

**ALGORITHMS AND PROBLEM SOLVING LAB**

**GUIDED BY: DR. BHARAT GUPTA**

Travelling Salesman Problem

MEMBERS:

TANAY MISHRA 16103159

SHREYAS JAIN 16103200

KARAN NEGI 16103062

RHYTHM MALHOTRA 16103156

Content

1. Introduction

2. Description of Algorithms

A. Dynamic Programming

B. Greedy

4. Results and Analysis

A.Time Comparison

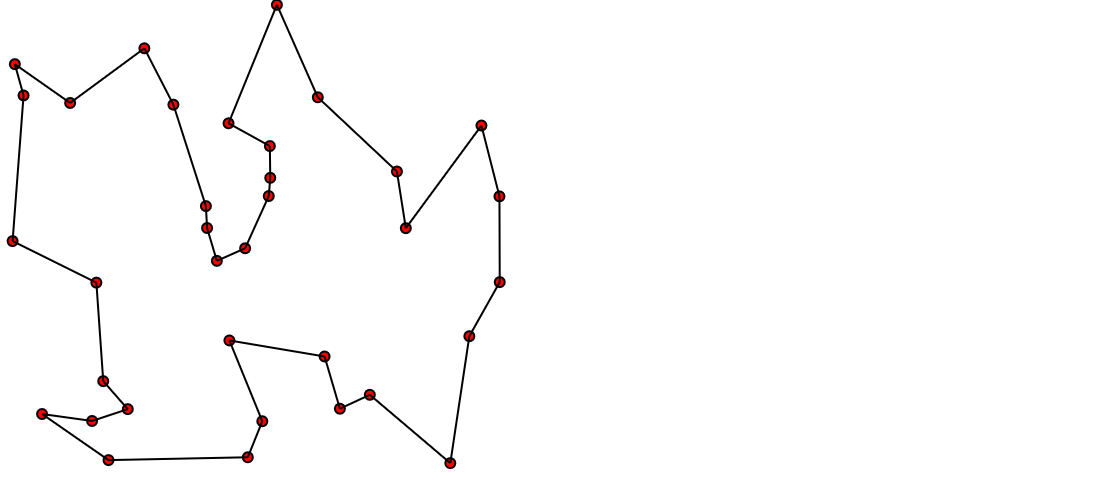
# 1. Introduction and Utility of the project

The Travelling Salesman Problem (TSP) is a deceptively simple

combinatorial problem. It can be stated very simply: A salesman spends his time visiting N cities (or nodes) cyclically. In one tour he visits each city just once, and finishes up where he started. In what order should he visit them to minimize the distance traveled? TSP is applied in many different places such as warehousing, material handling and facility planning.

The travelling salesman problem (TSP) can be stated as follows:

Given:

* Complete undirected graph G(V, E)
* Metric edge cost Ce >=0 

Problem:

* Find a hamiltonian cycle with minimal cost.

