Traveling Salesman Problem

For my coding solution, I am using the priority queue to store each node, and at the beginning, I have to set up the root node which is my starting location. This node will be at level 0, then it will use the bond method to calculate and look for each row and columns min value and add them together as my fist reduced cost. Also, the bound method will make a new reduced matrix table to store in each node. The root node will then be added to the priority queue. Next, the program will enter the while loop and we will remove the root node from the priority queue. The program will use its vertex number to indicate it was already visited. The vertex number will be set to int i. This i will be used later to locate the rows in the matrix. The program will start to calculate each child’s new reduced bound and check which one is the lowest. The code is

Child.bound = min.bound + min.reducedMatrix[i][j] + bound(child)

The min.bound is the root node’s reduced bond, min.reducedMatrix[i][j] mean location i to j reduced cost, and bound (child) is the current node’s reduced cost. Then each node that is not visited before will be added into the priority queue. The node with the lowes bound will then be removed after the while loop cycle again. The node that got removed will be marked as visited and saved to the path. The program will continue the process until it reaches the last level. The if loop

(min.level == inputQuestion.Length -1)

Will store all the visited path to length and also add 0 to the path, which means the salesperson return to the starting point. Also, it will store the last visited node’s bond as the total length. The time complexity of this problem is very difficult to find any information from the textbook. I think as my input question has 5 vertexes, and first, it will compare 4 vertexes bound then 3 then 2 then 1 to 0. The total compare number is 10. I think the time complexity might be O.