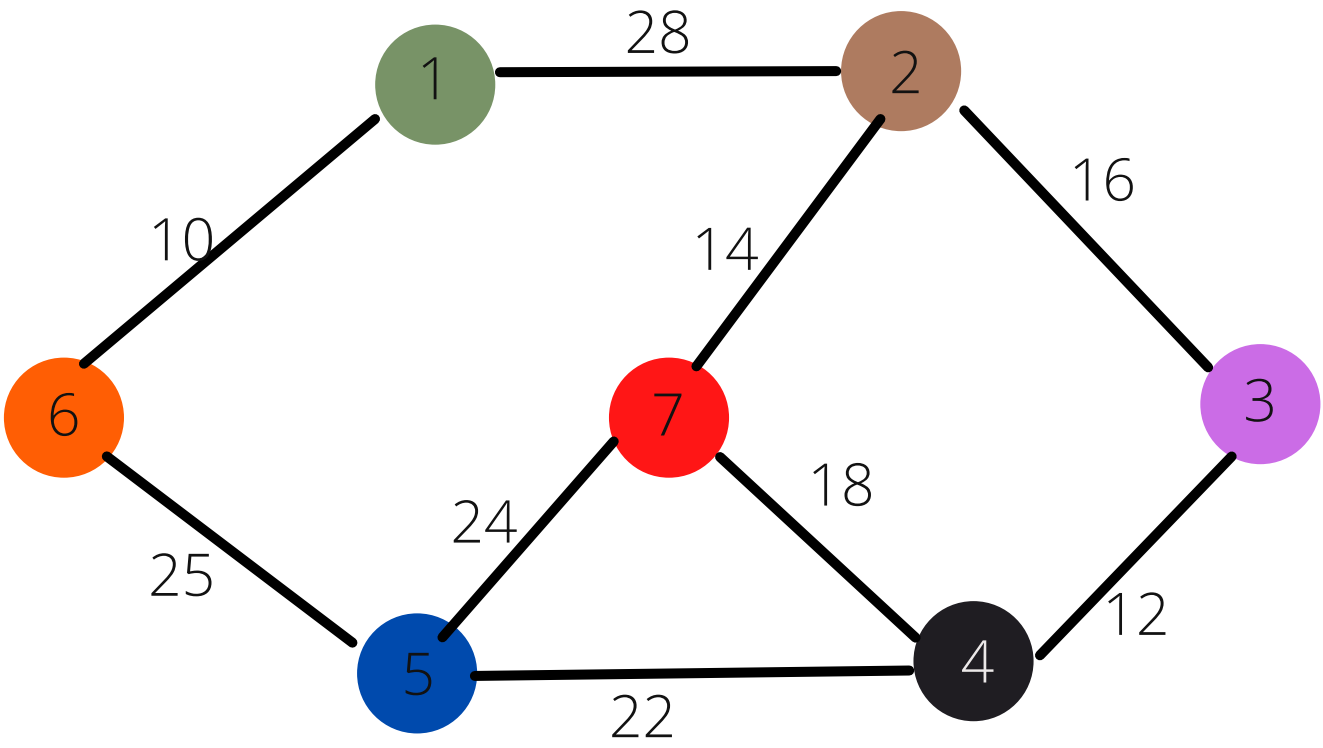
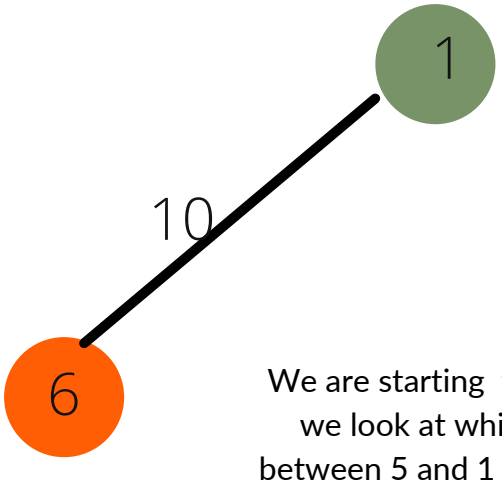
HANDLING MINIMUM SPANNING TREES WITH PRIM'S ALGORITHMS