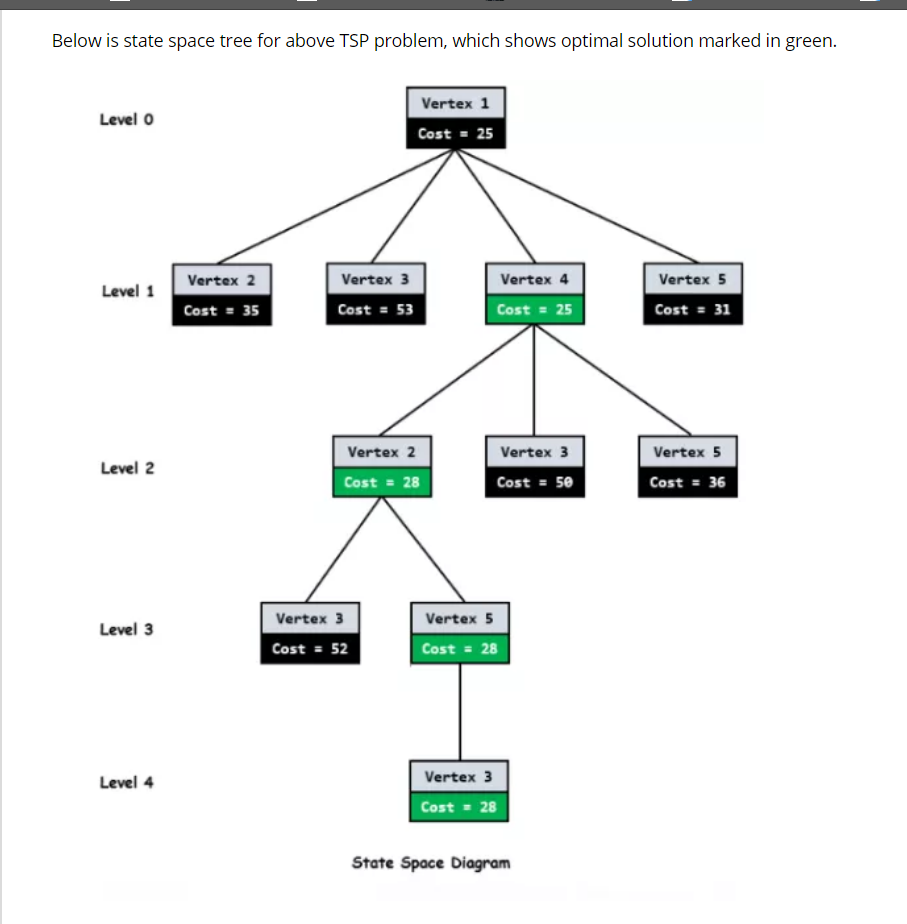
Although optimal algorithms exist for solving the TSP, for large-size TSP, as following Brute Force and Dynamic Programming algorithms show that, it is almost impossible to generate an optimal solution within a reasonable amount of time. Heuristics, instead of optimal algorithms, are extensively used to solve such problems. People conceived many heuristic algorithms to get near-optimal solutions. Here we implement Greedy and Branch and Bound and analyzed the result.

# 2. Description of Algorithms

**2.A (Branch and Bound)**

A branch-and-bound algorithm consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidate s are discarded, by using upper and lower estimated bounds of the quantity being optimized. The Branch and Bound strategy divides a problem to be solved into a number of sub-problems. It is a system for solving a sequence of sub problems each of which may have multiple possible solutions and where the solution chosen for one sub-problem may affect the possible solutions of later sub-problems. Suppose it is required to minimize an objective function. Suppose that we have a method for getting a lower bound on the cost of any solution among those in the set of solutions represented by some subset. If the best solution found so far costs less than the lower bound for this subset, we need not explore this subset at all. Let s be some subset of solutions. L(S)=a lower bound on the cost of any solution belonging to S Let C=cost of the best solution found so far If C ≤ L(S), there is no need to explore S because it does not contain any better solution. If C > L(S), then we need to explore S because it may contain a better solution

**Pseudo code:**

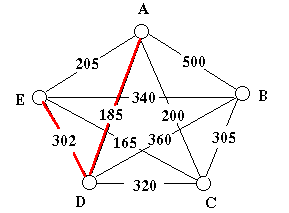
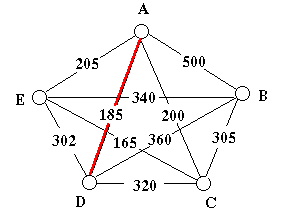


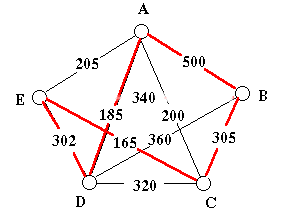
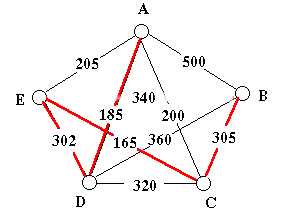
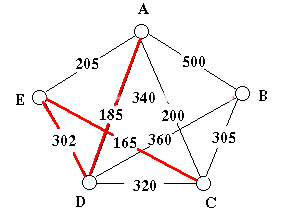
## 2.B Greedy (Nearest Neighbor)

In TSP, we usually need an efficient way to obtain a suboptimal result for other improvement algorithms. This is where Greedy is used.

