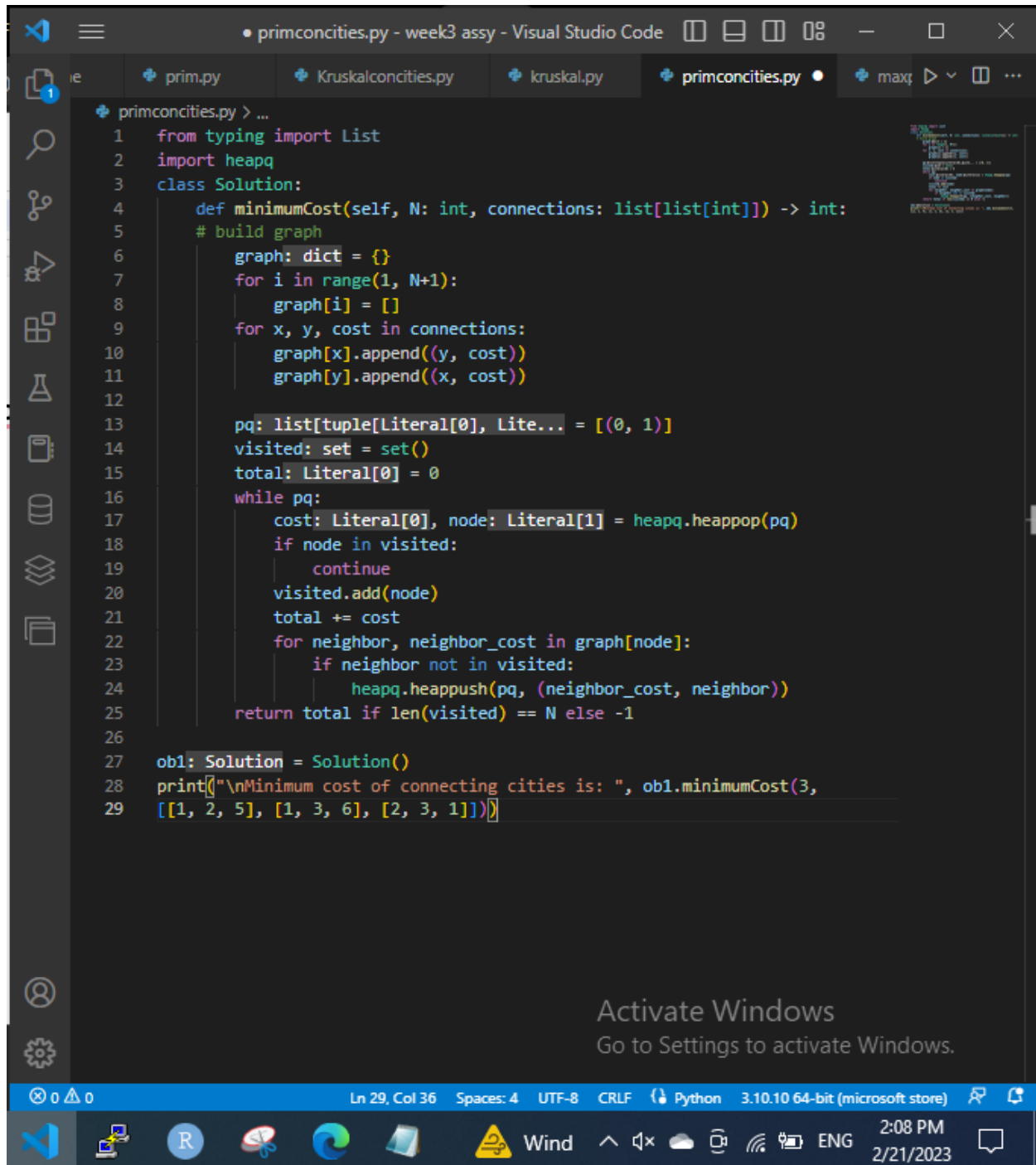


# Week 5 Homework 1: Greedy Algorithm : Minimum Spanning Tree (Prim) - LC

## STEP 1



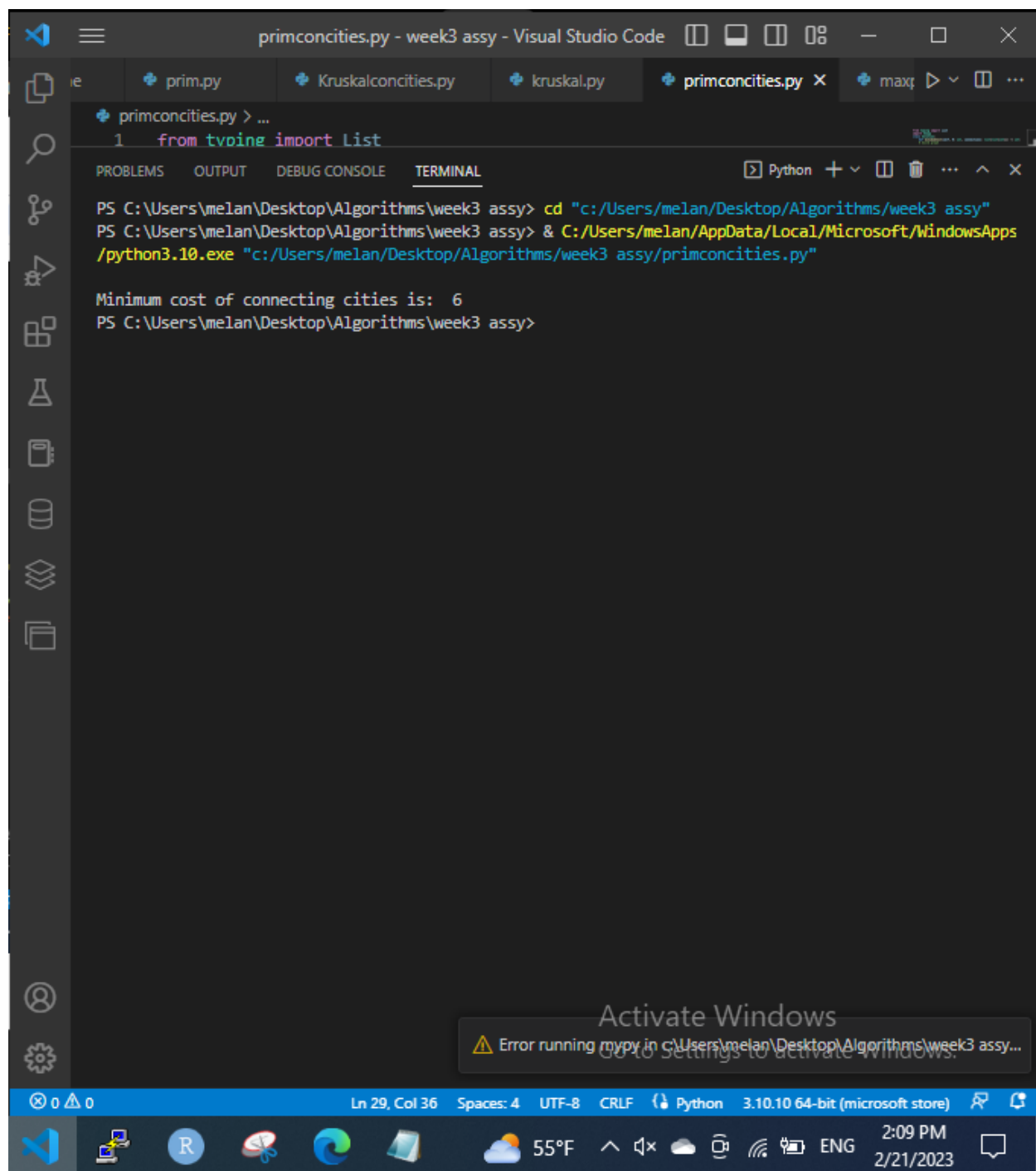
```
primconcities.py - week3 assy - Visual Studio Code
prim.py  primconcities.py  kruskalconcities.py  kruskal.py  primconcities.py  maxq

primconcities.py > ...
1  from typing import List
2  import heapq
3  class Solution:
4      def minimumCost(self, N: int, connections: list[list[int]]) -> int:
5          # build graph
6          graph: dict = {}
7          for i in range(1, N+1):
8              graph[i] = []
9          for x, y, cost in connections:
10             graph[x].append((y, cost))
11             graph[y].append((x, cost))
12
13             pq: list[tuple[Literal[0], Literal[1]]] = [(0, 1)]
14             visited: set = set()
15             total: Literal[0] = 0
16             while pq:
17                 cost: Literal[0], node: Literal[1] = heapq.heappop(pq)
18                 if node in visited:
19                     continue
20                 visited.add(node)
21                 total += cost
22                 for neighbor, neighbor_cost in graph[node]:
23                     if neighbor not in visited:
24                         heapq.heappush(pq, (neighbor_cost, neighbor))
25             return total if len(visited) == N else -1
26
27 ob1: Solution = Solution()
28 print("\nMinimum cost of connecting cities is: ", ob1.minimumCost(3,
29 [[1, 2, 5], [1, 3, 6], [2, 3, 1]]))
```

Activate Windows  
Go to Settings to activate Windows.

