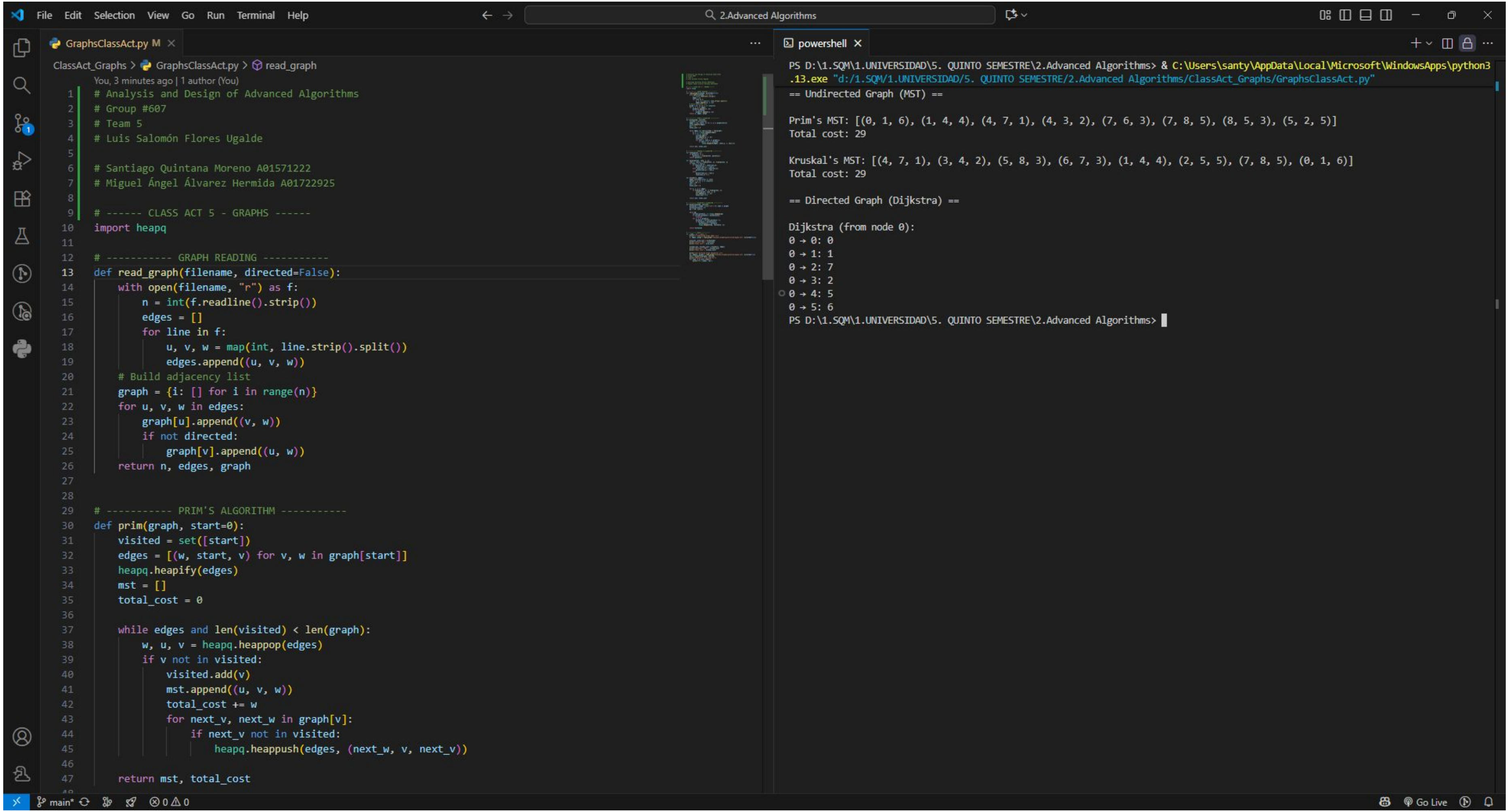
Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Class Activity 5:
Graphs