Greedy algorithm is the simplest algorithm. Greedy algorithm is starts with the departure Node 1. Then the algorithm calculates all the distances to other n-1 nodes. Go to the next closest node. Take the current node as the departing node, and select the next nearest node from the remaining n-2 nodes. The process continues until all the nodes are visited once and only once then back to Node 1. When the algorithm is terminated, the sequence is returned as the best tour; and its associated OFV is the final solution.

For example,



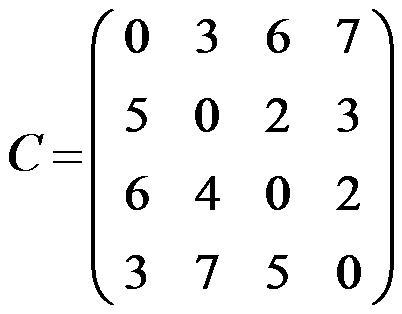
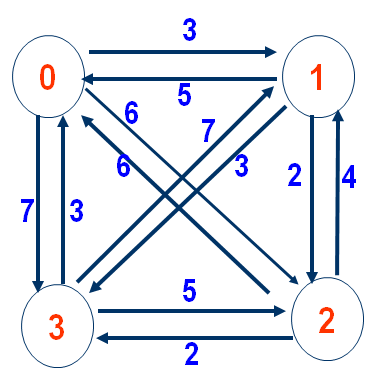


We start from Node A, then we select a shortest path A-D and go to D point. Repeat this process and after we go through nodes, we return to A point. Finally we get a suboptimal solution A-D-E-C-B-A.

## 2.C Dynamic Programming (although not used in project)

Assume that there is a TSP problem as follow.

There are 4 cities, 0 1 2 3, shown as picture. The adjacency matrix represents the distance between cities.



### 1. Why DP can work?

Assume that {s, s1 ,s2 , … , sp, s} is an optimal circle. if dist(s,s1) is known, then problem becomes to find optimal path from s1 to s. If dist(s1,s2) is known, then problem becomes to find optimal path from s2 to s. Obviously, it’s a optimize substructure problem.

### 2. DP equation

Assume start from s，d(i, V’) represent the optimal weight of the circle, which go from vertex i, go through all vertices in V’ (a set) once and only once, then go back to s.

There are two scenarios:

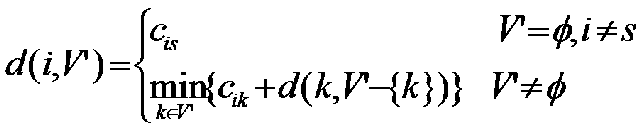
① if V’ is empty，d(i, V’) represents go from i ,go through nothing and go back to s. In this scenario,

d(i, V’)=C(i,s) (distance between i and s).

② if V’ is not empty，it comes to optimize a sub-problem. Try every vertices in set V’, choose the best (minimum) one.

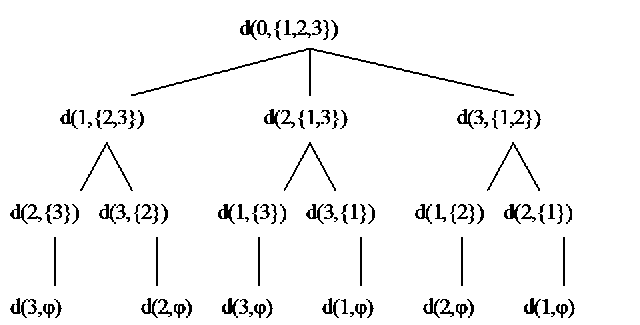
d(i, V’)=min{C(i,k) + d(k, V’-{k})}

Ps : d(k, V’-{k}) is a sub-problem



### 3. Case study

Now, use the example mentioned before to explain the process of calculation of DP equation. (Assume start from City/Vertex 0)



①The final result we want is d(0,{1,2,3}) representing the cost of shortest path, which start from City 0, go through City 1/2/3 and go back to City 0.

