LECTURE 40: GRAPHS

Today

- Reading
  - JS Chapter 16
- Objectives
  - Minimum Spanning Trees
- Announcements
  - Updated final exam study guide posted on Piazza
  - Apply to be a mentor next year!

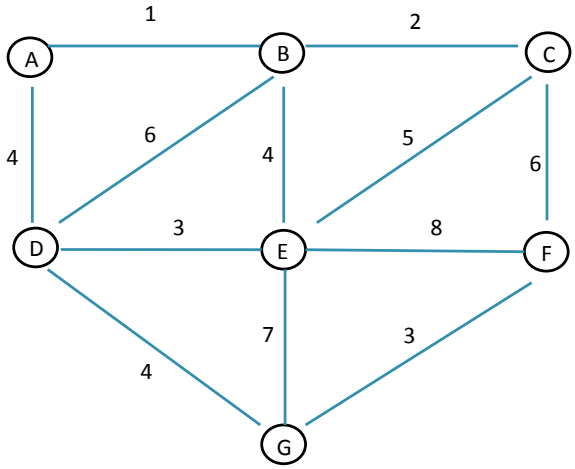
Dijkstra's Algorithm

```
map<int,int> shortest paths(int start,
                            const map<int,list<pair<int,int> > > & graph) {
  map<int,int> parents;
  priorityqueue62 frontier;

  parents[start]=start;
  frontier.push(start, 0);

  while (!frontier.is_empty()) {

    int v = frontier.top_serialnumber();
    int p = frontier.top_priority();
    frontier.pop();

    for (the neighbors (n,w) of v)
Case 1  if (n == parents[v])
          ; // do nothing
        else if (n is not in the frontier and has not been visited) {
Case 2    parents[n] = v;
          frontier.push(n, p + w);
        }else if (p + w < frontier.get_priority(n)) {
Case 3    parents[n] = v;
          frontier.reduce_priority(n, p + w);
        }

    } // end while

  return parents;
}
```

Minimum Spanning Trees

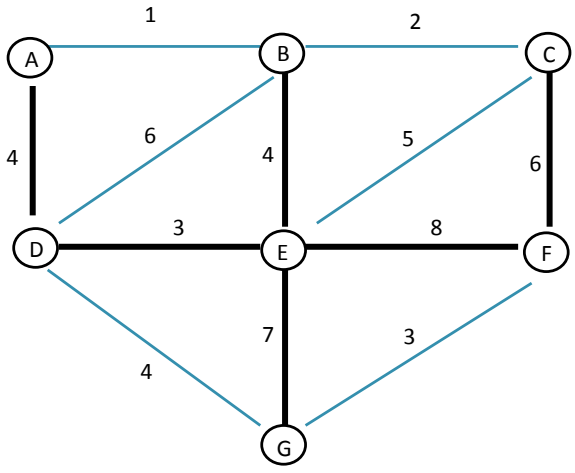
- G' = (V', E') is a subgraph of G=(V,E) if G' is a graph and V' is a subset of V and E' is a subset of E
- A spanning tree is a subgraph of G that is a tree and connects all of the vertices together
- A minimum spanning tree is a spanning tree whose weight is <= the weight of any other spanning tree
- Weight is the sum of the weights of the edges

Graph



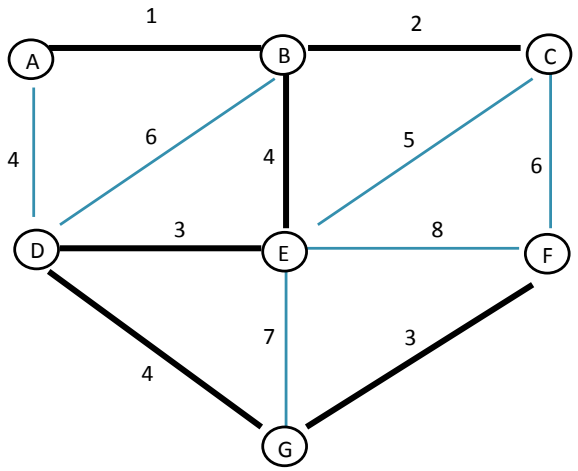
Spanning Tree

weight = 32



Minimum Spanning Tree

weight = 17



Prim's Algorithm

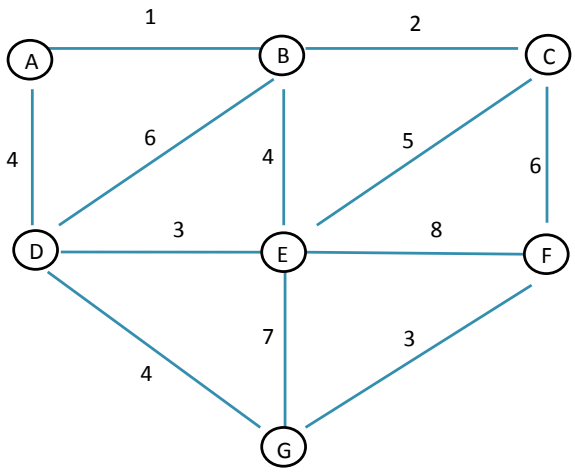
- Finds a minimum spanning tree
- Connected, weighted (possibly neg.), undirected graph
- Greedy algorithm
- Basic algorithm:
  - Initialize MST with randomly chosen vertex
  - Find minimum weight edge that connects MST to vertices not yet in MST
  - Add this edge/vertex to the MST

Prim's Algorithm

Data structures:

- Priority queue of *nodes* ordered by least-cost edge found so far joining node to MST
- Parent[v]=u where u is the vertex that added v to MST

Graph



Graph