Lecture 9.1

- Anurag Sharma & Shymal Chandra

Minimum Spanning Trees with Greedy Algorithms

Spanning Trees

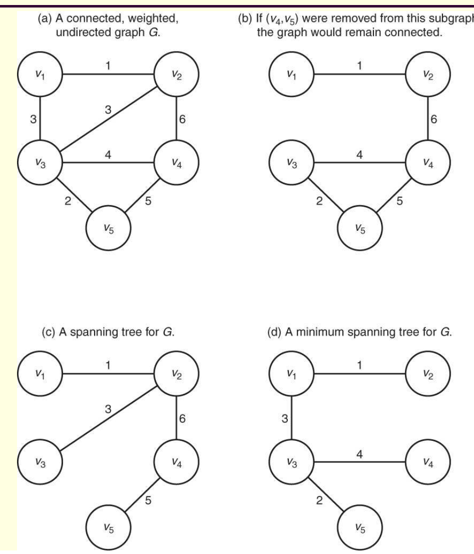
A review of graph theory:

- A graph is **undirected** when its edges do not have direction. An undirected graph is called **connected** if there is a path between every pair of vertices.
- An undirected graph with no simple cycles is called **acyclic**. A tree is an acyclic, connected, undirected graph.
- A **spanning tree** for a given graph is a connected subgraph that contains all the vertices of the given graph and is a tree.
- A spanning tree can be defined as an undirected graph G = , where V is the set of vertices and E is the set of edges

Minimum Spanning Trees

- If G is a weighted graph, the spanning tree will have a total weight.
- A graph may have different spanning trees, but not every spanning tree has the minimum weight
- A spanning tree with minimum weight is called minimum spanning tree
- The problem of finding the minimum spanning tree in an undirected, weighted, connected graph has many applications such as Google Maps, networking in telecommunications, operations research etc.
- We can represent such graphs using an adjacency matrix
- We will look at two algorithms (Prim's and Kruskal's) which produce minimum spanning trees

Minimum Spanning Tree Formation



Prim's Algorithm

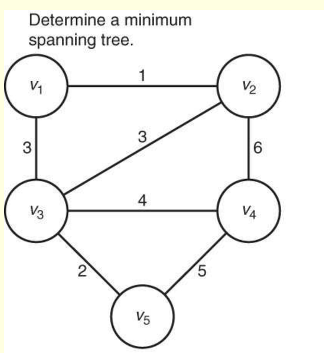
- Prim's algorithm starts with an empty subset of edges F and a subset of vertices Y initialized to contain an arbitrary vertex - we can initialize Y to {v1}
- The set of all vertices is the set V
- A vertex nearest to Y is a vertex in V – Y that is connected to a vertex in Y by an edge of minimum weight
- The vertex that is nearest to Y is added to Y and the edge is added to F
- Repeat these steps until all vertices have been included in the set Y

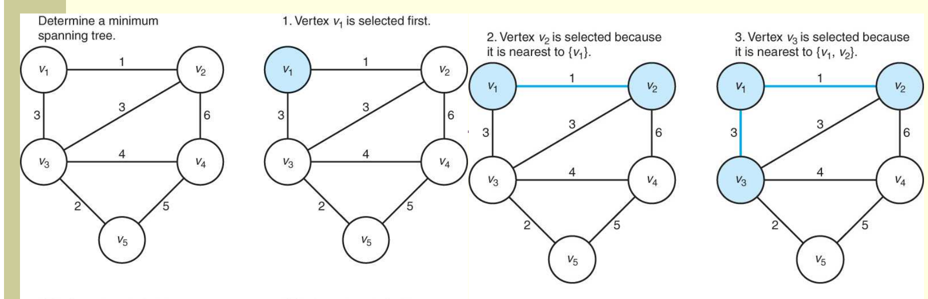
Pseudocode for Prim's Algorithm

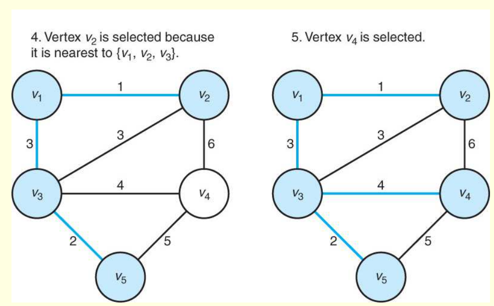
```
F = Ø;                                  // Initialize set of edges
                                        // to empty.
Y = {v1};                               // Initialize set of vertices to
                                        // contain only the first one.
while (the instance is not solved){

  select a vertex in V - Y that is      // selection procedure and
  nearest to Y;                         // feasibility check

  add the vertex to Y;
  add the edge to F;

  if (Y == V)                           // solution check
     the instance is solved;
}
```


Example

- Find the minimum spanning tree for the following graph:







Kruskal's Algorithm

- Kruskal's algorithm starts by creating disjoint subsets of V – one for each vertex and containing only that vertex, and an empty set of edges F
- It then inspects the edges according to increasing weight
- If an edge connects two vertices in disjoint subsets, the edge is added to F and the subsets are merged into one set
- This process is repeated until all the subsets are merged into one set and the final set F gives the minimum spanning tree

Pseudocode – Kruskal's algorithm

```
F = Ø;                                              // Initialize set of
                                                    // edges to empty.
create disjoint subsets of V, one for each
vertex and containing only that vertex;

sort the edges in E in nondecreasing order;

while (the instance is not solved){

  select next edge;                                 // selection procedure

  if (the edge connects two vertices in             // feasibility check
                    disjoint subsets){
      merge the subsets;
      add the edge to F;
  }
  if (all the subsets are merged)                   // solution check
      the instance is solved;
}
```