```java
// Function to calculate minimum cost for Travelling Salesman problem  using dynamic programming.
// finding the minimum cost path staring from 1 and ending to i
// and all vertices appearing exactly once and the cost of this path be cost(i)
// the cost of a cycle would be cost(i) + dist(i, 1) where dist(i, 1) is the distance from i to 1
public int calculateTravellingSalesManCost(int[][] cost) {
    int n = cost.length;
    // maintaining a set for lower tree.
    StringBuilder set = new StringBuilder();
    for (int i = 1; i < n; i++) {
        set.append(String.valueOf(i));
    }
    return travellingSalesMan(cost, set, 0, Integer.MAX_VALUE);
}

public int travellingSalesMan(int[][] graph, StringBuilder set, int node, int min) {
    if (set.length() == 0) {
        return graph[node][0];
    }
    for (int i = 0; i < set.length(); i++) {
        String s = set.substring(0, i) + set.substring(i + 1, set.length());
        StringBuilder newSet = new StringBuilder(s);
        int cost = graph[node][set.charAt(i) - '0'] +
                travellingSalesMan(graph, newSet, set.charAt(i) - '0', min);
        min = Math.min(min, cost);
    }
    return min;
}
```