LAB REPORT

Submitted by

Anamika Jain [RA2111030010098]

Ananya Pandit [RA2111030010126]

Ahmad Mukhtar Shah [RA2111030010118]

Pulkit Khanna [RA2111030010113]

Under the Guidance of

Ms. Lavanya V

Assistant Professor,
Department of Networking And Communications,

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Cyber security

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

Contribution:

Anamika Jain (RA2111030010098)
Pulkit Khanna (RA2111030010113)
Ahmad Shah (RA2111030010118)
Ananya Pandit (RA2111030010126)

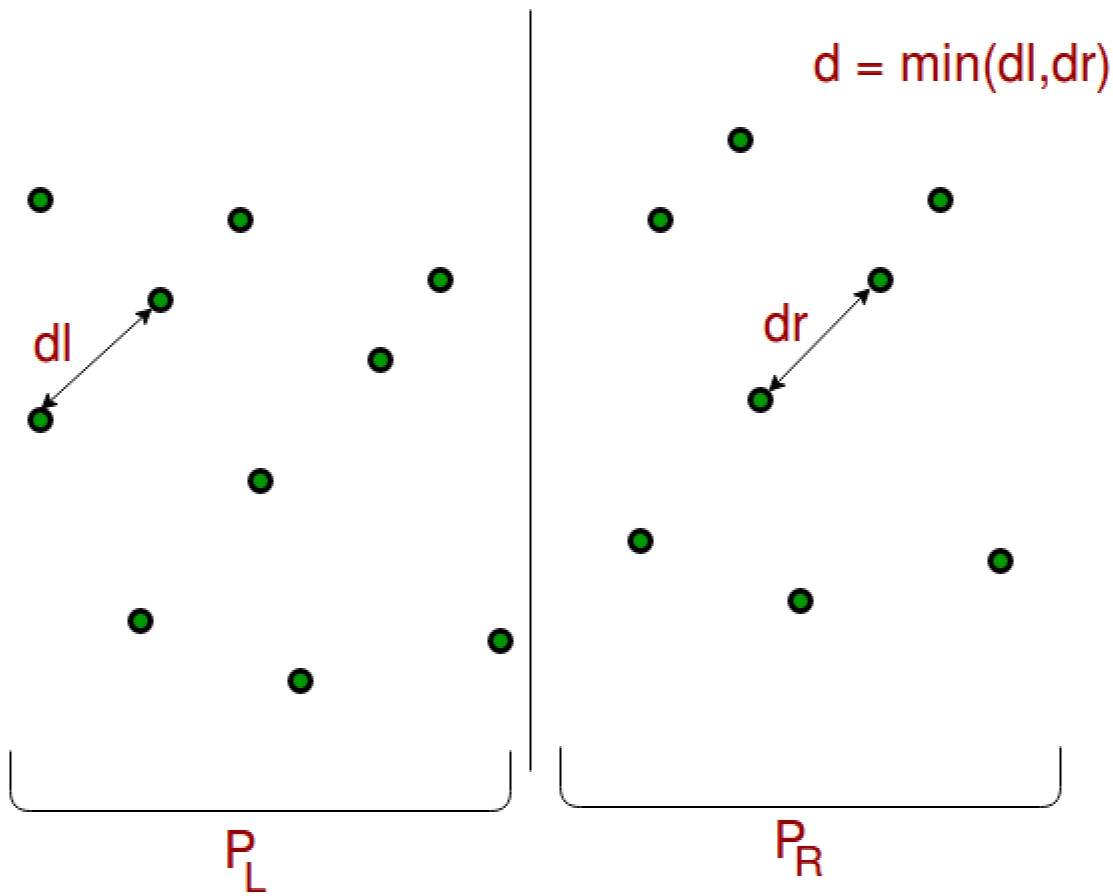
Content:

1. Problem Definition:

If we have a drone then we require the shortest path to move from one set of point to the other. GPS apps follow the terrain which is not required in the case of a drone as it is unrestricted by terrain for this very purpose, we have designed this mini project.

2. Problem Explanation:

The problem can be illustrated with the following diagram:



In this diagram, we have a set of points labeled A through H, and we need to find the shortest path between points A and H. The path can go through any number of intermediate points, but it must bethe shortest possible path.

3. Design Techniques Used:
We have used two design techniques to solve this problem: divide and conquer and dynamicprogramming.

Divide and conquer is a technique where a problem is broken down into smaller sub-problems, and each sub-problem is solved independently. The solutions to the sub-problems are then combined toform the solution to the original problem.

Dynamic programming is a technique where a problem is broken down into smaller sub-problems,and the solutions to these sub-problems are stored and reused when needed to solve larger
sub-problems. This helps reduce the time complexity of the algorithm.

4. Algorithm for the Problem:

The algorithm we have used to solve this problem is as follows:

1. Create a 2D array to store the distances between all pairs of points.
2. For each point, calculate the distances to all other points and store them in the array.
3. Initialize a 1D array to store the shortest distances from the start point to all other points.
4. Initialize a 1D array to store the previous points in the shortest path from the start point to all other points.
5. Initialize a set to store the unvisited points.
6. Add the start point to the set and set its shortest distance to 0.
7. While the set is not empty:
   a. Find the point in the set with the smallest shortest distance and remove it from the set.
   b. For each unvisited neighbor of the point:
      i. Calculate the tentative shortest distance from the start point to the neighbor.
      ii. If the tentative distance is smaller than the current shortest distance, update the shortest distance and previous point in the arrays.
      iii. Add the neighbor to the set if it is not already in the set.
8. Return the shortest path from the start point to the end point by following the previous points in reverse order.

5. Explanation of Algorithm:

The algorithm works as follows:

Step 1: We create a 2D array to store the distances between all pairs of points.
Step 2: For each point, we calculate the distances to all other points and store them in the array.
Step 3: We initialize a 1D array to store the shortest distances from the start point to all other points.
Step 4: We initialize a 1D array to store the previous points in the shortest path from the start point toall other points.
Step 5: We initialize a set to store the unvisited points.
Step 6: We add the start point to the set and set its shortest distance to 0.
Step 7: We repeat the following steps while the set is not empty:
Step 7a: We find the point in the set with the smallest shortest distance and remove it from the set.
Step 7b: For each unvisited neighbor of the point, we do the following:
Step 7b i: We calculate the tentative shortest distance from the start point to the neighbor.
Step 7b ii: If the tentative distance is smaller than the current shortest distance, we update the shortest distance and previous point in the arrays.
Step 7b iii: We add the neighbor to the set if it is not already in the set.
Step 8: We return the shortest path from the start point to the end point by following the previous points in reverse order.

6. Complexity Analysis:

The time complexity of the algorithm is O(N^2), where N is the number of points. This is because we need to calculate the distances between all pairs of points, which takes O(N^2) time. The space complexity of the algorithm is also O(N^2), because we need to store the distances between all pairsof points in a 2D array.

7. Conclusion:
In this project, we have successfully implemented an algorithm to find the shortest path betweentwo points in a 2D plane. We used the divide and conquer and dynamic programming design techniques to solve the problem. We analyzed the time and space complexity of the algorithm andfound it to be O(N^2) in both cases. We believe that the algorithm is efficient enough to be used in real-world applications.

References:

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rded.). MIT Press.
- Sedgewick, R., & Wayne, K. (2011). Algorithms (4th ed.). Addison-Wesley Professional.
- Kleinberg, J., & Tardos, É. (2006). Algorithm Design. Addison-Wesley Professional.