Group: 607
Team #5
Luis Salomón Flores Ugalde

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925



```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 5
# Luis Salomón Flores Ugalde

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ------ CLASS ACT 5 - GRAPHS ------
import heapq

# ---------- GRAPH READING ----------
def read_graph(filename, directed=False):
    with open(filename, "r") as f:
        n = int(f.readline().strip())
        edges = []
        for line in f:
            u, v, w = map(int, line.strip().split())
            edges.append((u, v, w))
    # Build adjacency list
    graph = {i: [] for i in range(n)}
    for u, v, w in edges:
        graph[u].append((v, w))
        if not directed:
            graph[v].append((u, w))
    return n, edges, graph


# ---------- PRIM'S ALGORITHM ----------
def prim(graph, start=0):
    visited = set([start])
    edges = [(w, start, v) for v, w in graph[start]]
    heapq.heapify(edges)
    mst = []
    total_cost = 0

    while edges and len(visited) < len(graph):
        w, u, v = heapq.heappop(edges)
        if v not in visited:
            visited.add(v)
            mst.append((u, v, w))
            total_cost += w
            for next_v, next_w in graph[v]:
                if next_v not in visited:
                    heapq.heappush(edges, (next_w, v, next_v))

    return mst, total_cost
```

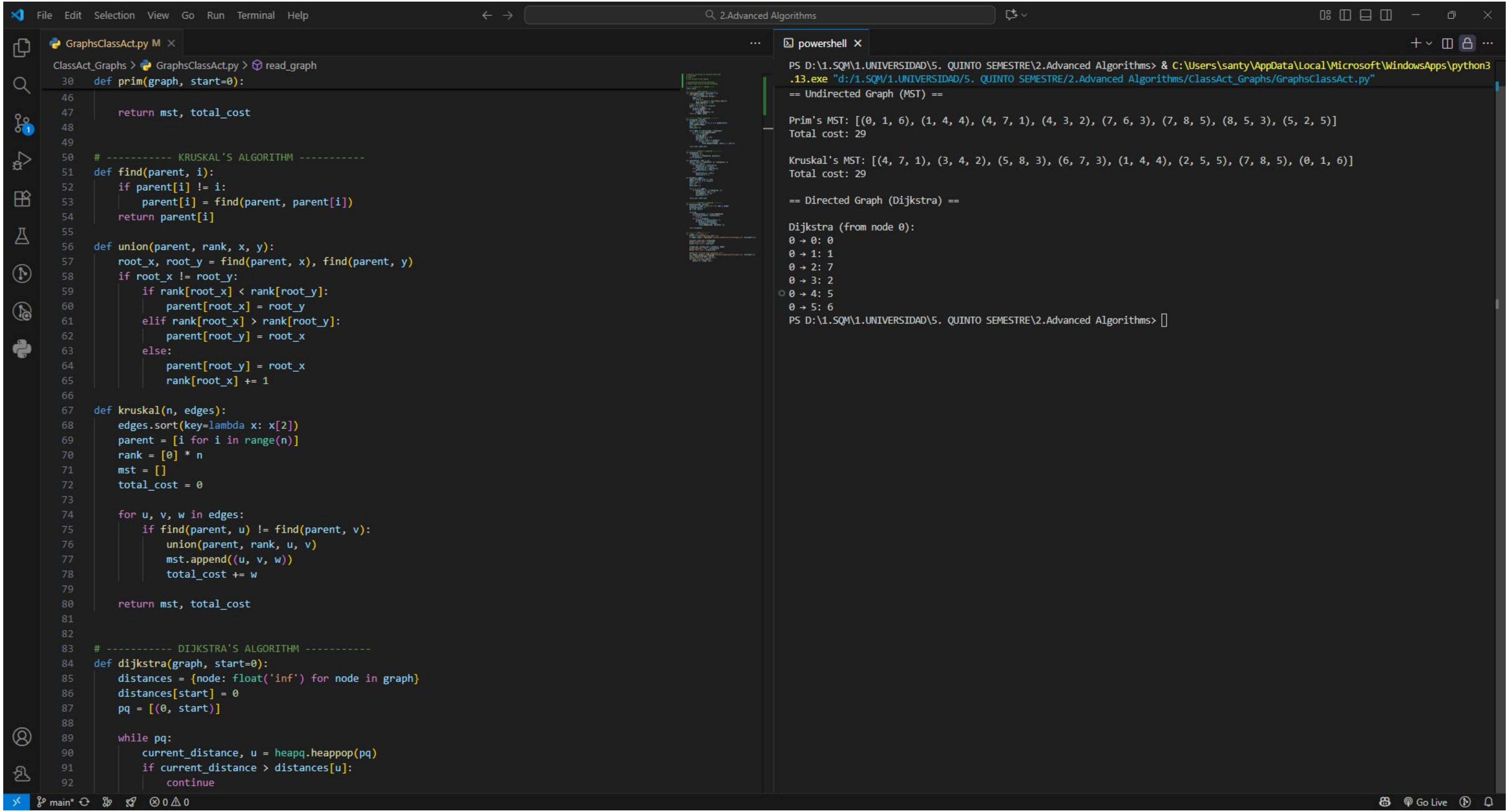
```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3
.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/ClassAct_Graphs/GraphsClassAct.py"
== Undirected Graph (MST) ==

Prim's MST: [(0, 1, 6), (1, 4, 4), (4, 7, 1), (4, 3, 2), (7, 6, 3), (7, 8, 5), (8, 5, 3), (5, 2, 5)]
Total cost: 29

Kruskal's MST: [(4, 7, 1), (3, 4, 2), (5, 8, 3), (6, 7, 3), (1, 4, 4), (2, 5, 5), (7, 8, 5), (0, 1, 6)]
Total cost: 29

== Directed Graph (Dijkstra) ==

Dijkstra (from node 0):
0 → 0: 0
0 → 1: 1
0 → 2: 7
0 → 3: 2
0 → 4: 5
0 → 5: 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
GraphsClassAct.py M
ClassAct_Graphs > GraphsClassAct.py > read_graph
powershell
main*
⊗ 0 ⚠ 0
Go Live

