To solve the rook-path problem, we must first represent our array of data points as a graph. This could be done with an adjacency matrix or adjacency list, but for the sake of efficiency, the latter should be used. This adjacency list would initially consist only of nodes, as the array provides no information on edges. However, these edges can be defined by connecting nodes that have the same X value or Y value. Then, with the graph constructed, we can then perform Dijkstra’s Algorithm, using a defined start and end point, to determine the shortest path in the graph.

**Input:** *data*: array of data points

**Input:** *n*: number of data points

1. **Algorithm:** ShortestRookPath
2. Create an empty adjacency list *adj*, whose nodes have an x and y value
3. Add each element in *data* to *adj*, storing the element’s x and y value
4. **for** *j* = 0 to *n*
5. **for** *k* = 0 to *n*
6. // End iteration if comparing node to itself
7. **if** adj[j] == adj[k] continue
8. // If two nodes have same x or y value
9. **if** adj[*j*].x == adj[*k*].x **or** adj[*j*].y == adj[*k*].y
10. Add an edge between nodes adj[*j*] and adj[*k*]
11. **end**
12. **end**
13. **end**
14. Use Dijkstra’s Algorithm on *adj* to determine shortest path from given start and end point, return array of shortest path