Algorithm Laboratory Exam

Objective

The problem under analysis concerns communication between outposts, which can occur through satellite channels and/or radio channels. Two outposts with satellite channels can communicate directly with each other, while two outposts with radio channels, to communicate, must be placed at a distance D that depends on the power of the radio transceiver. Given as input the number of satellite channels, the number of outposts, and their coordinates, we want to establish how to assign satellite channels to the respective outposts in such a way as to minimize the value of D as much as possible, and consequently save on the purchase of radio transmitters.

Solution Strategy

To solve this problem, we acquired the input in the form of a graph: the outposts are represented as nodes and their distance represents the weight of the edges connecting the outposts. To determine the minimum distance that allows connecting the outposts, we decided to calculate the Minimum Spanning Tree of the graph. Once the MST is computed, in order to calculate a minimum value of D to connect the outposts, we assign satellite channels to the ends of the edges with the greatest weight.

Input Acquisition

The main function allows acquiring multiple test cases, specifying for each of them the number of satellites, outposts, and their coordinates; verifying that each input respects the indications of the problem. In the case of correct input, a complete graph is created with a number of nodes equal to the number of outposts and with each vertex directly connected to all other vertices for a total of n*(n-1)/2 edges. Each edge will be labeled with the Euclidean distance between the coordinates of the various outposts. This graph is implemented using an adjacency list.

Operation	Spatial Complexity	Temporal Complexity
Graph Creation	O(V+E)	O(V+E)

MST Calculation

To determine the optimal way to connect the outposts, we decided to calculate the MST, obtaining the tree that connects the outposts with minimum distance. Among the various algorithms that allow calculating the MST, we decided to use eager Prim, which is more efficient where the number of vertices is less than the number of edges (as in the case under examination). The greedy approach for calculating the MST is correct, and this is guaranteed by the following theorem: Given any cut in a graph, the minimum weight crossing edge is in the MST. Different algorithms for calculating the MST are characterized by the choice of the cut and for the extraction of the minimum weight crossing edge, specifically Prim at each step selects as a cut the set of

vertices belonging to the MST currently computed and the set containing the remaining vertices; among the various crossing edges that have only one end in T, it chooses the one with minimum weight.

Specifically, to implement eager Prim, we used the following data structures:

- A vector of integers **edgeTo** of size V to keep track of the starting vertex of the edge that we insert into the MST. For example, edgeTo[v]=w indicates that we reach v in the MST through an edge that starts from w.
- A vector of doubles **distTo** of size V that indicates the weight of the edge indicated by edgeTo.
- A vector of bool **marked** of size V that keeps track of the vertices belonging to the MST, in order to avoid an edge that would create a cycle.
- A priority queue **pq** on vertices of size V to perform the extraction of the minimum and to possibly update the cost of the edge that leads to that vertex.
- A priority queue **maxEdgeHeap** on vertices of size V that keeps track of the edges belonging to the MST ordered by maximum key, necessary for the assignment of satellite channels.

For the two priority queues, we use the **PriorityQueue** class we created. The constructor of this class, in addition to the size, takes a boolean value as a parameter that allows us to establish whether to manage the heap for maximum or minimum. The PriorityQueue type object contains 3 internal arrays: the **pq** array represents our heap in which the first element (the max or min) is at index 1, the **qp** array where qp[i] indicates the position of vertex i in the heap, the **key** array in which key[i] indicates the cost to reach vertex i. (N.B.: for the qp and key arrays, the first effective value is at index 0).

Removing element i from the priority queue involves inserting the macro INF into qp[i]. In this way, understanding if an element is not present in the priority queue costs O(1) rather than performing the search in the heap, which in the worst case requires O(n).

Operation	**Spatial Complexity**	**Temporal Complexity**	**Explanation**
Construction of the two PQs	O(V)	O(VlogV)	V insertion operations of cost logV
Minimum Extraction	------	O(VlogV)	V deletion operations of cost logV
Priority Change	------	O(ElogV)	In the worst case, we perform E priority changes of cost logV

The operations of insertion, deletion, and priority change in the priority queue for vertices implemented with a heap cost O(logV). The search operation (contains in the code) in the priority queue occurs in O(1) time.

In total, in the worst case, the MST calculation operation requires **O(ElogV)**.

Satellite Channel Assignment

The choice of nodes to which to assign satellite channels in order to decrease the minimum value of "D" is made by extracting the edge of maximum weight from the MST, whose ends will be chosen for the installation of the satellite channel. The extraction of the maximum weight edge occurs by means of the **edgeTo** and **maxEdgeHeap** vectors, whose values were set during the execution of Prim's algorithm.

```
int w=this->maxEdgeHeap.getRoot();
int v=this->edgeTo[w];
```

Thanks to the previous instructions, we can determine the v->w edge of maximum weight of the MST. Once this edge is obtained, by means of the **hasSatellite** vector of boolean values, we assign satellite channels to the extreme vertices. We iterate this process until we exhaust the number of satellite channels to be distributed.

