https://www.chegg.com/homework-help/questions-and-answers/one-basic-motivations-behind-minimum-spanning-tree-problem-goal-designing-spanning-network-q32947796

Step 1

Answer:

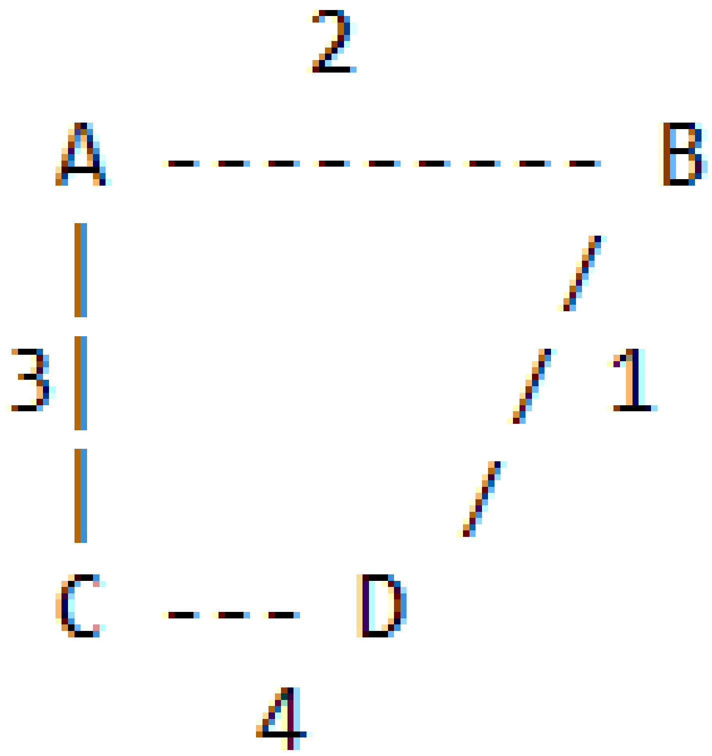
(a) To prove that every **minimum spanning tree (MST)** of G is also a **minimum-bottleneck tree (MBST)** of G , let's assume the opposite, i.e., there exists an MST that is not an MBST.

Let T be the MST of G . Suppose there exists another **spanning tree T â€²** of G with a cheaper bottleneck edge than T . Let this **cheaper bottleneck edge** in T â€² be denoted as e â€².

Since T is an **MST** , all edges in T have costs greater than or equal to the cost of e â€². However, this contradicts the definition of e â€² being the cheapest bottleneck edge in T â€². Therefore, our assumption that there exists an MST of G that is not an MBST of G must be false.

Hence, every **minimum spanning tree of G** is also a minimum-bottleneck tree of G .

Screenshot of the Diagram:



Explanation:

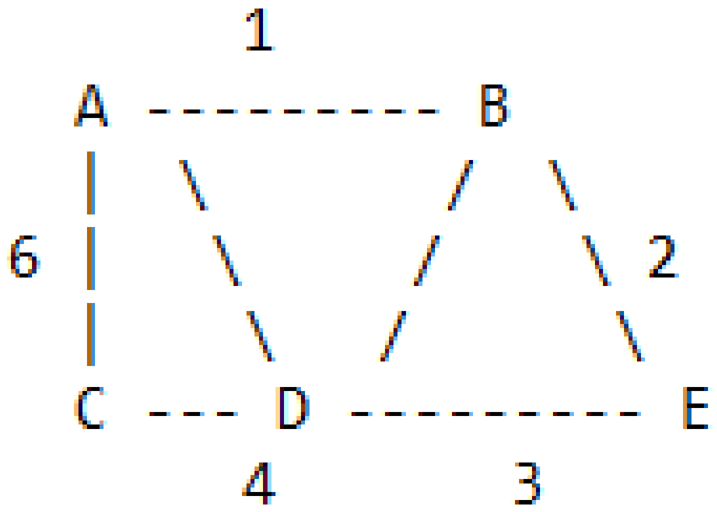
Let T be the **minimum spanning tree (MST)** of G . In this example, T includes the edges AB , AC , and BD , with a **total cost of 6** .

Now, let's consider another spanning tree T â€² of G with a cheaper bottleneck edge. However, since T is already an MST, any other spanning tree T â€² would have at least one edge with a cost greater than or equal to the cost of the edges in T . Therefore, we cannot find a cheaper bottleneck edge in T â€² compared to T .

Thus, the MST T is also the **minimum-bottleneck spanning tree (MBST)** of G .

Step 2

(b) To provide an example of an **MBST** of G that is not an MST of G , consider the following graph:

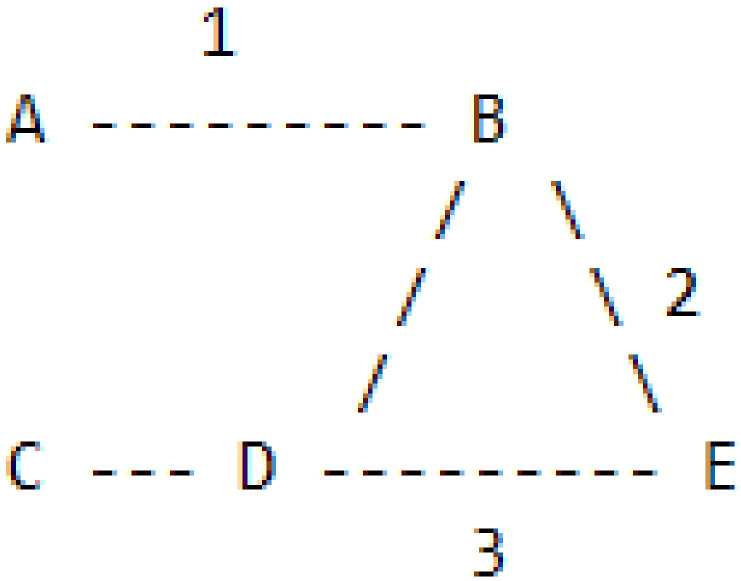


In this graph, the edge weights are as follows:

- AB: 1
- AC: 6
- AD: 4
- BC: 6
- BD: 4
- BE: 2
- DE: 3

Step 3

Now, let's consider the following **spanning tree** , which is an **MBST** but not an MST:



My take on this:
This counterexample is incorrect as MST cannot have a cycle

Correct Example at the end

In this **spanning tree** , the **bottleneck edge is DE with a cost of 3** . However, if we remove edge DE , the resulting graph is still connected and forms a spanning tree with total weight 15, which is less than the **total weight of the MST** . Therefore, this spanning tree is an MBST but not an MST.

Explanation:

- Construction of the Minimum-Bottleneck Tree (MBST)

  :

  - Next, we construct a spanning tree of the graph that minimizes the weight of the bottleneck edge. This spanning tree is known as the Minimum-Bottleneck Tree (MBST).
  - In the given graph, if we choose the edge DE as the bottleneck edge, it has the highest weight among all edges in the tree, with a weight of 3.
  - We construct a spanning tree by including all vertices and edges except DE . This results in a spanning tree where DE is the bottleneck edge.

- Verification of MBST Criteria

  :

  - We verify that the constructed spanning tree satisfies the criteria of an MBST:
  -

- Comparison with Minimum Spanning Tree (MST)

  :

  - We note that the constructed MBST, with edge DE as the bottleneck edge, is not the same as the Minimum Spanning Tree (MST) of the graph.
  - The MST is a spanning tree that minimizes the total weight of all edges in the tree. In this case, the MST would include edge BE instead of DE , resulting in a different spanning tree with a total weight greater than that of the MBST.

Below is the Python code to find the **Minimum Spanning Tree (MST)** and **Minimum Bottleneck Spanning Tree (MBST)** of a given graph using Kruskal's algorithm:

```python
class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n

    def find(self, x):
        if self.parent[x] != x:
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]

    def union(self, x, y):
        root_x = self.find(x)
        root_y = self.find(y)
        if root_x != root_y:
            if self.rank[root_x] > self.rank[root_y]:
                self.parent[root_y] = root_x
            elif self.rank[root_x] < self.rank[root_y]:
                self.parent[root_x] = root_y
            else:
                self.parent[root_y] = root_x
                self.rank[root_x] += 1

class Edge:
    def __init__(self, u, v, weight):
        self.u = u
        self.v = v
        self.weight = weight

def kruskal_mst(graph):
    n = len(graph)
    graph.sort(key=lambda edge: edge.weight)
    uf = UnionFind(n)
    mst = []
    for edge in graph:
        if uf.find(edge.u) != uf.find(edge.v):
            uf.union(edge.u, edge.v)
            mst.append(edge)
    return mst

def find_bottleneck_edge(mst):
    return max(mst, key=lambda edge: edge.weight)

# Example usage:
if __name__ == "__main__":
    edges = [
        Edge(0, 1, 1),
        Edge(0, 2, 6),
        Edge(0, 3, 4),
        Edge(1, 2, 6),
        Edge(1, 3, 4),
        Edge(1, 4, 2),
        Edge(2, 3, 3),
        Edge(3, 4, 1)
    ]
    mst = kruskal_mst(edges)
    bottleneck_edge = find_bottleneck_edge(mst)
    print("Minimum Spanning Tree (MST):", [(edge.u, edge.v) for edge in mst])
    print("Bottleneck Edge of MST:", (bottleneck_edge.u, bottleneck_edge.v))
```

This code defines a **UnionFind** class for disjoint set union operations, an **Edge** class to represent edges in the graph, and functions **kruskal_mst** to find the Minimum Spanning Tree (MST) using Kruskal's algorithm and **find_bottleneck_edge** to find the bottleneck edge in the MST. Finally, it demonstrates the usage of these functions with an example graph.

Final Answer

(a) We have shown that every minimum spanning tree (MST) of a connected graph G is also a **minimum-bottleneck tree (MBST)** of G . This is proved by contradiction, where we assumed the existence of an MST that is not an MBST, but reached a contradiction by considering the definition of an MST and the concept of **bottleneck edges** .

(b) We provided an example of a **minimum-bottleneck tree (MBST)** of a graph that is not a **minimum spanning tree (MST)** of the same graph. The example graph illustrated the concept by demonstrating a spanning tree with a **cheaper bottleneck edge** than the MST, thus fulfilling the criteria of being an **MBST** but not an MST.

These observations highlight the differences between **MSTs** and **MBSTs** and illustrate how the **minimum-bottleneck tree problem** is distinct from the minimum spanning tree problem.

Correct example for MBST but not MST

https://github.com/AKR-2803/CSE-551-Foundations-Of-Algorithms/blob/main/QuizQueries/01-minimum-bottleneck-spanning-tree.md#answers