```python
def prim(graph, start=0):

    return mst, total_cost


# ---------- KRUSKAL'S ALGORITHM ----------
def find(parent, i):
    if parent[i] != i:
        parent[i] = find(parent, parent[i])
    return parent[i]

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

def kruskal(n, edges):
    edges.sort(key=lambda x: x[2])
    parent = [i for i in range(n)]
    rank = [0] * n
    mst = []
    total_cost = 0

    for u, v, w in edges:
        if find(parent, u) != find(parent, v):
            union(parent, rank, u, v)
            mst.append((u, v, w))
            total_cost += w

    return mst, total_cost


# ---------- DIJKSTRA'S ALGORITHM ----------
def dijkstra(graph, start=0):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    pq = [(0, start)]

    while pq:
        current_distance, u = heapq.heappop(pq)
        if current_distance > distances[u]:
            continue
```

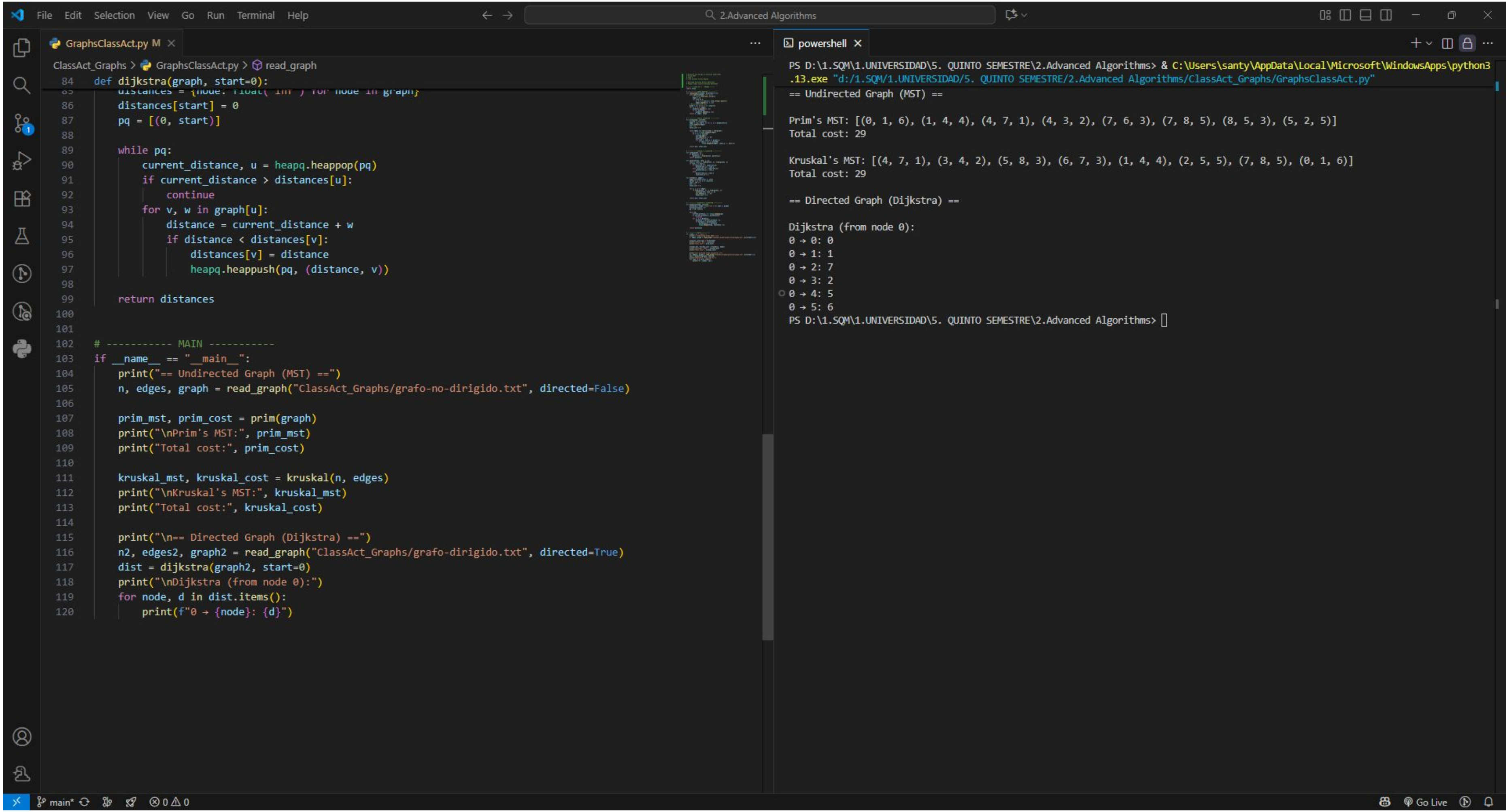
```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3
.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/ClassAct_Graphs/GraphsClassAct.py"
== Undirected Graph (MST) ==