BY HEATHER MATARUSE

PROCESS



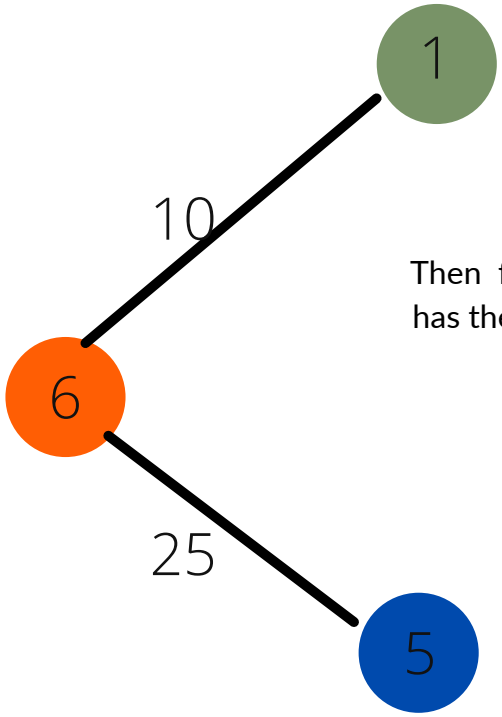
STEP 1



We are starting from 6 -> 1: 10 because when we look at which side has the least weight between 5 and 1 its the 1 which has a weight of 10