Ln 29, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.10 64-bit (microsoft store)

2:08 PM 2/21/2023



primconcities.py - week3 assy - Visual Studio Code

prim.py Kruskalconcities.py kruskal.py primconcities.py x max

primconcities.py > ...

```
1 from typing import List
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + -

PS C:\Users\melan\Desktop\Algorithms\week3 assy> cd "c:/Users/melan/Desktop/Algorithms/week3 assy"

PS C:\Users\melan\Desktop\Algorithms\week3 assy> & C:/Users/melan/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/melan/Desktop/Algorithms/week3 assy/primconcities.py"

Minimum cost of connecting cities is: 6

PS C:\Users\melan\Desktop\Algorithms\week3 assy>

Activate Windows  
Go to Settings to activate Windows.

Error running mypy in c:\Users\melan\Desktop\Algorithms\week3 assy...

Ln 29, Col 36 Spaces: 4 UTF-8 CRLF Python 3.10.10 64-bit (microsoft store) ENG 2:09 PM 2/21/2023

## CODE

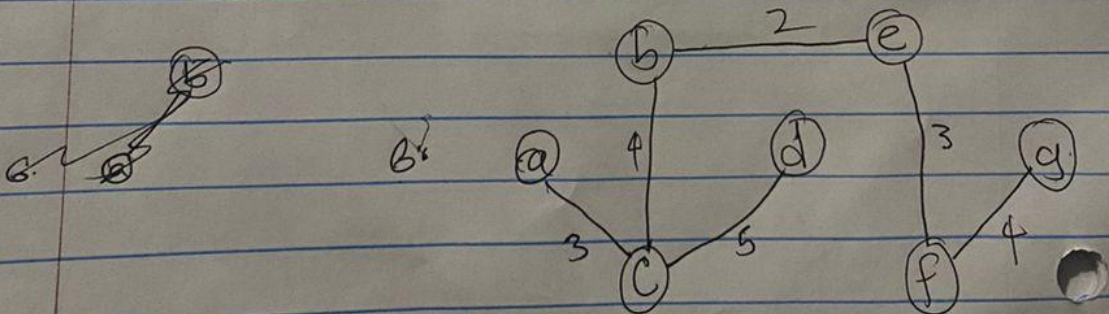
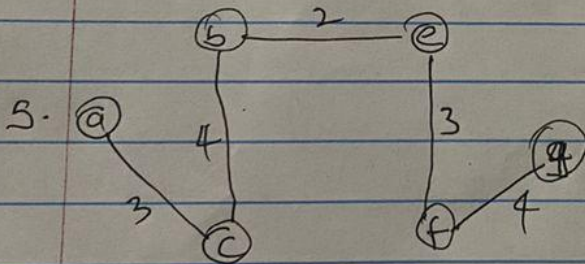
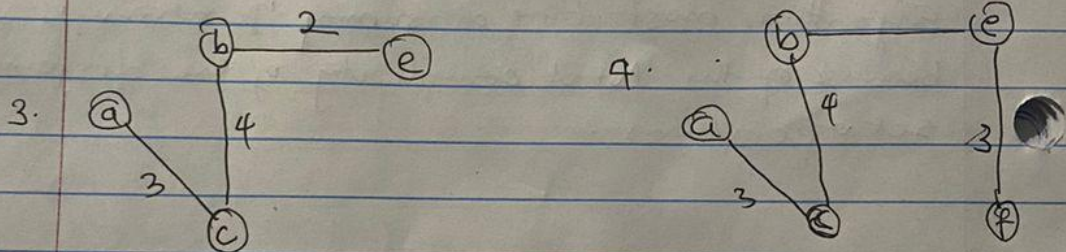
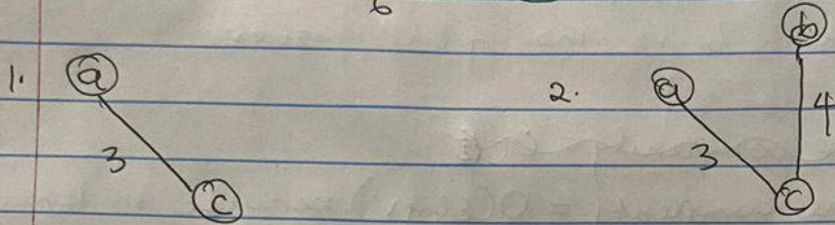
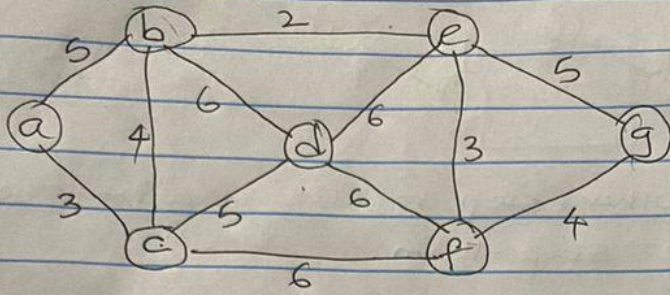
```
from typing import List
import heapq
class Solution:
    def minimumCost(self, N: int, connections: list[list[int]]) -> int:
        # build graph
        graph = {}
        for i in range(1, N+1):
            graph[i] = []
        for x, y, cost in connections:
            graph[x].append((y, cost))
            graph[y].append((x, cost))

