

**Objective:** Implementation and analysis of Prim's Algorithm

# Prim's Algorithm

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

## Prim's algorithm

The steps for implementing Prim's algorithm are as follows:

1. Initialize the minimum spanning tree with a vertex chosen at random.
2. Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree
3. Keep repeating step 2 until we get a minimum spanning tree.

**Code:**

```

INF, V = 9999999, 5
G = [[0, 9, 75, 0, 0],
      [9, 0, 95, 19, 42],
      [75, 95, 0, 51, 66],
      [0, 19, 51, 0, 31],
      [0, 42, 66, 31, 0]]
selected = [0, 0, 0, 0, 0]
no_edge = 0
selected[0] = True
print("Edge : Weight\n")
while no_edge < V - 1:
    minimum = INF
    x, y = 0, 0
    for i in range(V):
        if selected[i]:
            for j in range(V):
                if (not selected[j]) and G[i][j]:
                    if minimum > G[i][j]:
                        minimum = G[i][j]
                        x, y = i, j
    print(str(x) + "-" + str(y) + ":" + str(G[x][y]))
    selected[y] = True
    no_edge += 1

```

The screenshot shows an IDE with the following components:

- Project Explorer:** Lists various files including '2\_Bubble Sort Final.docx', '3\_Selection Sort.py', '4\_Merge Sort.docx', '5\_Quick Sort.docx', '6\_Heap Sort.docx', '7\_Counting Sort.docx', '8\_Radix Sort.py', '9\_Knapsack Problem.docx', '10\_Breadth First Search.docx', '11\_Depth First Search.docx', and '12\_Prime's Algorithm.py'.
- Code Editor:** Displays the Python code for Prim's Algorithm. The code defines a graph G, initializes a selected array, and uses a while loop to find the minimum weight edge that doesn't create a cycle or a vertex with a degree greater than 2. The output is printed as 'Edge : Weight' followed by the edge and its weight.
- Terminal/Console:** Shows the output of the program, which is 'Edge : Weight'.
- Status Bar:** Indicates the current file is '12\_Prime's Algorithm.py', the encoding is 'UTF-8', and the Python version is 'Python 3.8 (base)'.

## Output:

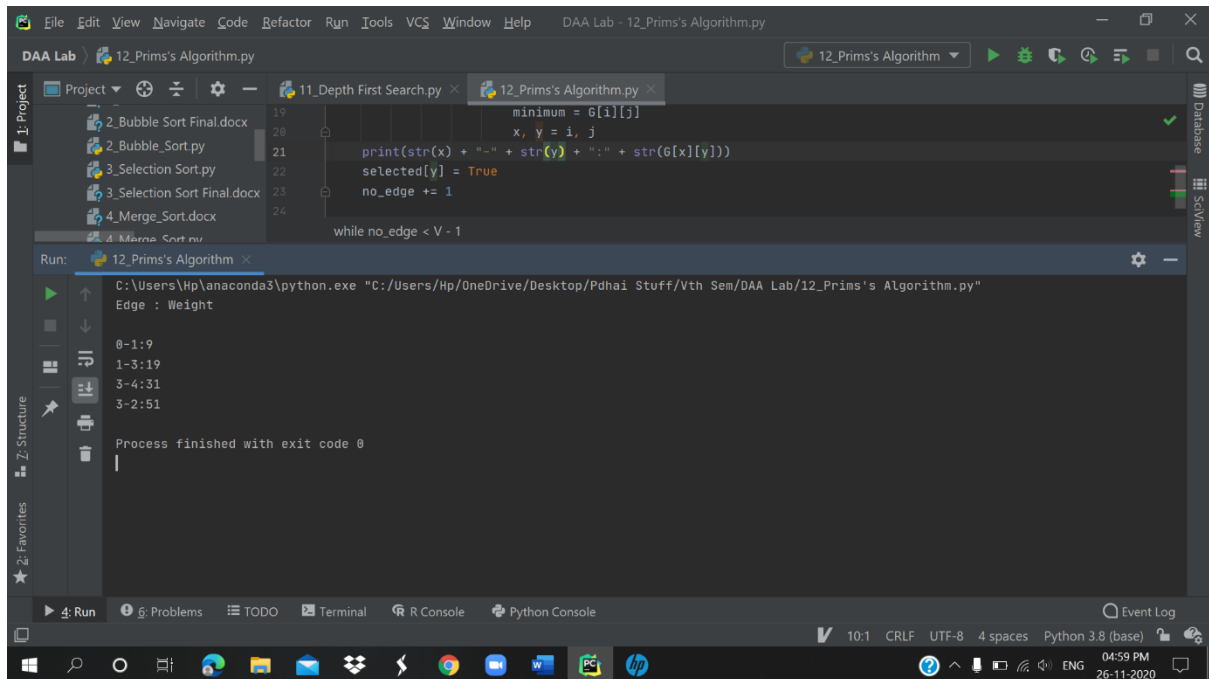
Edge : Weight

0-1:9

1-3:19

3-4:31

3-2:51



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help DAA Lab - 12_Prims's Algorithm.py
DAA Lab 12_Prims's Algorithm.py
Project 11_Depth First Search.py 12_Prims's Algorithm.py
2_Bubble Sort Final.docx
2_Bubble_Sort.py
3_Selection Sort.py
3_Selection Sort Final.docx
4_Merge_Sort.docx
4_Merge_Sort.py
19 minimum = G[i][j]
20 x, y = i, j
21 print(str(x) + "-" + str(y) + ":" + str(G[x][y]))
22 selected[y] = True
23 no_edge += 1
24 while no_edge < V - 1
Run: 12_Prims's Algorithm
C:\Users\Hp\anaconda3\python.exe "C:/Users/Hp/OneDrive/Desktop/Pdhai Stuff/Vth Sem/DAA Lab/12_Prims's Algorithm.py"
Edge : Weight
0-1:9
1-3:19
3-4:31
3-2:51
Process finished with exit code 0
4: Run Problems TODO Terminal R Console Python Console
10:1 CRLF UTF-8 4 spaces Python 3.8 (base)
04:59 PM
26-11-2020
```

## Time Complexities:

The time complexity of Prim's algorithm is  $O(E \log V)$ .

## Prim's Algorithm Applications:

- Laying cables of electrical wiring
- In network designed
- To make protocols in network cycles