7



		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	T	2	3
6	T	1	2
7	T	4	5
8	T	1	5
9	T	3	8

PRIM’S ALGORITHM

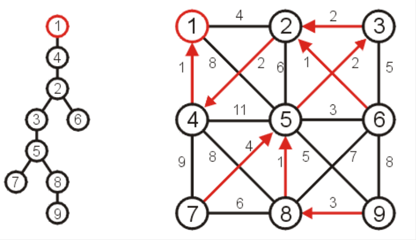
At this point, there are no more unvisited vertices, and therefore we are done

If at any point, all remaining vertices had a distance of ∞, this would indicate that the graph is not connected

- in this case, the minimum spanning tree would only span one connected sub-graph

PRIM’S ALGORITHM

- Using the parent pointers, we can now construct the minimum spanning tree



		Distance	Parent
1	T	0	0
2	T	2	4
3	T	2	2
4	T	1	1
5	T	2	3
6	T	1	2
7	T	4	5
8	T	1	5
9	T	3	8

PRIM'S ALGORITHM

To summarize:

- we begin with a vertex which represents the root
- starting with this trivial tree and iteration, we find the shortest edge which we can add to this already existing tree to expand it

This is a reasonably efficient algorithm: the number of visits to vertices is kept to a minimum