②d(0,{1,2,3}) can not be calculate immediately and straight forward. As the second level of the “tree” picture shows:

d(0,{1,2,3})=min {

C01+d(1,{2,3})

C02+d{2,{1,3}}

C03+d{3,{1,2}}

}

③the same as d(0,{1,2,3}), d(1,{2,3})，d(2,{1,3})，d(3,{1,2})can not be calculate immediately and straight forward either.

d(1,{2,3})=min{

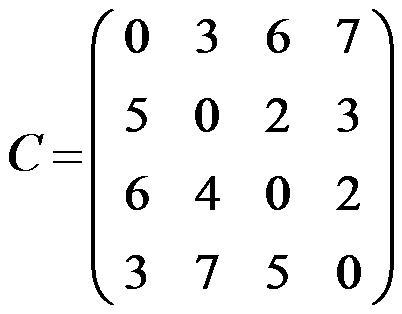
C12+d(2,{3})

C13+d(3,{2})

}

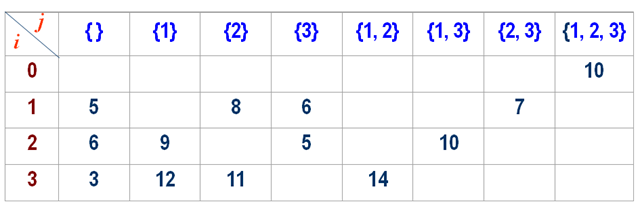
d(2,{1,3})，d(3,{1,2}) are calculated by same process.

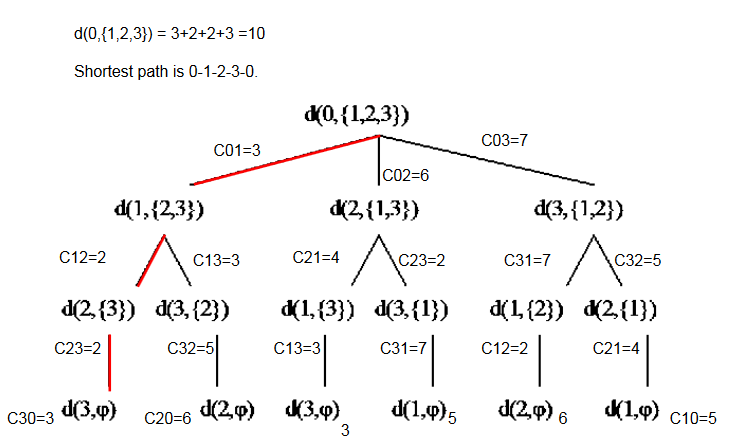
④Based on this equation, only in the last level, the set V’ is empty , Cis can be known directly from adjacency matrix as follow.



4. Calculation process

Represent d(i, V’) as 2D-table, d[i][j] :