Correctness:

Once the MST is calculated, if we were not to assign any satellite channel, the choice of the minimum distance that all outposts must cover for radio communication coincides with the greatest distance that connects two outposts. This thesis suggests assigning satellite channels to the most distant outposts. In this way, attributing 2 satellite channels to the vertices of the maximum weight edge, the minimum distance will coincide with the next maximum weight edge of the MST, or with the next maximum distance that radio communication must cover.

Let's prove then that the minimum distance coincides with the edge of greatest weight of the MST, remembering that the MST connects all vertices of the graph with minimum cost, where the cost in our case represents the distance between the outposts. Let's assume by contradiction that the minimum D is given by the cost of an edge "**e**" that is not the one of maximum cost "**f**" in the MST. This means that all outposts will be assigned the same radio transmitter with power capable of covering the distance given by the cost of edge "e". Since the cost of "e" is less than the cost of "f", the outposts corresponding to the extreme vertices of "f" would not be able to communicate with such a transmitter. Conversely, with a transmitter capable of covering the maximum distance of the MST, all outposts will be able to communicate.

Given as input S, the number of satellite channels to be assigned, we denote the cost of the algorithm:

Operation	**Spatial Complexity**	**Spatial Complexity**	**Explanation**
Maximum Extraction	------	O(SlogV)	In the worst case, i.e., when we extract from the queue each time an edge of which one end is marked, the extraction operation is performed exactly S-1 times each of cost logV.
Satellite Marking	O(V)	O(S)	We use an array of size V, the verification if an edge is marked by a satellite occurs at most exactly S times.

In total, in the worst case, the operation of calculating the minimum distance requires **O(SlogV)**.

Overall, in order to solve the problem under examination, we calculate the MST in O(ElogV) and the minimum distance in O(SlogV), which means that the temporal complexity of Prim's algorithm prevails: **O(ElogV)**.

Input Tests

We have carried out several tests to verify the functioning of the algorithm. Below we report 5 tests given as input to our program (in the output below there are further prints to verify the correct creation of the graph and calculation of the MST, launching the program only the minimum D will be displayed for each test).

- Test 1:

Input	Output
2 4 0 100 0 300 0 600 150 750	Adjacency List 0: 3 -> 667.08 2 -> 500.00 1 -> 200.00 1: 3 -> 474.34 2 -> 300.00 2: 3 -> 212.13 3: MST 0 -> 1 c: 200.00 1 -> 2 c: 300.00 2 -> 3 c: 212.13 Minimum D: 212.13

- Test 2:

Input	Output
4 5 66 10 900 310 36 60 150 750 65 250	Adjacency List 0: 4 -> 240.00 3 -> 744.75 2 -> 58.31 1 -> 886.32 1: 4 -> 837.15 3 -> 869.54 2 -> 899.44 2: 4 -> 192.20 3 -> 699.35 3: 4 -> 507.17 4: MST 0 -> 2 c: 58.31 2 -> 4 c: 192.20 2 -> 3 c: 699.35 0 -> 1 c: 886.32 Minimum D: 192.20

- Test 3:

Input	Output
4 4 237 12 356 15 67 111 12 35	Error in test 3! Enter a number of outposts greater than the number of satellites and less than 500!

- Test 4:

Input	Output
3 5 22 34 66 35 112 47 175 134 0 120	Adjacency List 0: 4 -> 88.77 3 -> 182.78 2 -> 90.93 1 -> 44.01 1: 4 -> 107.62 3 -> 147.25 2 -> 47.54 2: 4 -> 133.69 3 -> 107.42 3: 4 -> 175.56 4: MST 0 -> 1 c: 44.01 1 -> 2 c: 47.54 0 -> 4 c: 88.77 2 -> 3 c: 107.42 Minimum D: 88.77

- Test 5:

Input	Output
5 8 20 30 45 130 27 840 521 332 115 900 750 23 661 427 500 500	Adjacency List 0: 7 -> 671.79 6 -> 753.98 5 -> 730.03 4 -> 875.17 3 -> 584.98 2 -> 810.03 1 -> 103.08 1: 7 -> 586.45 6 -> 683.86 5 -> 713.07 4 -> 773.18 3 -> 517.09 2 -> 710.23 2: 7 -> 582.52 6 -> 756.65 5 -> 1090.97 4 -> 106.51 3 -> 708.59 3: 7 -> 169.31 6 -> 169.19 5 -> 384.61 4 -> 698.18 4: 7 -> 555.18 6 -> 722.39 5 -> 1082.75 5: 7 -> 538.54 6 -> 413.69 6: 7 -> 176.78 7: MST 0 -> 1 c: 103.08 1 -> 3 c: 517.09 3 -> 6 c: 169.19 3 -> 7 c: 169.31 3 -> 5 c: 384.61 3 -> 4 c: 698.18 1 -> 2 c: 710.23 Minimum D: 169.31