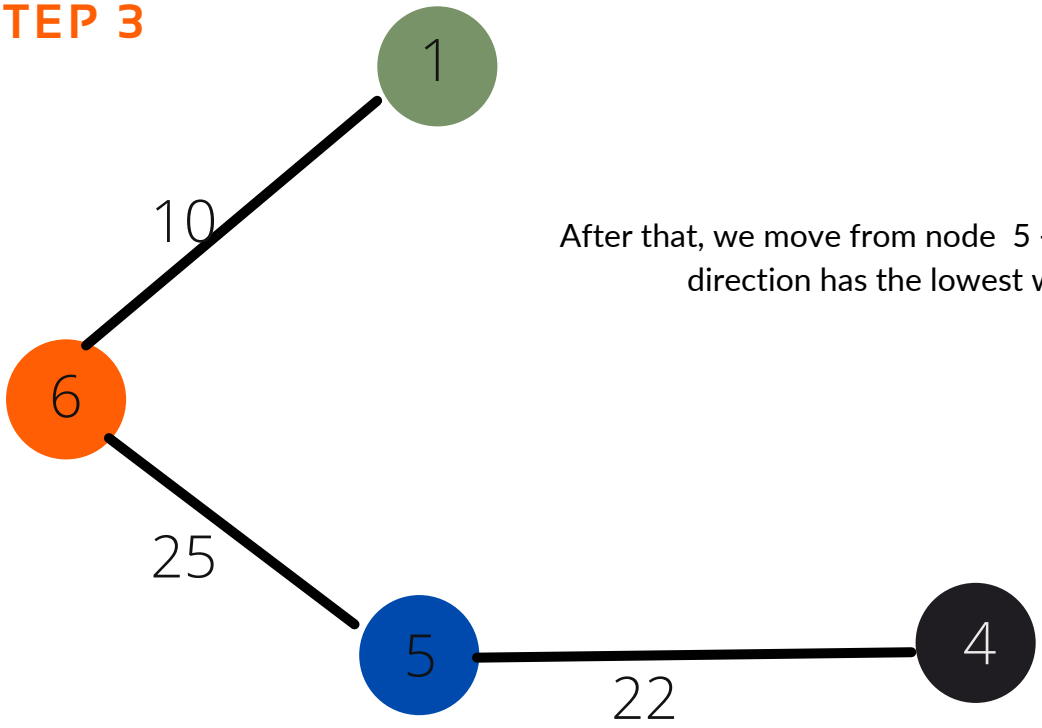
PROCESS

STEP 2



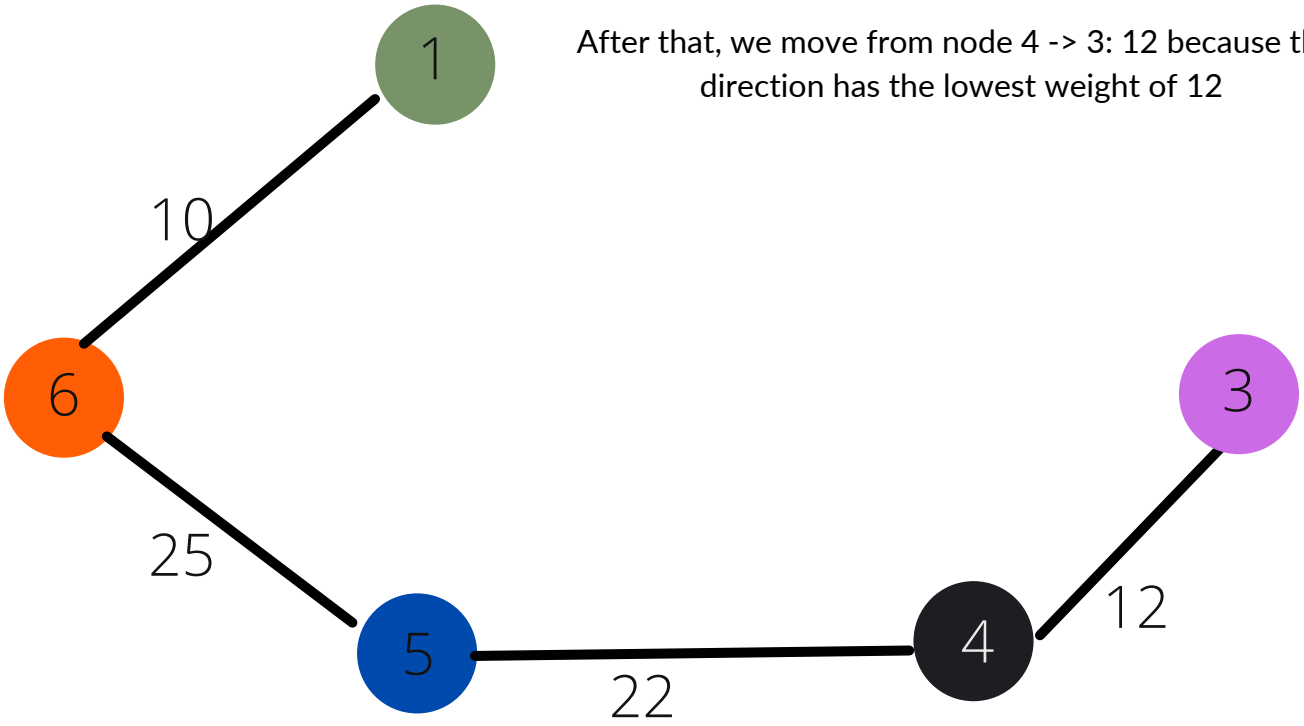
Then from there, we take route 6 -> 5: 25 because node 5 has the weight of 25 and again it is connected to 6 which I have chosen to be the root

STEP 3



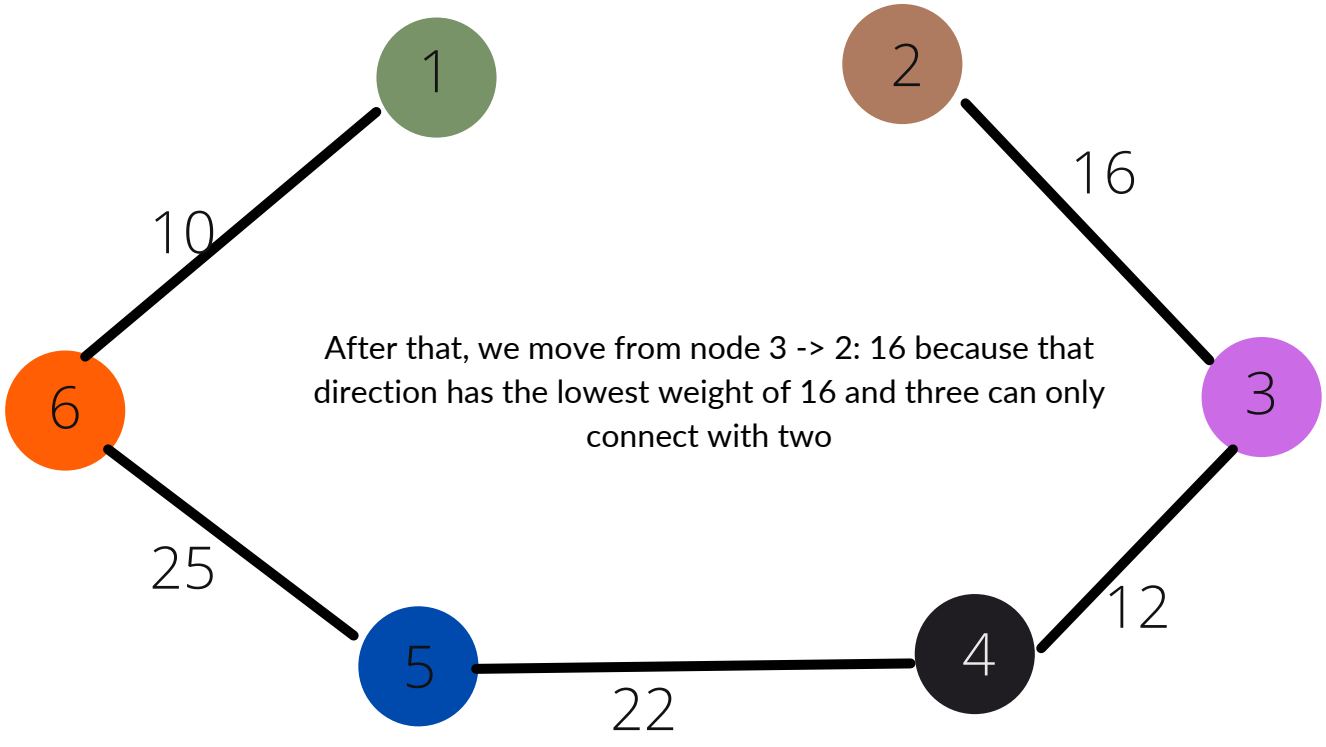
After that, we move from node 5 -> 4: 22 because that direction has the lowest weight of 22

