Lecture 36: Graphs IV

CS 62

Fall 2018

Alexandra Papoutsaki & William Devanny

Spanning Trees

- A spanning tree `T` of a graph `G` is a subset of the edges of `G` such that:
  - `T` contains no cycles and
  - Every vertex in `G` is connected to every other vertex using just the edges in `T`
- An unconnected graph has no spanning trees.
- A connected graph will have at least one spanning tree; it may have many

Minimum Spanning Trees

- A weighted graph is a graph that has a weight associated with each edge.
- If `G` is a weighted graph, the cost of a tree is the sum of the costs (weights) of its edges.
- A tree `T` is a minimum spanning tree of `G` iff:
  - it is a spanning tree and
  - there is no other spanning tree whose cost is lower than that of `T`.

Minimum Spanning Trees

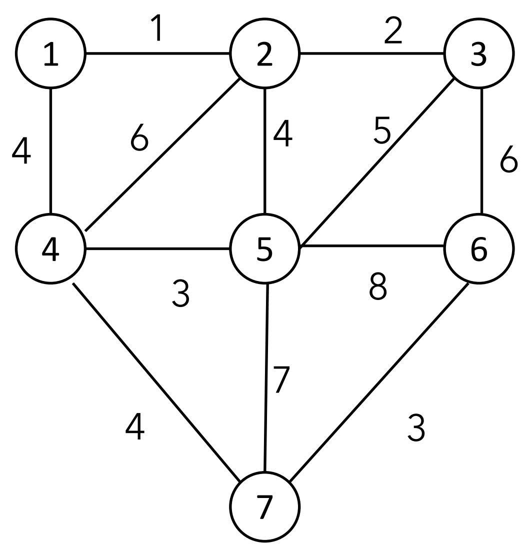
- Application:
  - The cheapest way to lay cable that connects a set of points is along a minimum spanning tree that connects those points.
- Many algorithms exist to find minimum spanning trees, most run in $O(m \log m)$ time.
- In 1995 Karger, Klein & Tarjan found a linear time randomized algorithm, but there is no known linear time deterministic algorithm

Kruskal’s Algorithm

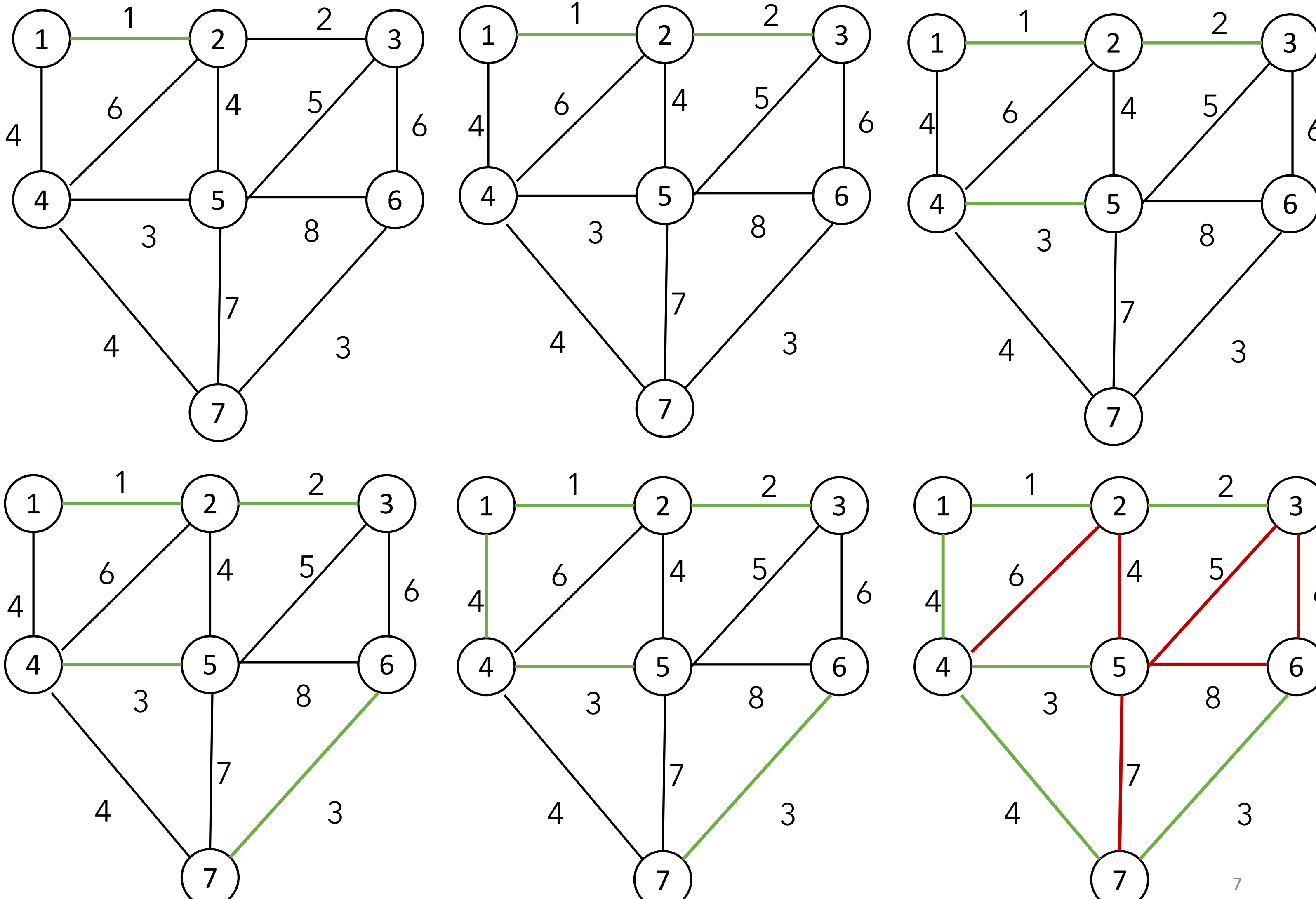
- Create forest F with no edges, using vertices in V
- Sort the edges in the graph by their weight (smallest to largest)
- For each edge e in sorted order:
  - if e connects two different trees in F , then add e to F