Prim's MST: [(0, 1, 6), (1, 4, 4), (4, 7, 1), (4, 3, 2), (7, 6, 3), (7, 8, 5), (8, 5, 3), (5, 2, 5)]
Total cost: 29

Kruskal's MST: [(4, 7, 1), (3, 4, 2), (5, 8, 3), (6, 7, 3), (1, 4, 4), (2, 5, 5), (7, 8, 5), (0, 1, 6)]
Total cost: 29

== Directed Graph (Dijkstra) ==

Dijkstra (from node 0):
0 → 0: 0
0 → 1: 1
0 → 2: 7
0 → 3: 2
0 → 4: 5
0 → 5: 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms>
```



```python
def dijkstra(graph, start=0):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    pq = [(0, start)]

    while pq:
        current_distance, u = heapq.heappop(pq)
        if current_distance > distances[u]:
            continue
        for v, w in graph[u]:
            distance = current_distance + w
            if distance < distances[v]:
                distances[v] = distance
                heapq.heappush(pq, (distance, v))

    return distances


# ---------- MAIN ----------
if __name__ == "__main__":
    print("== Undirected Graph (MST) ==")
    n, edges, graph = read_graph("ClassAct_Graphs/grafo-no-dirigido.txt", directed=False)

    prim_mst, prim_cost = prim(graph)
    print("\nPrim's MST:", prim_mst)
    print("Total cost:", prim_cost)

    kruskal_mst, kruskal_cost = kruskal(n, edges)
    print("\nKruskal's MST:", kruskal_mst)
    print("Total cost:", kruskal_cost)