STEP 4



After that, we move from node 4 -> 3: 12 because that direction has the lowest weight of 12

STEP 5

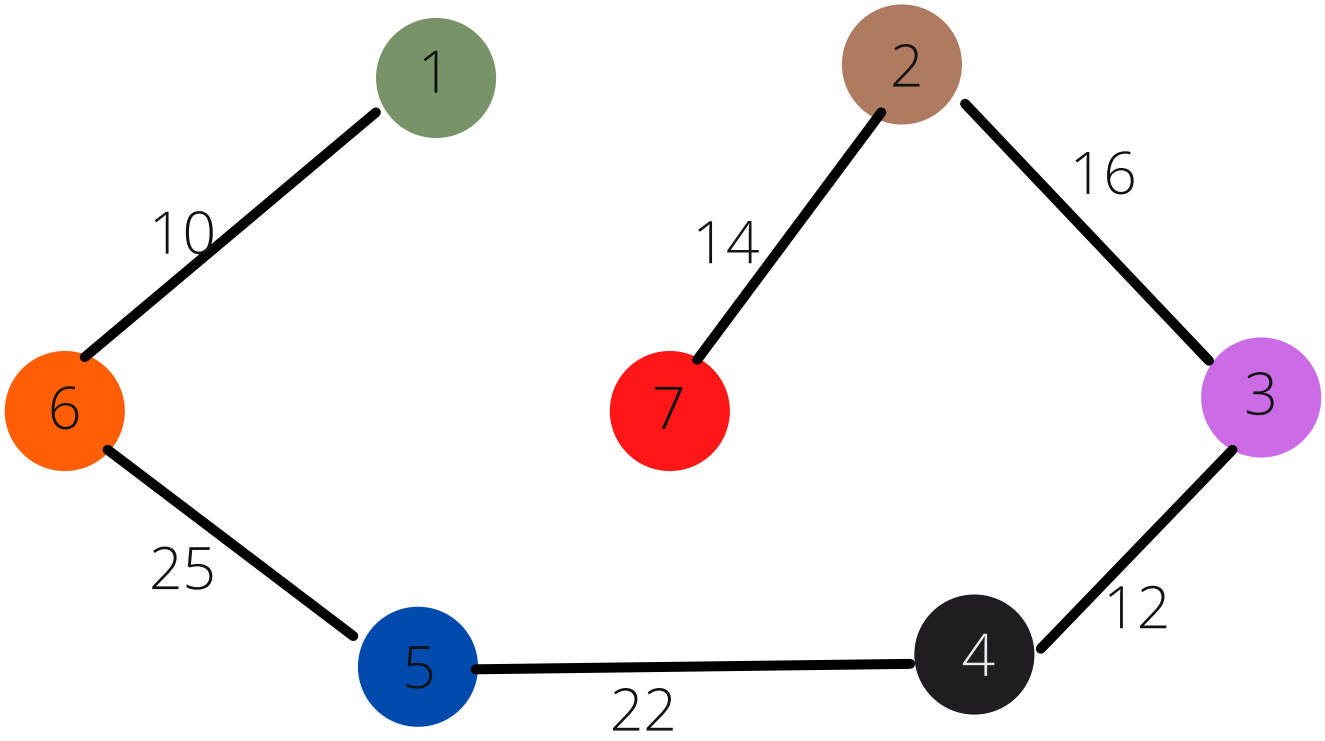


After that, we move from node 3 -> 2: 16 because that direction has the lowest weight of 16 and three can only connect with two