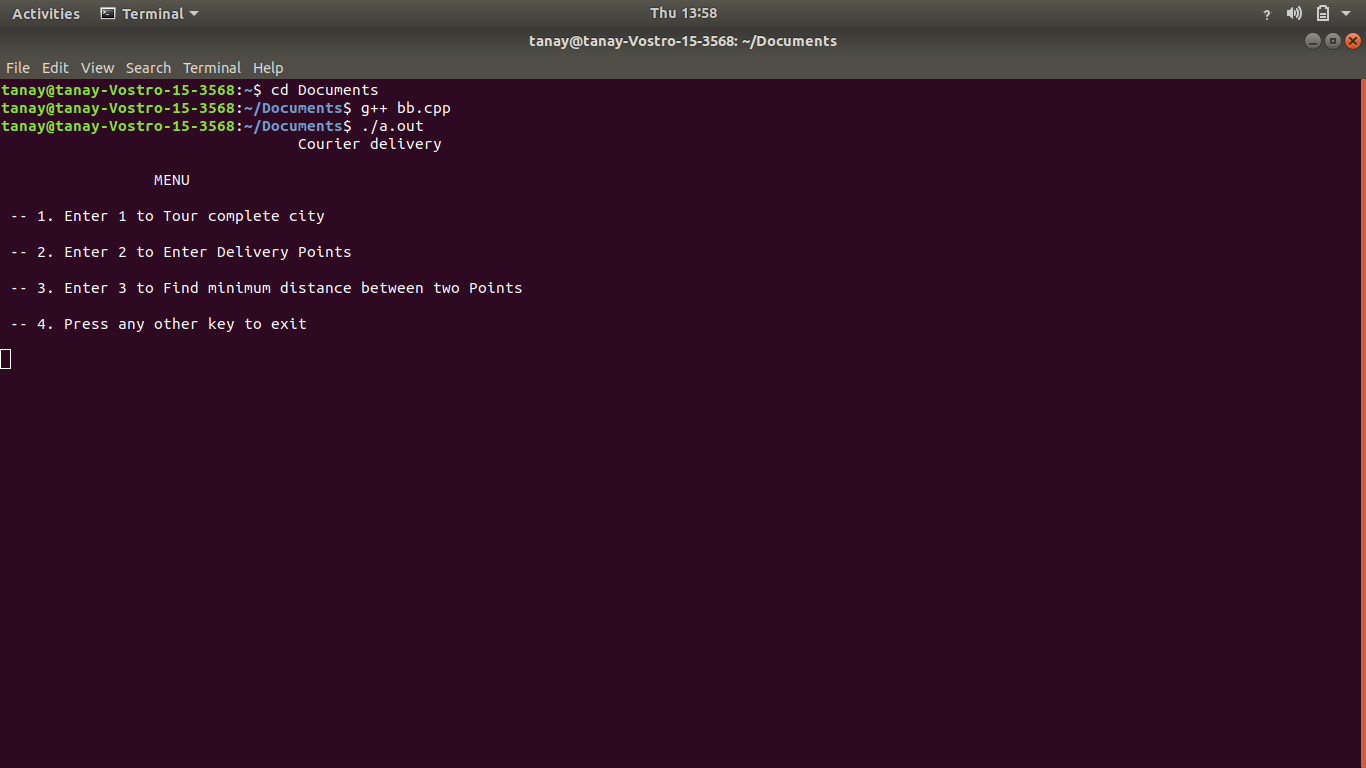


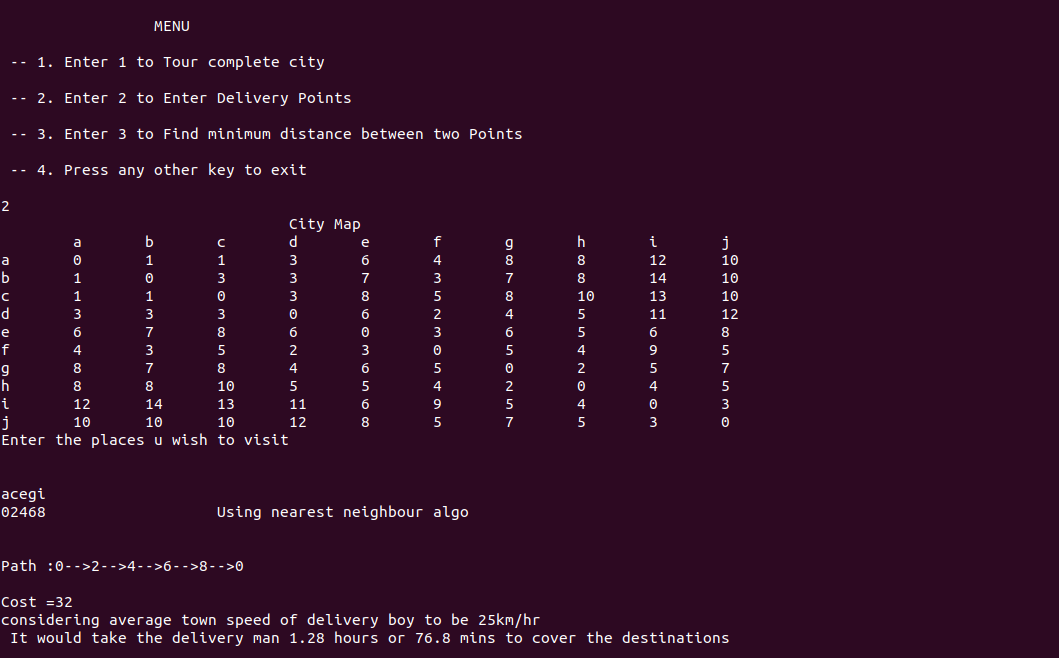
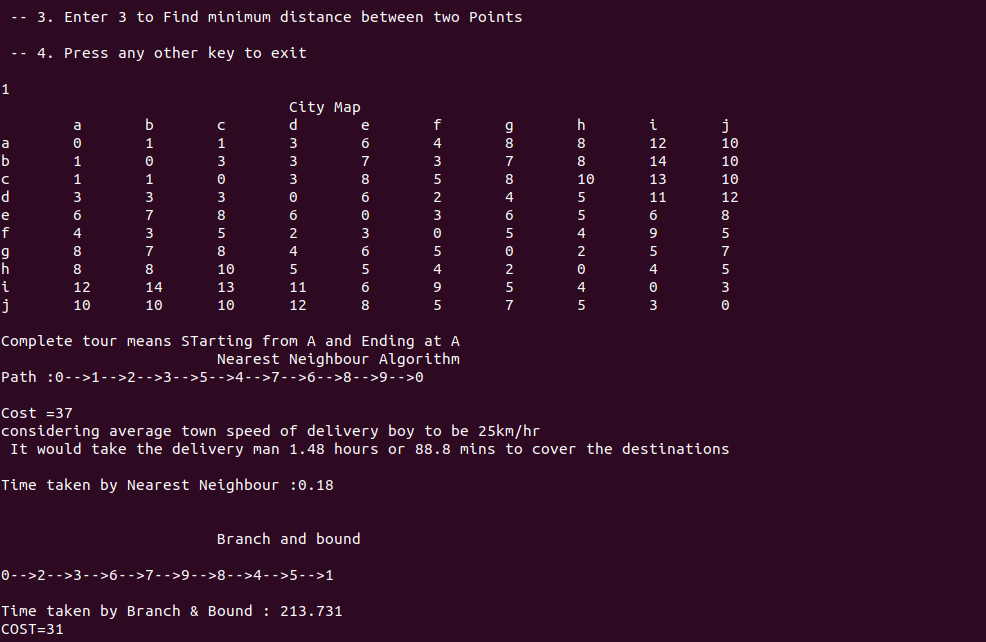
# 

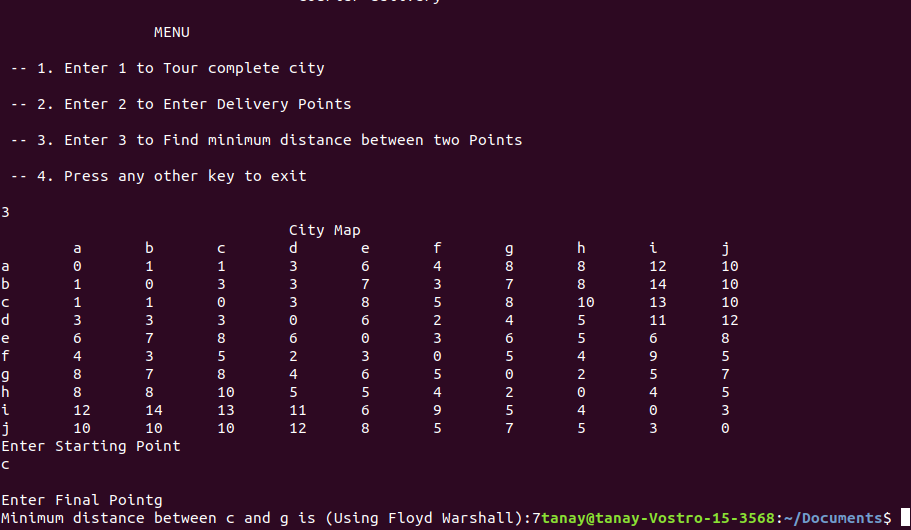
# 4. Results and Analysis.

## 4.A Time Comparison

According to the output, greedy algorithm is taking less time than the branch and bound algorithm used.







References :

<http://www.universalteacherpublications.com/univ/ebooks/or/ch6/travsales.htm>