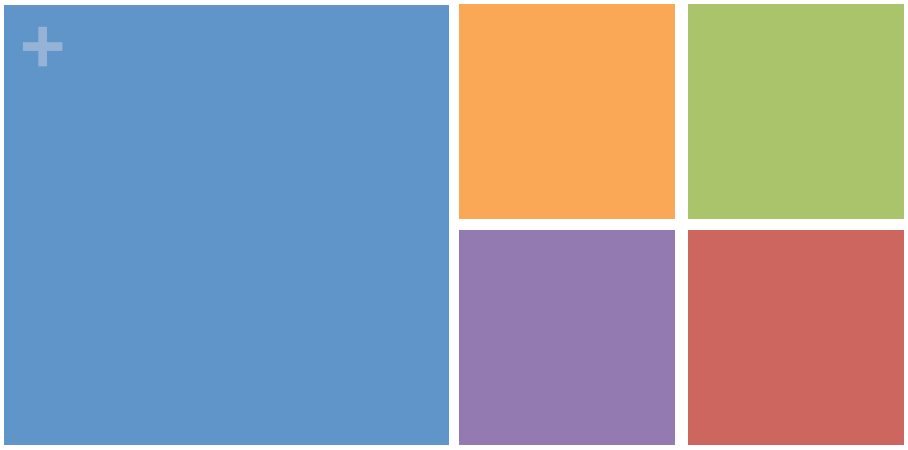


Lecture 40: Graphs

Today

- Reading
  - JS Chapter 16
- Objectives
  - Graph algorithm running times
  - Minimum Spanning Trees
- Announcements
  - Final exam study guide posted on Piazza
  - Apply to be a mentor next year!

Breadth-first Search

```
1  enqueue start node
2  while(!q.empty()) {
3     dequeue node
4     if(node not visited) {
5        mark as visited
6       for(adjacent nodes) {
7         if(adjacent node not visited)
8           enqueue adjacent node
9      }
10    }
11 }
```

What's the running time of this algorithm in terms of |V| and |E|?

Dijkstra's Algorithm

```
while (!frontier.is_empty()) {

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

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

  } // end while
```

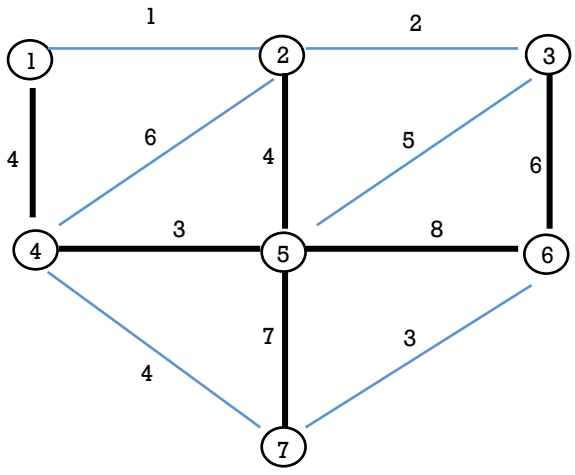
What's the running time of this algorithm in terms of |V| and |E|?

Minimum Spanning Trees

- G' = (V', E') is a subgraph of G=(V,E) if 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 minimum weight spanning tree
  - Weight is the sum of the weights of the edges in the MST

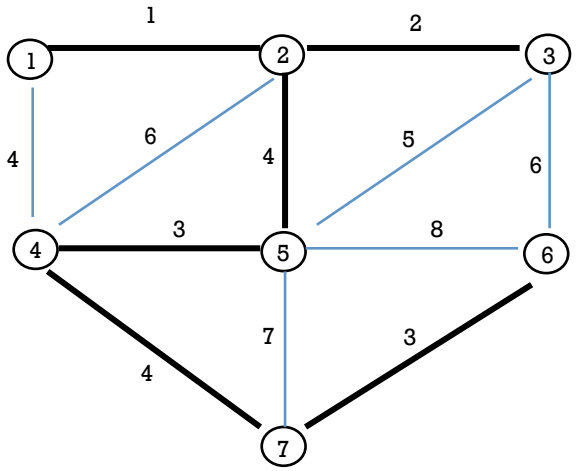
Minimum Spanning Tree

weight = 32



Minimum Spanning Tree

weight = 17



Prim's Algorithm

- Algorithm for finding a minimum spanning tree
- Runs on a connected, weighted (possibly negative), undirected graph
- Greedy algorithm – makes the greedy choice each time
- Basic algorithm:
  - Initialize tree with randomly chosen vertex
  - Find minimum weight edge that connects tree to vertices not yet in tree
  - Add this edge/vertex to the tree

Prim's Algorithm

- Data structures:
- For each vertex v, key[v] is least-cost edge (found so far) joining v to tree
- For each vertex v, parent[v] is vertex u in edge(u,v) that added v to the tree
- Q is priority queue ordered by least-cost edges (i.e. by key)

Prim's Algorithm

```
prim(g) {
   pick start node r
   foreach(u in V-{r}) key[u] = ∞
   key[r] = 0; parent[r] = null;
   add all vertices to Q (by key)
   while(!Q.empty())
     u = min node from Q
     foreach v adjacent to u
     if v in Q and edge_weight(u,v) < key[v]
     parent[v] = u; key[v] = edge_weight(u,v)
     adjust priority of v in Q
 return parent
}
```