Programming Assignment 2
Submitted by : Tushar Bansal (tb34826)

1. **PSEUDO CODE :**

findMinimumStudentCost(start, destination):

```
for each student in students:                                          O(V)
        minimum cost of student ← infinity

minimum cost of start ← 0                                              O(1)

build the min-Heap ← students                                          O(V)

while min-Heap is ≠ ϕ:                                                 O(V)
        current student ← extract minimum from min-Heap                O(log V)

        if current student ==  destination:
                return min. cost of destination                        O(1)

        for each neighbor of current student:                          O(E)
                new cost ←  min. cost of current + price from current to neighbor

                if new cost < min. cost of neighbor:                   O(1)
                        neighbor's key in min-Heap ←  new cost         O(log V)
                        min. cost of neighbor ←  new cost              O(1)

return -1 // If the destination is not reachable from start
```

DIJKSTRA'S ALGORITHM COMPLEXITY (also check the red part above)

- With values stored in min-heap size V
- Initialization of costs for students takes O(V)
- Initializing Heap takes O(V)
- EXTRACT-MIN and DECREASE-KEY take O(log V)
- While loop runs for O(V)
- Each doing EXTRACT-MIN, so O(V log V) overall
- For loop runs O(E) times and if statement contains changing KEY which is O(log V), so O(E log V) overall
- Overall running time O((V+E) log V)
- Since connected graph V = O(E), so O(E lg V)

2. **PSEUDO CODE :**

findMinimumClassCost():

```
total cost ← 0

for each student in students:                                          O(V)
        minimum cost of student ← infinity

start = first student in students                                      O(1)
minimum cost of start ← 0                                              O(1)

build the min-Heap ← students                                          O(V)

while min-Heap ≠ ϕ:                                                    O(V)
        current student ← extract minimum from min-Heap                O(log V)

        totalCost += min. cost of current student

        for each neighbor of current student:                          O(E)
                price ← price from current to neighbor                 O(1)

                if neighbor ϵ min-Heap and price < min. cost of neighbor:   O(1)
                        neighbor's key in min-Heap ← price              O(log V)
                        min. cost of neighbor ← price                  O(1)

return total cost
```

PRIM'S ALGORITHM COMPLEXITY (also check the red part above)

- With values stored in min-heap size V
- Initialization of costs for students takes O(V)
- Initialization of first student and its cost takes O(1)
- Initializing Heap takes O(V)
- EXTRACT-MIN and DECREASE-KEY take O(log V)
- While loop runs O(V)
- Each doing EXTRACT-MIN, so O(V log V) overall
- For loop runs O(E) times and if statement contains changing KEY which is O(log V), so O(E log V) overall
- In the if statement, to check if the neighbor belongs to min-heap, there is a Hash-Map of indices in the Min-Heap which makes the return equal to O(1)
- Overall running time O((V+E) log V)
- Since connected graph V = O(E), so O(E lg V)