EXPLANATION

After that, we move from node 2-> 7: 14, because that direction has the lowest weight of 14

STEP 6



Pseudocode

- You build vertice U, which contains the list of visited vertices; The list of vertices that have not been visited is contained in the created vertice V-U.
- By connecting the least weight edge, move vertices from vertice V-U to vertice U one by one.
- We utilize a min-heap to store the vertices not yet included in the MST after traversing all the vertices in the graph using breadth-first search.
- We utilize min-heap as a priority queue to get the minimum weight edge.
- Operations on the min-heap, such as extracting the smallest element and reducing the key value, take O(log V) time.
- Therefore Cost of Minimum Spanning Tree = Sum of all edge weights = 10 + 25 + 22 + 12 + 16 + 14 = 99 units

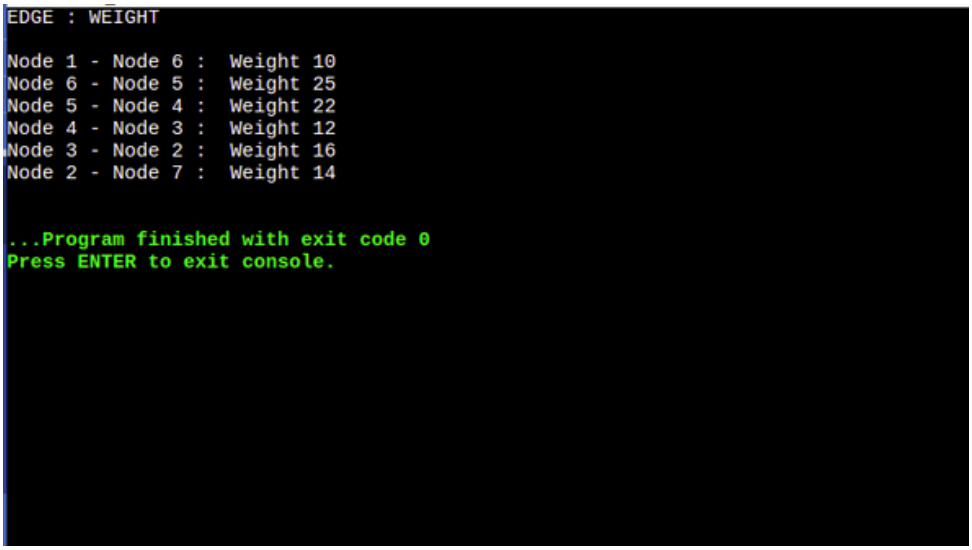
Time Complexity

- The time complexity is O(VlogV + ElogV) = O(ElogV , this is because there is use of a binary heap

Weakness of the Algorithm

- As a new edge is introduced, the list of edges must be searched from the beginning.
- If more than one edge has the same weight, all feasible spanning trees must be identified before the final minimal tree can be obtained.

Code output



Code in python

```
infinity_variable = 9999999

no_of_vertices = 7

matrix = [[0, 28, 0, 0, 0, 10, 0],
  [28, 0, 16, 0, 0, 0, 14],
  [0, 16, 0, 12, 0, 0, 0 ],
  [0, 0, 12, 0, 22, 0, 18],
  [0, 0, 0, 22, 0, 25, 24],
  [10, 0, 0, 0, 25, 0, 0],
  [0, 14, 0, 18, 24, 0, 0]]

visited_vertices = [0, 0, 0, 0, 0, 0, 0]

number_of_edges = 0

visited_vertices[0] = True

print("EDGE : WEIGHT\n")

while (number_of_edges < no_of_vertices - 1):

  mini = infinity_variable
  a = 0
  b = 0

  for k in range(no_of_vertices):

    if visited_vertices[k]:
      for i in range(no_of_vertices):
        if ((not visited_vertices[i]) and matrix[k][i]):

          if mini > matrix[k][i]:
            mini = matrix[k][i]
            a = k
            b = i

  print("Node " + str(a+1) + " -" + " Node " + str(b+1) + " :  Weight " + str(matrix[a][b]))

  visited_vertices[b] = True

  number_of_edges += 1
```