        pq = [(0, 1)]
        visited = set()
        total = 0
        while pq:
            cost, node = heapq.heappop(pq)
            if node in visited:
                continue
            visited.add(node)
            total += cost
            for neighbor, neighbor_cost in graph[node]:
                if neighbor not in visited:
                    heapq.heappush(pq, (neighbor_cost, neighbor))
        return total if len(visited) == N else -1

ob1 = Solution()
print("\nMinimum cost of connecting cities is: ", ob1.minimumCost(3, [[1, 2, 5], [1, 3, 6], [2, 3, 1]]))
```

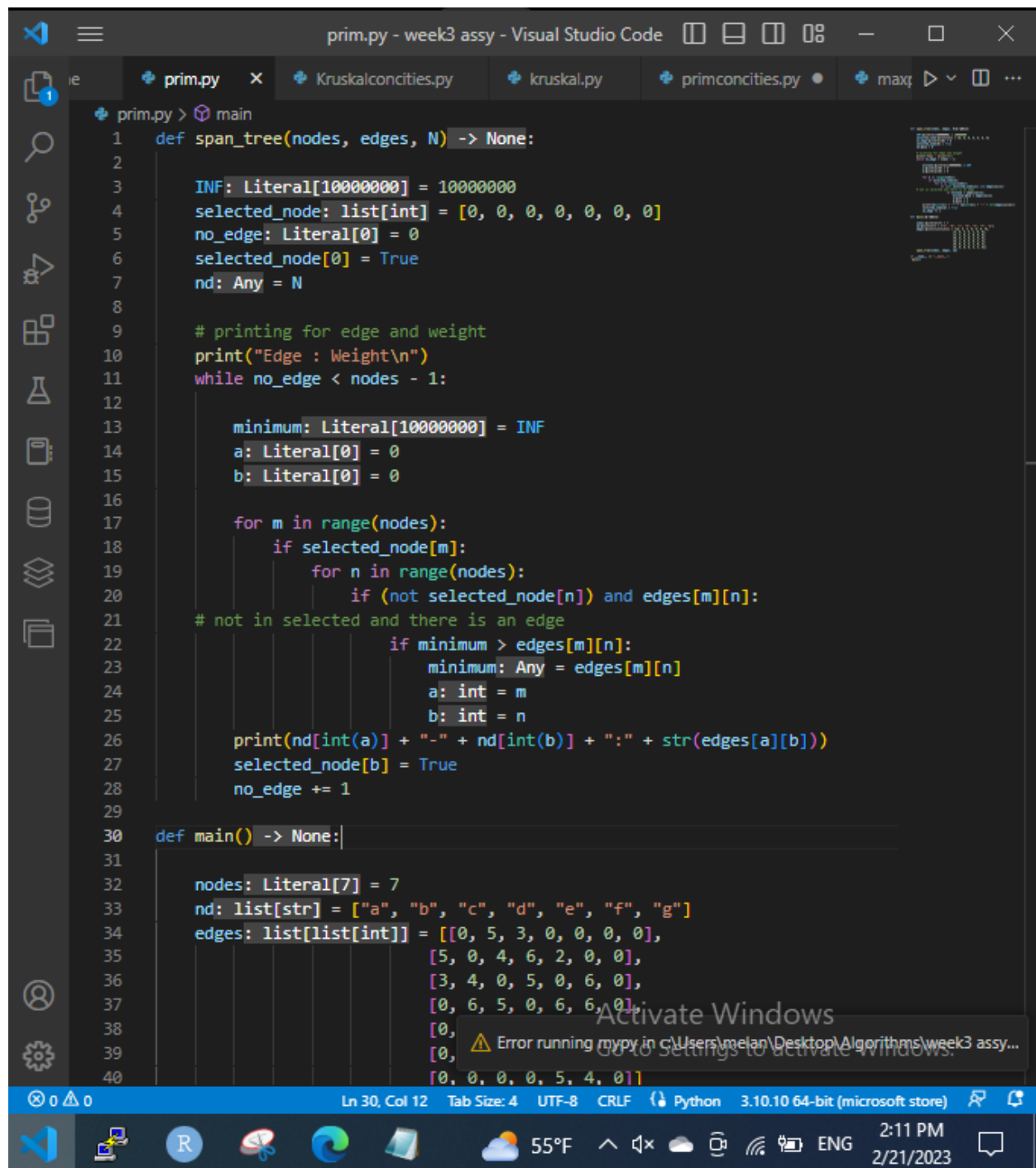
## STEP 2

Prims



Time complexity =  $O((V+E) \log V)$  because each edge is inserted in the priority queue only once and insertion in priority queue take logarithmic time.

### STEP 3

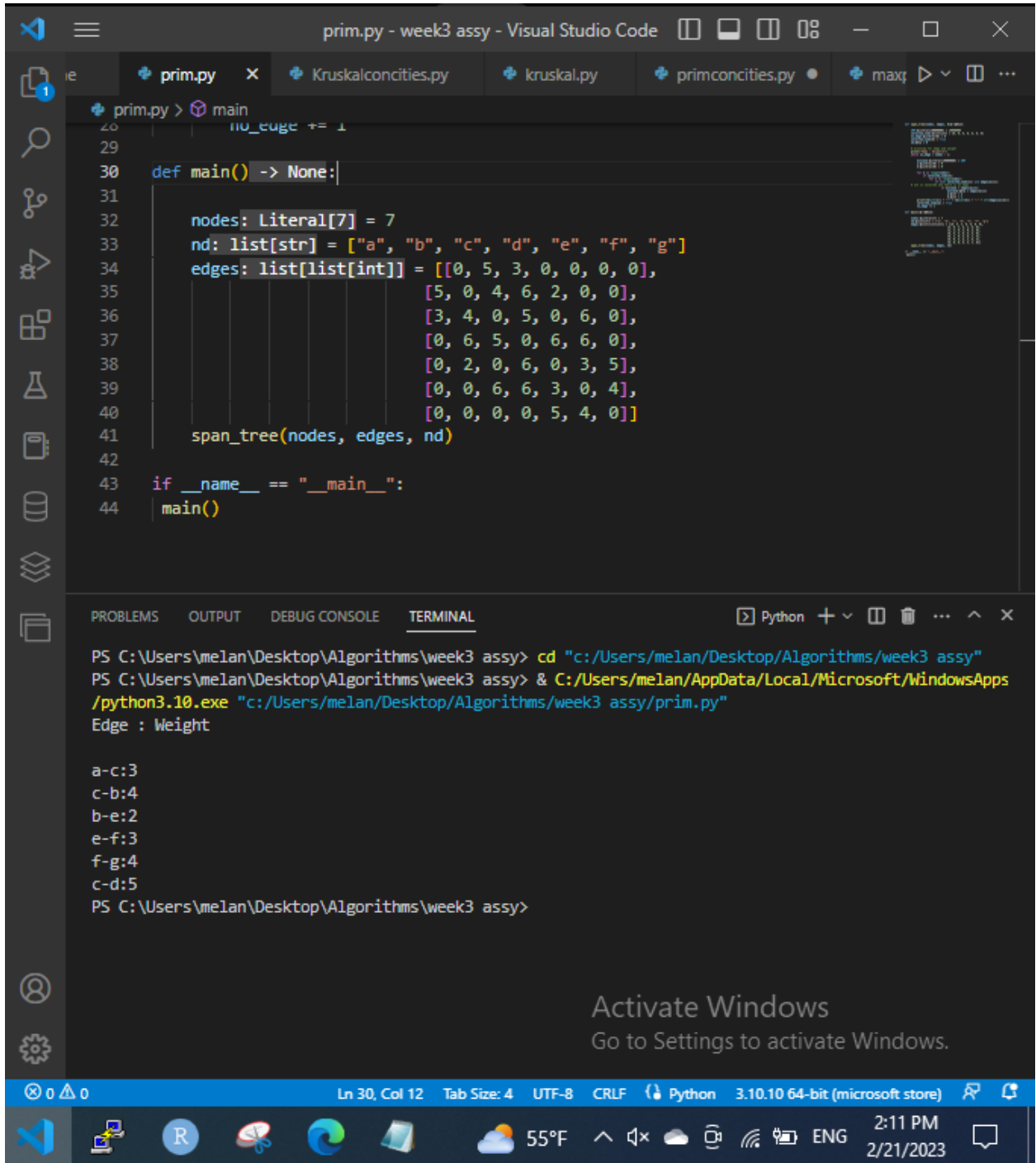