    print("\n== Directed Graph (Dijkstra) ==")
    n2, edges2, graph2 = read_graph("ClassAct_Graphs/grafo-dirigido.txt", directed=True)
    dist = dijkstra(graph2, start=0)
    print("\nDijkstra (from node 0):")
    for node, d in dist.items():
        print(f"0 → {node}: {d}")
```

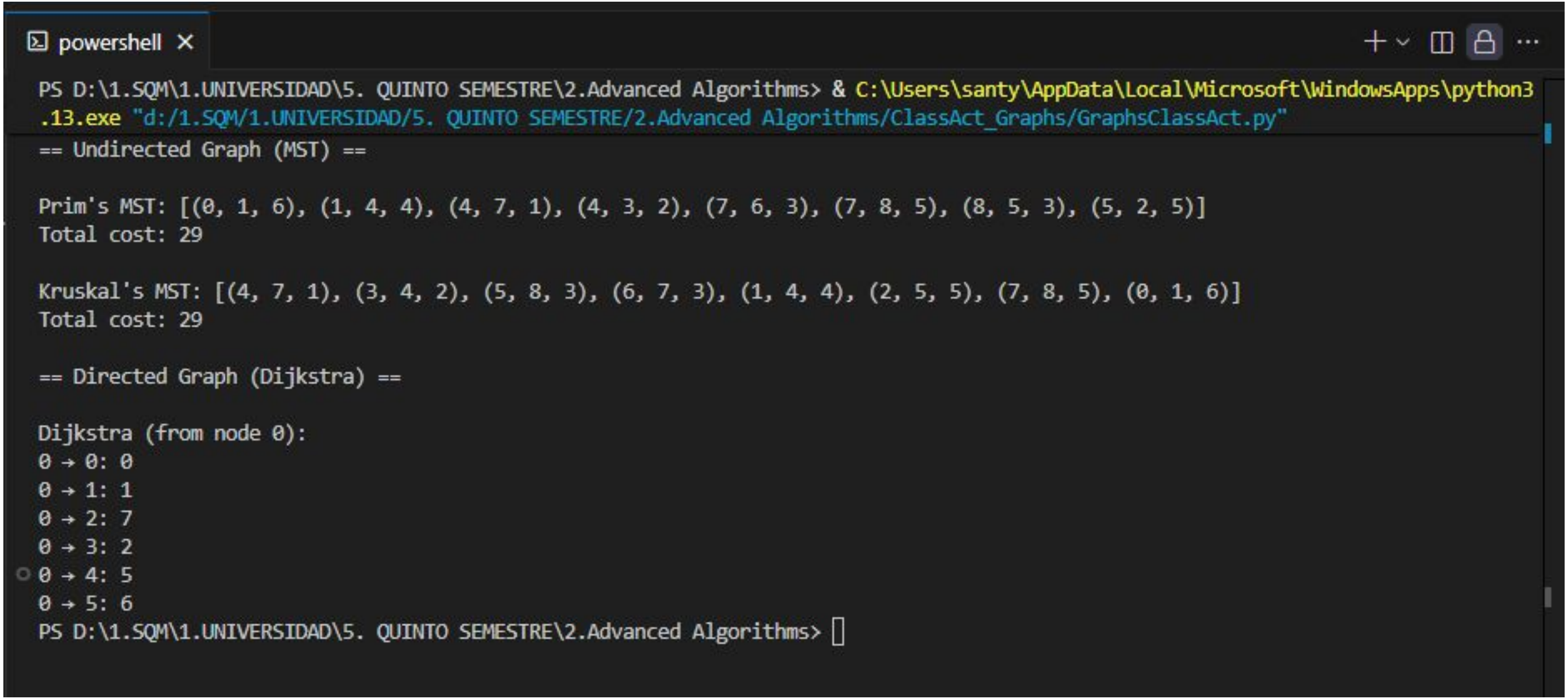
```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3
.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/ClassAct_Graphs/GraphsClassAct.py"
== Undirected Graph (MST) ==

Prim's MST: [(0, 1, 6), (1, 4, 4), (4, 7, 1), (4, 3, 2), (7, 6, 3), (7, 8, 5), (8, 5, 3), (5, 2, 5)]
Total cost: 29

Kruskal's MST: [(4, 7, 1), (3, 4, 2), (5, 8, 3), (6, 7, 3), (1, 4, 4), (2, 5, 5), (7, 8, 5), (0, 1, 6)]
Total cost: 29

== Directed Graph (Dijkstra) ==

Dijkstra (from node 0):
0 → 0: 0
0 → 1: 1
0 → 2: 7
0 → 3: 2
0 → 4: 5
0 → 5: 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> 
```



```
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & C:\Users\santy\AppData\Local\Microsoft\WindowsApps\python3
.13.exe "d:/1.SQM/1.UNIVERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/ClassAct_Graphs/GraphsClassAct.py"
== Undirected Graph (MST) ==

Prim's MST: [(0, 1, 6), (1, 4, 4), (4, 7, 1), (4, 3, 2), (7, 6, 3), (7, 8, 5), (8, 5, 3), (5, 2, 5)]
Total cost: 29

Kruskal's MST: [(4, 7, 1), (3, 4, 2), (5, 8, 3), (6, 7, 3), (1, 4, 4), (2, 5, 5), (7, 8, 5), (0, 1, 6)]
Total cost: 29

== Directed Graph (Dijkstra) ==

Dijkstra (from node 0):
0 → 0: 0
0 → 1: 1
0 → 2: 7
0 → 3: 2
0 → 4: 5
0 → 5: 6
PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> 
```

https://colab.research.google.com/drive/1f4WjwENwSfpbqYkGJKTiAdzXbO26vy48?usp=sharing

REFERENCES

GeeksforGeeks. (2025, July 23). *Introduction to graphs in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/python/introduction-to-graphs-in-python/

GeeksforGeeks. (2025, August 26). *Kruskal’s Minimum Spanning Tree (MST) Algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

GeeksforGeeks. (2025, August 29). *Prim’s Algorithm for Minimum Spanning Tree (MST)*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/prims-minimum-spanning-tree-mst-greedy-algo-5/