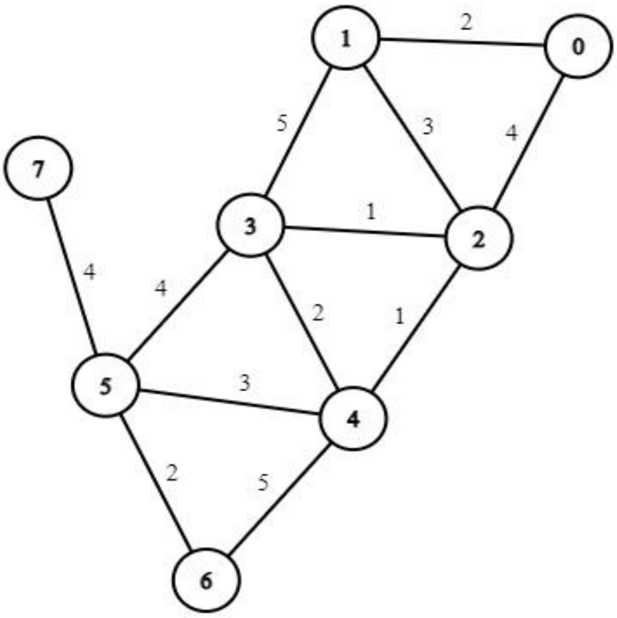
Manual Executions



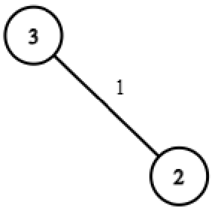
The input file:

```
8 12

0 1 2
0 2 4
1 2 3
1 3 5
2 3 1
2 4 1
3 4 2
3 5 4
4 5 3
4 6 5
5 6 2
5 7 4
```

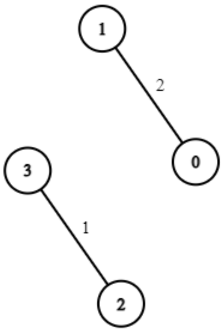
1. Initialize the variables:
    - minimum_cost = 0
    - count = 0
    - tree = [0, 1, 2, 3, 4, 5, 6, 7]
    - minimum_tree = []
2. Sort edges by cost:
    - ordered_edges = ((2, 3), 1), ((0, 1), 2), ((3, 4), 2), ((4, 5), 2), ((1, 2), 3), ((4, 6), 3), ((5, 7), 4), ((0, 2), 4), ((3, 5), 4), ((1, 3), 5), ((2, 4), 6)
3. Process the first edge ((2, 3), 1) with cost 1:
    - edge = ((2, 3), 1)
    - v = tree[2] = 2
    - w = tree[3] = 3
    - Since v != w and ((3, 2), 1) is not in minimum_tree:
        - minimum_cost += 1 (add the cost of the edge)

        - count += 1
        - minimum_tree.append((2, 3))
        - Update the tree:



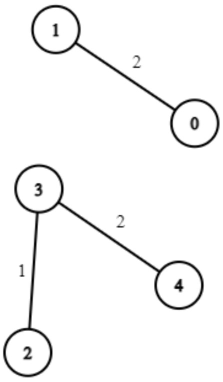
    - Increment x to 1

4. Process the second edge ((0, 1), 2) with cost 2:
    - edge = ((0, 1), 2)
    - v = tree[0] = 0
    - w = tree[1] = 1
    - Since v != w and ((1, 0), 2) is not in minimum_tree:
        - minimum_cost += 2 (add the cost of the edge)
        - count += 1
        - minimum_tree.append((0, 1))
        - Update the tree:



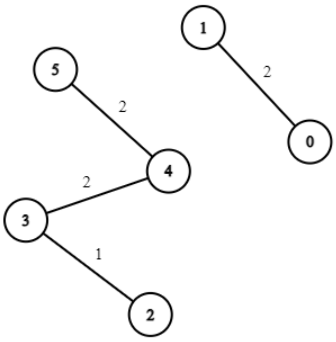
    - Increment x to 2

5. Process the third edge ((3, 4), 2) with cost 2:
    - edge = ((3, 4), 2)
    - v = tree[3] = 2
    - w = tree[4] = 4
    - Since v != w and ((4, 3), 2) is not in minimum_tree:
        - minimum_cost += 2 (add the cost of the edge)
        - count += 1
        - minimum_tree.append((3, 4))
        - Update the tree:



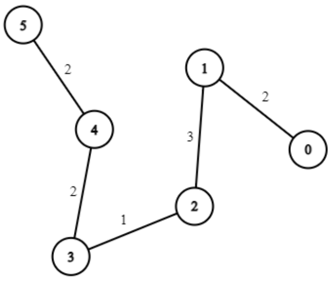
- Increment x to 3

6. Process the fourth edge ((4, 5), 2) with cost 2:
    - edge = ((4, 5), 2)
    - v = tree[4] = 2
    - w = tree[5] = 5
    - Since v != w and ((5, 4), 2) is not in minimum_tree:
        - minimum_cost += 2 (add the cost of the edge)
        - count += 1
        - minimum_tree.append((4, 5))
        - Update the tree:



    - Increment x to 4

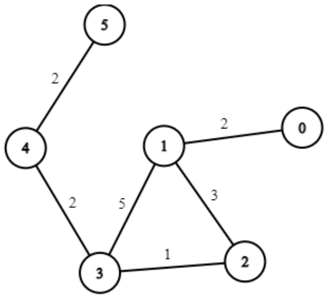
7. Process the fifth edge ((1, 2), 3) with cost 3:
    - edge = ((1, 2), 3)
    - v = tree[1] = 0
    - w = tree[2] = 2
    - Since v != w and ((2, 1), 3) is not in minimum_tree:
        - minimum_cost += 3 (add the cost of the edge)
        - count += 1
        - minimum_tree.append((1, 2))
        - Update the tree:



    - Increment x to 5

8. Process the tenth edge ((1, 3), 5) with cost 5:
    - Since v != w and ((3, 1), 5) is not in minimum_tree:

        - minimum_cost += 5 (add the cost of the edge)
        - count += 1
        - minimum_tree.append((1, 3))
        - Update the tree:

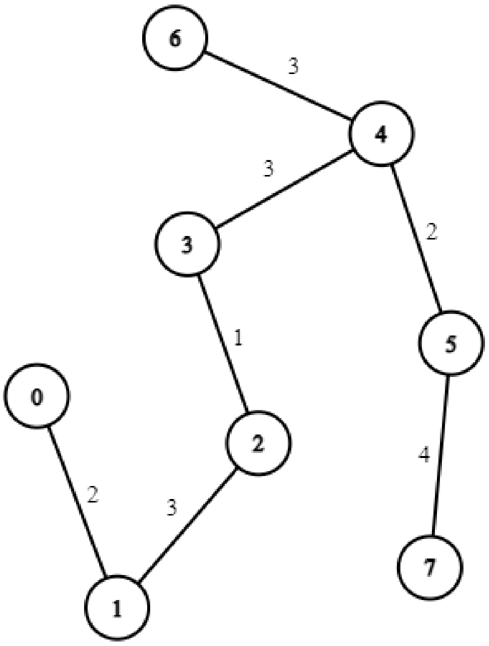


    - Increment x to 10

9. Process the eleventh edge ((2, 4), 6) with cost 6:
    - edge = ((2, 4), 6)
    - v = tree[2] = 0
    - w = tree[4] = 2
    - Since v == w, it forms a cycle and is not included in minimum_tree.
10. End of the loop:
    - Since x = 11 and count = 7 (equal to the number of vertices - 1), the loop terminates.
11. Return the minimum cost:

    The minimum cost: 17

    The minimum cost tree:



The list of edges:

2 3 1

0 1 2

3 4 3

4 5 2

1 2 3

4 6 3

5 7 4