```
prim.py - week3 assy - Visual Studio Code
prim.py x Kruskalconcities.py kruskal.py primconcities.py maxt
prim.py > main
1 def span_tree(nodes, edges, N) -> None:
2
3     INF: Literal[10000000] = 10000000
4     selected_node: list[int] = [0, 0, 0, 0, 0, 0, 0]
5     no_edge: Literal[0] = 0
6     selected_node[0] = True
7     nd: Any = N
8
9     # printing for edge and weight
10    print("Edge : Weight\n")
11    while no_edge < nodes - 1:
12
13        minimum: Literal[10000000] = INF
14        a: Literal[0] = 0
15        b: Literal[0] = 0
16
17        for m in range(nodes):
18            if selected_node[m]:
19                for n in range(nodes):
20                    if (not selected_node[n]) and edges[m][n]:
21                        # not in selected and there is an edge
22                        if minimum > edges[m][n]:
23                            minimum: Any = edges[m][n]
24                            a: int = m
25                            b: int = n
26
27                        print(nd[int(a)] + "-" + nd[int(b)] + ":" + str(edges[a][b]))
28                        selected_node[b] = True
29                        no_edge += 1
30
31    def main() -> None:
32
33        nodes: Literal[7] = 7
34        nd: list[str] = ["a", "b", "c", "d", "e", "f", "g"]
35        edges: list[list[int]] = [[0, 5, 3, 0, 0, 0, 0],
36                                  [5, 0, 4, 6, 2, 0, 0],
37                                  [3, 4, 0, 5, 0, 6, 0],
38                                  [0, 6, 5, 0, 6, 6, 0],
39                                  [0,
40                                  [0, 0, 0, 0, 5, 4, 0]]
```

Activate Windows  
Go to Settings to activate Windows.

Error running mypy in c:\Users\melan\Desktop\Algorithms\week3 assy...

Ln 30, Col 12 Tab Size: 4 UTF-8 CRLF Python 3.10.10 64-bit (microsoft store) 2:11 PM 2/21/2023



CODE

```
def span_tree(nodes, edges, N):
```

```
    INF = 10000000
```

```
    selected_node = [0, 0, 0, 0, 0, 0, 0]
```

```
    no_edge = 0
```

```
    selected_node[0] = True
```

```
    nd = N
```

```
    # printing for edge and weight
```

```
    print("Edge : Weight\n")
```

```
    while no_edge < nodes - 1:
```

```
        minimum = INF
```

```
        a = 0
```

```
        b = 0
```

```
        for m in range(nodes):
```

```
            if selected_node[m]:
```

```
                for n in range(nodes):
```

```
                    if (not selected_node[n]) and edges[m][n]:
```

```
                        # not in selected and there is an edge
```

```
                            if minimum > edges[m][n]:
```

```
                                minimum = edges[m][n]
```

```
                                a = m
```

```
                                b = n
```

```
        print(nd[int(a)] + "-" + nd[int(b)] + ":" + str(edges[a][b]))
```

```
        selected_node[b] = True
```

```
        no_edge += 1
```

```
def main():
```

```
    nodes = 7
```

```
    nd = ["a", "b", "c", "d", "e", "f", "g"]
```

```
    edges = [[0, 5, 3, 0, 0, 0, 0],
```

```
             [5, 0, 4, 6, 2, 0, 0],
```

```
             [3, 4, 0, 5, 0, 6, 0],
```

```
             [0, 6, 5, 0, 6, 6, 0],
```

```
             [0, 2, 0, 6, 0, 3, 5],
```

```
             [0, 0, 6, 6, 3, 0, 4],
```

```
             [0, 0, 0, 0, 5, 4, 0]]
```

```
    span_tree(nodes, edges, nd)
```

```
if __name__ == "__main__":  
    main()
```