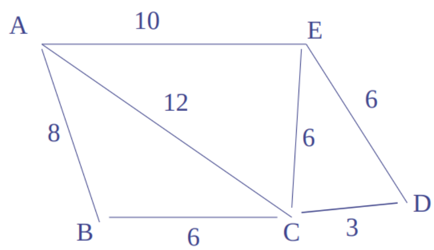
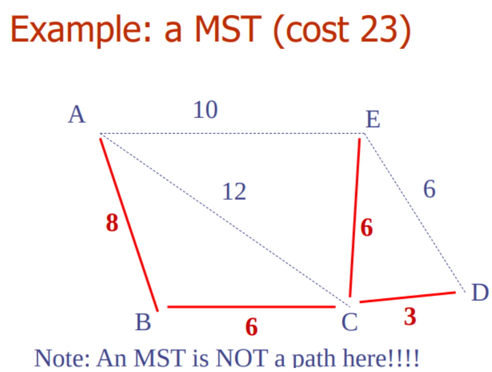
1 Spanning tree vs Minumum spanning tree

- Spanning Tree
  - Input: connected, undirected graph
  - Output: a tree which connects all vertices in the graph using only the edges present in the graph
- Minimum Spanning Tree
  - Input: connected, undirected, weighted graph
  - Output: a spanning tree
    - connects all vertices in the graph using only the edges present in the graph without any cycles and with the minimum possible total edge weight.
    - is minimum in the sense that the sum of weights of the edges is the smallest possible for any spanning tree
- The main difference is that shortest path looks for a lowest cost distance between two nodes, where as **MST** looks for a shortest path that would go trough all of given vertices





2 Why MST is a tree

We really want a minimum spanning **sub-graph** ,that is, a subset of the edges that is connected and that contains **every node**. (Assuming all weights are non-negative) If the graph has a cycle then we can remove an edge of the cycle, and the graph will still be connected, and will have a smaller weight If a graph is **connected** and **acyclic** then it is a tree

3 Prim's algorithm for constructing MST

- Start by picking any vertex **M**
- hoose the shortest edge from M to any other vertex N
- Add edge (M,N) to the MST
- Loop:
  - Continue to add at every step a shortest edge from a vertex in MST to a vertex outside, until all vertices are in MST
  - (If there are multiple shortest edges, then can take any arbitrary one)

3.1 Greedy algorithm

- Prims algorithm for constructing a Minimal Spanning Tree is a **greedy algorithm :**
- it just adds a minimum weight edge
- without worrying about the overall structure, without looking ahead.
- It makes a locally optimal choice at each step

4 Proof of MST being optimal

- Let `G` be a weighted connected graph
- let `V1` and `V2` be a partition of the vertices of `G` into two **disjoint non-empty** sets.
- Furthermore, let `e` be an edge with **minimum weight** from among those with one endpoint in `V1` and the other in `V2`.
- There is an MST that has e as one of its edges.

4.1 Justification

- Suppose our MST `T1` does not include `e`
- We add `e` to `T1`, which creates a cycle
- That means that tree `T1` must have some edge `e1` that connects `V1` and `V2`,
- if we remove `e1` we now have a tree `T2` that has lower cost, which means `T1` was not MST

Reference section

placeholder