Kruskal on sample graph

(1,2):1
(2,3):2
(4,5):3
(6,7):3
(1,4):4
(2,5):4
(4,7):4
(3,5):5
(2,4):6
(3,6):6
(5,7):7
(5,6):8



(1,2):1
(2,3):2
(4,5):3
(6,7):3
(1,4):4
(2,5):4
(4,7):4
(3,5):5
(2,4):6
(3,6):6
(5,7):7
(5,6):8



Kruskal's Algorithm pseudocode

```
A = {};
for(every vertex v in V) {
    make-set(v)
    for(every edge (u, v) ordered by increasing weight) {
        if(find (u) != find (v)) {
            A.add((u, v));
            union(u, v);
    }
}
return A;
```

`make-set(v)` - makes a set from a single vertex v
`find(v)` - finds the set that v belongs to
`union(u, v)` - makes the union of the sets containing u and v

} Union-find structure

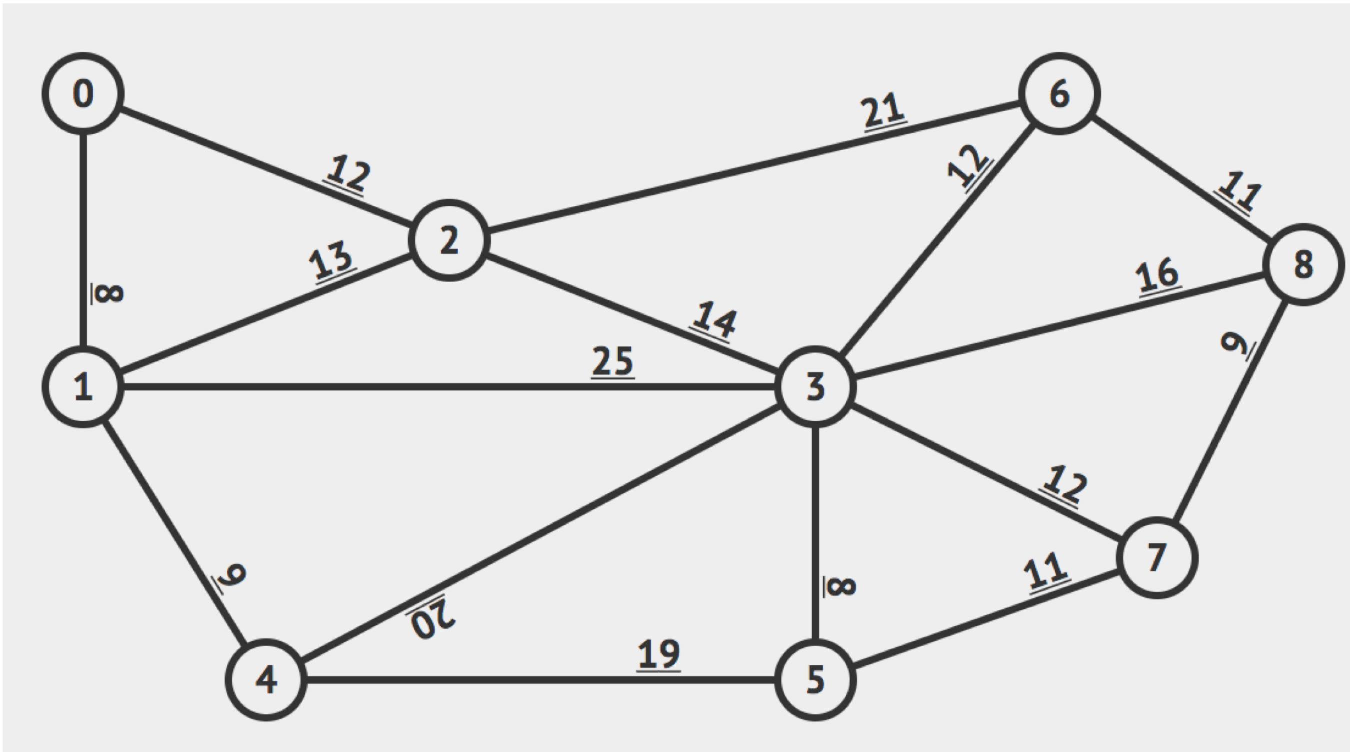
Union-Find Data Structure

keeps track of a set of elements partitioned into a number of disjoint subsets

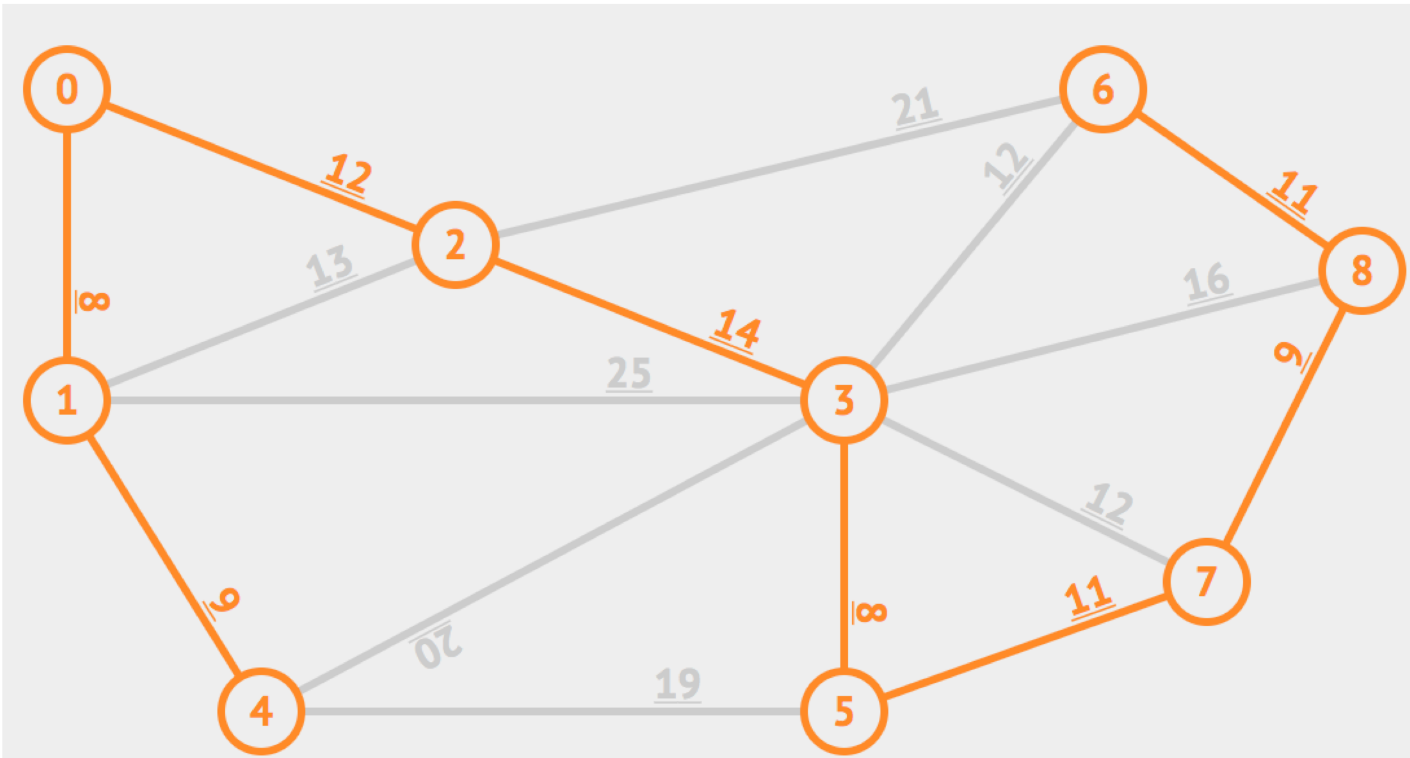
Find: Find what subset an element belongs. Use to find if two elements belong in the same subset

Union: Create a single subset out of two subsets

Practice Time



Answer



Graph Algorithms

- Very important in practice!
- Sophisticated data structures
- Careful analysis of correctness and complexity
- CS 140: Algorithms