SAVEETHA SCHOOL OF ENGINEERING

SIMATS, CHENNAI - 602105

CSA0695-DESIGN AND ANALYSIS ALGORITHMS FOR OPEN ADRESSING TECHNIQUES

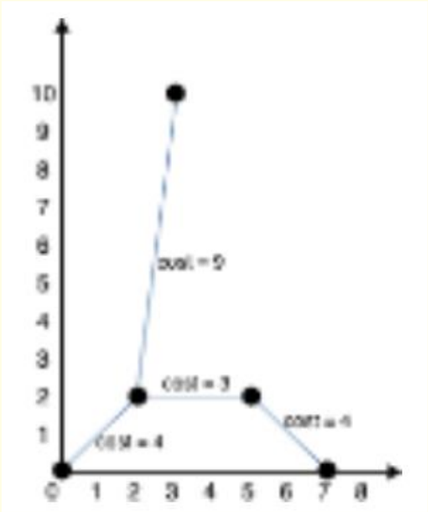
DIVIDE AND CONQUER-Minimizing the Cost to Connect All Points on a 2D Plane Using Manhattan Distance

Guided BY,
Dr . Dhanalakshmi

Presented by:-
K.Sasi kiran
CSE DEPARTMENT
SSE,SIMATS

PROBLEM STATEMENT:

- ✓ Min Cost to Connect All Points An array points representing integer coordinates of some points on a 2D-plane, where points[i] = [xi, yi].
- ✓ The cost of connecting two points [xi, yi] and [xj, yj] is the manhattan distance between them:
- ✓ |xi - xj| + |yi - yj|, where |val| denotes the absolute value of val. Return the minimum cost to make all points connected. All points are connected if there is exactly one simple path between any two points.



- ✓

- ✓ Example 1: Input: points = [[0,0],[2,2],[3,10],[5,2],[7,0]] Output: 20 Explanation: We can connect the points as shown above to get the minimum cost of 20. Notice that there is a unique path between every pair of points.

ABSTRACT:

The problem focuses on finding the minimum cost to connect a set of points in a 2D plane using the Manhattan distance between them. The task is to ensure that all points are connected such that there is only one simple path between any two points. This is equivalent to finding the Minimum Spanning Tree (MST) of the points using the Manhattan distance as the edge weights. In this solution, we will use Prim's algorithm for MST calculation, along with an efficient priority queue-based implementation. This paper discusses the approach, algorithm, C implementation, and time-space complexity for worst, best, and average cases

INTRODUCTION:

- The problem of connecting points on a 2D plane with minimal cost has applications in areas such as network design, urban planning, and computer graphics. The key challenge is to compute the minimum cost to connect all points such that no cycles are formed, resulting in exactly one simple path between any two points.

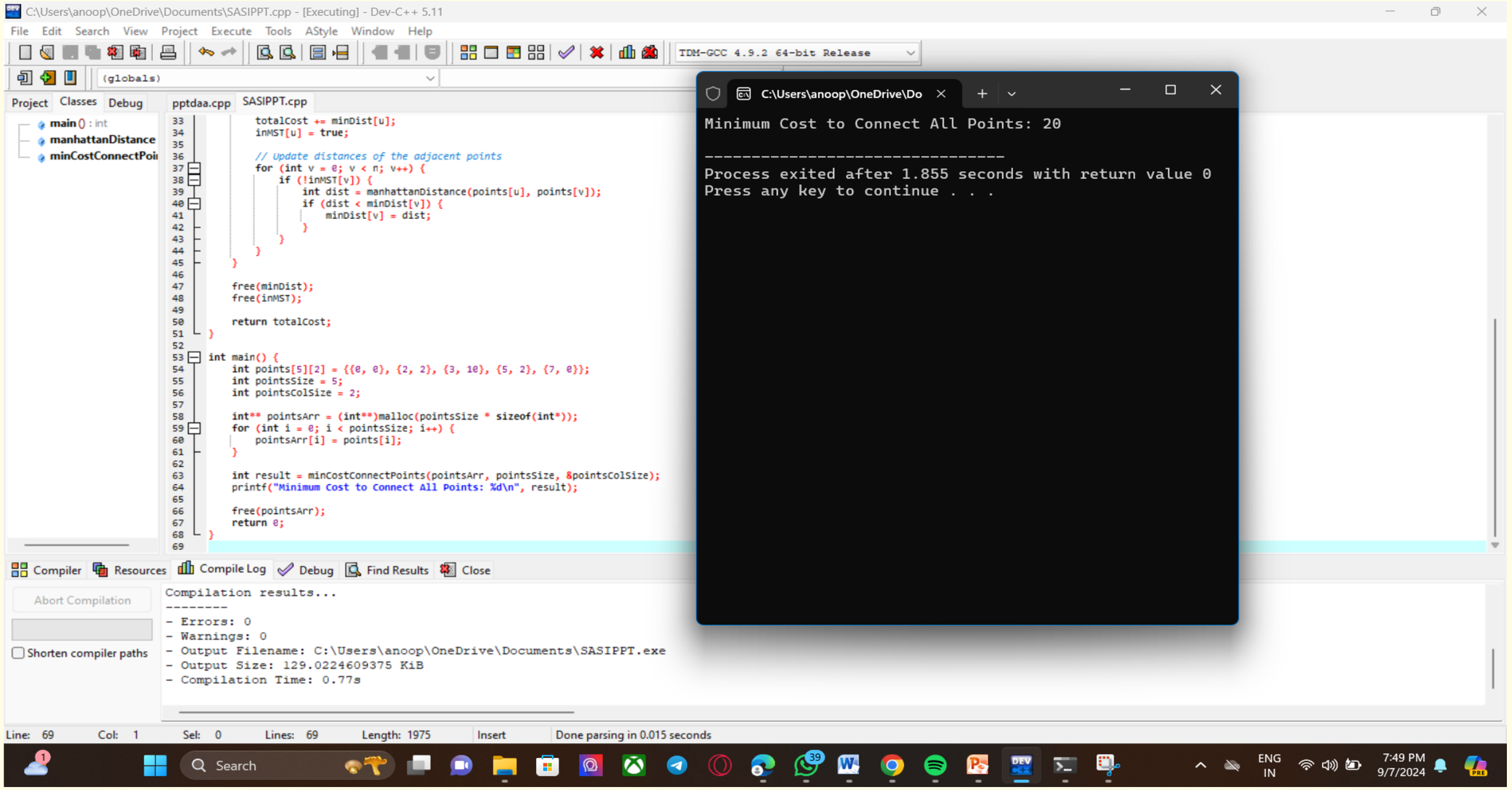
- Given that the distance metric between two points [xi,yi][x_i, y_i][xi,yi] and [xj,yj][x_j, y_j][xj,yj] is the Manhattan distance, calculated as:

- Manhattan Distance=∣xi−xj∣+∣yi−yj∣\text{Manhattan Distance} = |x_i - x_j| + |y_i - y_j|Manhattan Distance=∣xi−xj∣+∣yi−yj∣We can model the points as a complete graph where each edge between two points has a weight equal to the Manhattan distance between them. The task is to find the minimum cost to connect all points using a Minimum Spanning Tree (MST) algorithm, such as Prim's or Kruskal's algorithm.

ALGORITHM:

- **Input:** A list of points points[i]=[xi,yi]\text{points}[i] = [x_i, y_i]points[i]=[xi,yi], where each point represents a coordinate on the 2D plane.
- **Modeling as a Graph:** Treat each point as a node, and the edge between any two nodes has a weight equal to the Manhattan distance.
- **Prim's Algorithm:**Initialize the MST with one random point
- Use a priority queue (min-heap) to select the edge with the minimum cost.
- Add the selected point to the MST and update the priority queue with the distances of the newly added point to all remaining points.
- Continue until all points are added to the MST.
- **Return the total cost of the MST.**

CODING AND OUTPUT:



- **Worst Case:**
- **Time Complexity**: In the worst-case scenario, all points are distinct and must be connected. Each iteration of Prim's algorithm takes O(n)O(n)O(n) time to find the closest point to the MST, and the Manhattan distance between every pair of points is calculated, resulting in a time complexity of O(n2)O(n^2)O(n2).
- **Space Complexity**: We store the distances of all points and a priority queue to select the smallest edge, resulting in O(n)O(n)O(n) space complexity for the distance array and other auxiliary data structures.

- **Best Case:**
- **Time Complexity**: The best-case scenario occurs when all points lie on a straight line, reducing the number of necessary connections. However, the algorithm still needs to compute distances between every pair of points, resulting in the same time complexity O(n2)O(n^2)O(n2).
- **Space Complexity**: The space complexity remains O(n)O(n)O(n) since we need to store distances and other auxiliary data structures.

- **Average Case:**
- **Time Complexity**: In the average case, where points are randomly distributed in the plane, the time complexity will still be dominated by the distance calculations, resulting in O(n2)O(n^2)O(n2) complexity.
- **Space Complexity**: The space complexity is O(n)O(n)O(n), as we need to store auxiliary arrays and data structures to compute the MST.

FUTURE CASE: In the future, this problem could serve as a basis for more complex optimizations in fields like networking, logistics, and urban planning. For example, the algorithm could be adapted to more sophisticated models of transportation networks, where factors like congestion, capacity, and dynamic costs are involved. Additionally, this problem connects to real-world scenarios in wireless communication, where minimizing the cost of connecting nodes in a sensor network or designing efficient transportation paths for vehicles in smart cities could leverage similar principles. In data science, it can be applied to clustering problems, where the objective is to group similar data points with minimum connection costs, forming cohesive networks or communities with minimal expenses.

CONCLUSION:

- The problem of finding the minimum cost to connect all points in a 2D plane using the Manhattan distance can be effectively solved using Prim’s algorithm for computing the Minimum Spanning Tree (MST).
- The algorithm iteratively adds the minimum-cost edges while avoiding cycles, ensuring that the points are connected in a simple path.
- While the algorithm has a time complexity of O(n2)O(n^2)O(n2), which is feasible for moderate input sizes, it can be further optimized using more advanced data structures like Fibonacci heaps.
- Kruskal's algorithm with a Union-Find data structure for larger datasets. The consistent time complexity across best, worst, and average cases reflects the need to explore all pairs of points to guarantee the minimum connection cost.


THANK
YOU