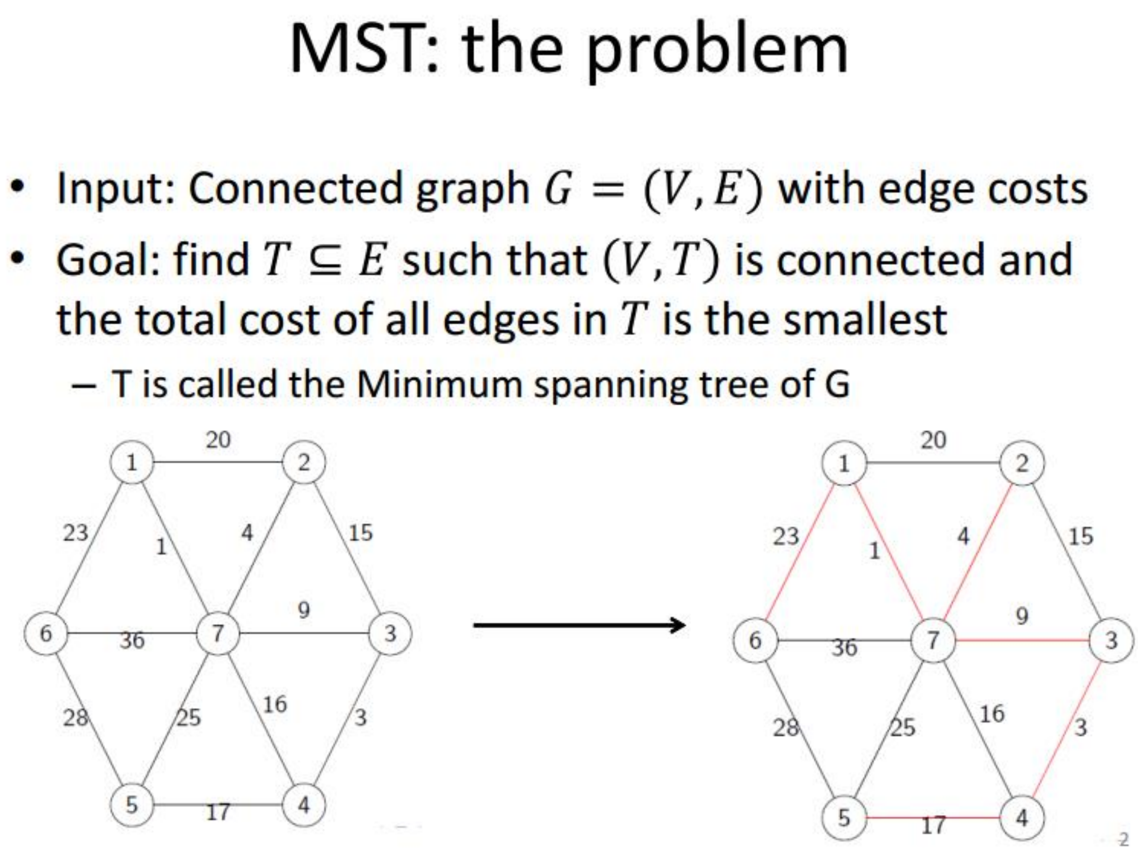
<u>Week 7 Day 1 Tuesday Lecture Notes</u>

Prep:

- Implementation 2 is due Thursday.
- Read DPV Chapter 5.1: Minimum Spanning Tree.
- Review Greedy Algorithm/Interval Scheduling Slides
- Midterm Stats: Median 36 Points. 54 Points Max

Minimum Spanning Trees:



- V: Set of all vertices in the graph
- E: Denotes all edges in the graph. (Each edge has a cost)
- Assume that the whole graph is connected.

Goal: Find a subset of edges that form a tree so that V-T are connected and the cost is minimum.

Minimum Cost Structure will be a Spanning Tree and have no Cycle.

Practical Applications of MST:

- Network Design → Trying to connect nodes in the cheapest way possible.
- Must be a Tree otherwise we can remove edges and still be fully connected and cheaper.
- Machine Learning Clustering.

Greedy Template:

Given G = (V,E) and E has a cost attached.

Start with an empty tree T (T will store edges of a MST)

While T is not fully connected

Select an Edge to add per the greedy criteria.

Return T

How do we use the greedy criteria to determine what edge to choose?

Kruskal's Algorithm: Start with Cheapest Edges nodes and add cheapest connecting edges that don't form a cycle.

Order the edges based on cost. And add an edge T, but don't form a cycle, until T is complete.

See Picture of Kruskal's Algorithm Final Picture* (page 10)

If the edge costs are the same, you can choose one Arbitrarily. Then you must check if it will form a cycle.

Prim's Algorithm: Start with arbitrary node. Add cheapest connecting edges that don't form a cycle.

Instead of starting with the cheapest edge we choose an arbitrary node.

In each iteration, pick an edge with Least Attachment cost to T.

Will end up with the exact same Tree, but might start with a different position.

Correctness: We assume that all edge costs are distinct. As such there is a unique MST for each graph.

- In the future we will not require all edge costs to be unique.

Cut Property:

In English:

- If we Break the MST Graph into two parts. The cheapest Edge between any of the two parts is guaranteed to be part of the Minimum Spanning Tree.
- Any partition separation the cheapest edge crossing the two parts will be part of the MST.

Now let's talk in Math:

Let $S \subset V (\neq \emptyset$ and $\neq V)$. Let $e = (v, w)$ be the minimum cost edge with one end in S and the other end in $V \setminus S$. Then e must be in the MST.

Proof by Contradiction: Assume the statement is false and then arrive at a conflict which is not possible.

[Not True] → (TRUE) → (TRUE) →…→ Conflict.

Contradiction might create a Cycle and Lose Full Connectivity. Thus you cannot simply replace any edge with a cheaper edge and this proof

Definition in English: With two partitions V and W. and a sub-optimal cross between the two if we find a more optimal cross we can replace it with sub-optimal cross and still maintain a complete graph with no cycles. This proves that T is not a minimum spanning true, end of proof.

Prims Algorithm Correctness:

Because in the definition of Prim's state that we select the cheapest edge across to the rest of the graph. The definition of Prims Proves the correctness of Kruskal's Algorithm.

Every edge that we add is obeying the Cut Property and will be part of the MST.

Kruskal's Algorithm Correctness:

Starts with every node in its own connected component.

Each additional will connect two parts. When edge e is selected it must be the cheapest edge connecting S(the connected component) to the rest of the graph. Thus the Cut Property Holds true.

Take any connecting component at one end and create a cut.

Every edge we have must belong to this unique MST.

What is the Edge Costs are not distinct? (Thought for next class)

How do we break ties?

Next time:

- Midterm returned (Median 36)
- Arguments over Midterm grades must be addressed by Friday.
- Read DPV Chapter 5.2
- Implementation 2 Due Thursday at Midnight.

<u>End of Week 7 Day 1 Thursday Lecture Notes</u>