Minimum Spanning Trees

Given an undirected graph **G**=(**V**,**E**), find a graph **G'**=(**V**, **E'**) such that:

- E' is a subset of E
- |E'| = |V| - 1
- G' is connected

G' is a minimum spanning tree.

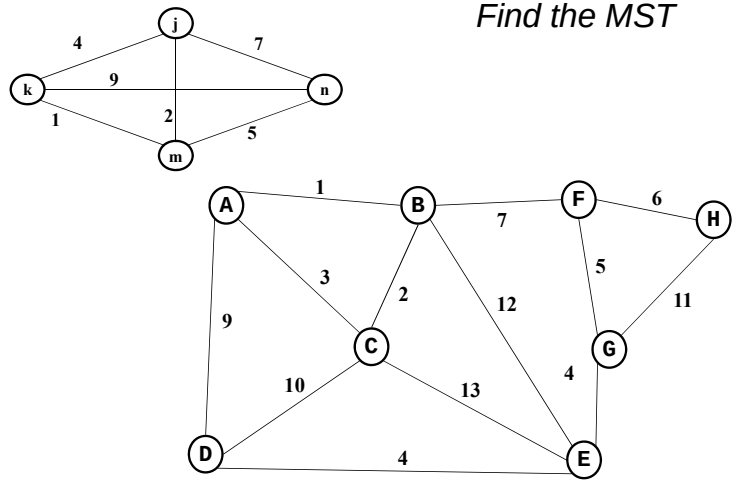
- $\sum_{(u,v)\in E'} C_{uv}$ is minimal

Applications:

- Example: Electrical wiring for a house or clock wires on a chip
- Example: A road network if you cared about asphalt cost rather than travel time

Student Activity

Find the MST

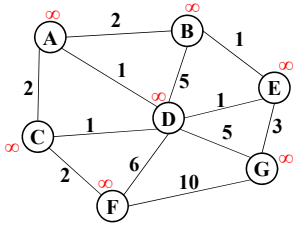


Prim's Algorithm for MST

1. For each node **v**, set **`v.cost = ∞`** and **`v.known = false`**
2. Choose any node **`v.`** (this is like your "start" vertex in Dijkstra)
   a) Mark **v** as known
   b) For each edge **`(v,u)`** with weight **w**:
      set **`u.cost=w`** and **`u.prev=v`**
3. While there are unknown nodes in the graph
   a) Select the unknown node **v** with lowest cost
   b) Mark **v** as known and add **`(v, v.prev)`** to output (the MST)
   c) For each edge **`(v,u)`** with weight **w**,

```
if(w < u.cost) {
  u.cost = w;
  u.prev = v;
}
```


Example: Find MST using Prim's



Order added to known set:

vertex	known?	cost	prev
A			
B			
C			
D			
E			
F			
G			

Kruskal's Algorithm for MST

An *edge-based* greedy algorithm

Builds MST by greedily adding edges

1. Initialize with
   - empty MST
   - all vertices marked unconnected
   - all edges unmarked
2. While all vertices are not connected
   a. Pick the lowest cost edge `(u,v)` and mark it
   b. If **u** and **v** are not already connected, add `(u,v)` to the MST and mark `u` and `v` as connected to each other

Aside: Union-Find aka Disjoint Set ADT

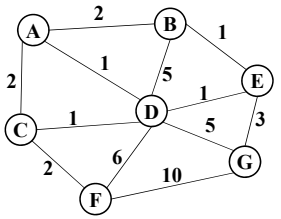
- **Union(x,y)** – take the union of two sets named x and y
  - Given sets: {3,5,7} , {4,2,8}, {9}, {1,6}
  - **Union(5,1)**
    Result: {3,5,7,1,6}, {4,2,8}, {9},
    To perform the union operation, we replace sets x and y by $(x \cup y)$
- **Find(x)** – return the name of the set containing x.
  - Given sets: {3,5,7,1,6}, {4,2,8}, {9},
  - **Find(1)** returns 5
  - **Find(4)** returns 8
- We can do Union in constant time.
- We can get Find to be ***amortized*** constant time
  (worst case O(log n) for an individual Find operation).

Kruskal's pseudo code

```
void Graph::kruskal(){
  int edgesAccepted = 0;
  DisjSet s(NUM_VERTICES);

  while (edgesAccepted < NUM_VERTICES – 1){
    e = smallest weight edge not deleted yet;
    // edge e = (u, v)
    uset = s.find(u);
    vset = s.find(v);
    if (uset != vset){
      edgesAccepted++;
      s.unionSets(uset, vset);
    }
  }
}
```


Example: Find MST using Kruskal's



Edges in sorted order:
1: (A,D), (C,D), (B,E), (D,E)
2: (A,B), (C,F), (A,C)
3: (E,G)
5: (D,G), (B,D)
6: (D,F)
10: (F,G)

Output:

Note: At each step, the union/find sets are the trees in the forest