UPES

UNIVERSITY WITH A PURPOSE

Report for Minor I

Submitted by

500069734	R134218123	Pratham Pandey
500068183	R134218125	Pulkit Mittal
500067543	R134218152	Shashwat Chitransh
500068730	R134218206	Karmanya Dadhich

Under the mentorship of

Dr. Jitendra Rajpurohit

Department of Systemics

Finding Optimal Route

A practical implementation of The Travelling Salesman Problem using Dynamic Programming

INDEX

Abstract

The project aims at viewing real life problems and how they can be solved by using some simple algorithms that do wonders. One such real life scenario is to solve the Travelling Salesman Problem(TSP). It was mathematically formulated in the 1800s by the Irish mathematician W.R. Hamilton and by the British mathematician Thomas Kirkman.

TSP asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" It is an NP-hard problem in combinatorial optimization, important in theoretical computer science and operations research.

In our project we'll be tackling the travelling salesman problem using dynamic programming approach to devise an optimal path for a Pizza delivery-person. House addresses and distances will be provided as input and the algorithm will work to get an overall path the delivery-person should take for the maximum efficiency.

The entire project will be made in C as instructed and our model will help our Pizza delivery guy.

Keywords: *Travel Salesman Problem, TSP, Local Search Algorithm, Hamiltonian Path*

Introduction

When solving optimization problems with computers, often the only possible approach is to calculate every possible solution and then choose the best of those as the answer. Unfortunately, some problems have such large solution spaces that this is impossible to do. These are problems where the solution cannot be found in a reasonable time. These problems are referred to as NP-hard or NP-complete problems. One such problem is The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) is a problem taken from a real-life analogy. Consider a salesman who needs to visit many cities for his job. Naturally, he would want to take the shortest route through all the cities. The problem is to find this shortest route without taking years of computation time.

Two principal reasons that make this topic a paramount one in the field: their inherent practical nature and social interest, which allow routing problems to be applicable not only in leisure or tourism scenarios, but also in situations related to logistics and business; and their complexity, making such problems very difficult to optimally solve them even for medium-sized datasets.

So, we'll be using the DP approach to find the optimal path by taking distances as an input from the user and giving him the best possible path suitable for him. In real world it is applicable to many scenarios like finding the best possible path for a Pizza delivery boy who has to deliver Pizzas to customer living at different location so that his time and money can be saved and the customer can also be satisfied with the services.

Suppose we start at Location 1 and after visiting some locations now we are at location 'x'. We certainly need to know 'x', since this will determine which locations are most convenient to visit next. We also need to know all the locations visited so far, so that we don't repeat any of them. Hence, this is an appropriate sub-problem.

Related Works

[1] D. L., Bixby found how these dual bound predictions can be improved by including domain propagation into strong branching.

[2] Cormen T. H., Leiserson helped in solving a well-specified computational problem. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship.

[3] W.J. Cook has contributed in the field of Computer Science in Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation, Traveling Salesman Problem: A case study in local optimization by David S. Johnson and Lyle A. McGeoch1995.

[4] V.Chv´atal, Cutting planes lie at the heart of the branch and cut method used by efficient solvers for the traveling salesman problem. Between 1988 and 2005, the team of David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook developed one such solver, Concorde.

Methodology

A salesman needs to visit 'n' cities in such a manner that all destination cities must be visited at once and in the end, he returns to the city from where he started with minimum cost. Suppose the cities are x_1, x_2..., x_n, where cost denotes the cost of travelling from city x_i to x_j. The travelling salesman problem is to find a route starting and ending at x_1 that will take in all the cities with the minimum cost.

Prerequisites:

- **Data Source or Representation using**: Data of Places and the Distances using Matrix.
- **Approach:** Dynamic Programming.
- **General Formula used**:

$$g(i, s) = min_{K \in S}\{C_{iK} + g(K, S - \{K\})\}$$

- **Mathematical model:**

1) Consider or choose the starting and ending point.

2) For every other vertex i other than the chosen one, find the minimum cost path with chosen vertex as the starting point, i as the ending point making sure all vertices appearing exactly once.

3) Finally, return the minimum of all [cost(i) + dist (i, chosen starting point)] values.

4) Find cost(i), define a term C (S, i) be the cost of the minimum cost path visiting each vertex in set S exactly once, starting vertex and ending at i.

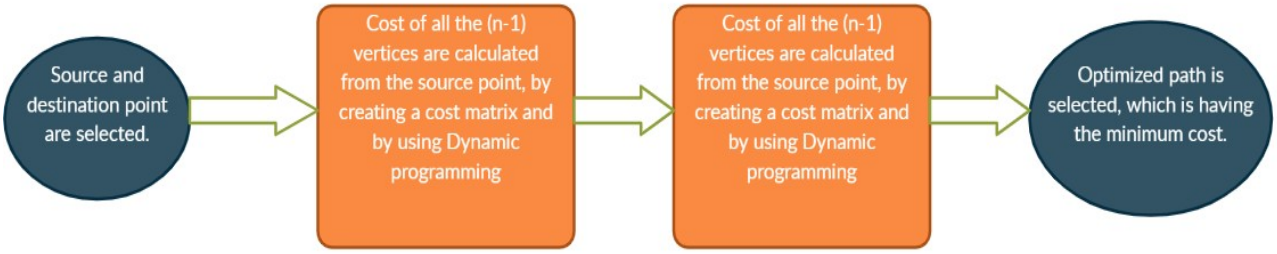
5) Start with all subsets of size 2 and calculate C(S, i) for all subsets where S is the subset.

6) Calculate C(S, i) for all subsets S of size 3 and so on.

7) If size of S is 2, then S must be {chosen starting point, i},C(S, i) = dist(starting point, i).

8) Else if size of S is greater than 2, C(S, i) = min { C(S-{i}, j) + dis(j, i)} where j belongs to S, j != i and j != 1.
9) For a set of size n, consider n-2 subsets each of size n-1 such that all subsets don't have nth in them.
10) Using the above recurrence relation, write dynamic programming-based solution.

- **Diagrammatic Representation:**



Steps Involved

System Requirements:

Hardware:

- **RAM:** 2GB
- **Disk Space:** 4GB

Software:

Any C compiler (GCC preferred)

OS:

Windows 7/10, MacOS (>10.0), Ubuntu 3.0.2

Flowchart


START
Choose Input Mode
ManualInput
File Handling
Input text file from user
Take the numberof addresses and their corresponding distances as input from user
Display cost matrix to user
Take current address as input from user
Calculate the shortest path
File Handling
YES
Create an O/P file in the same directory
End
NO
Display the shortest path
End

Program

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
# define min 9999
int number_of_cities,  cost = 0, count = 0;
static int visited[15];

int **input_matrix;
char **arr;

int **input(){

printf("**********************************************************
**********************************************\n");
   printf("--------------------------------PIZZA DELIVERY OPTIMAL
PATH FINDER---------------------------------\n");

printf("**********************************************************
**********************************************\n");
        printf("Enter The Number of Locations :\n");
        scanf("%d",&number_of_cities);

        input_matrix = (int**)malloc(number_of_cities * sizeof(int *));
        arr=(char**)malloc(number_of_cities*sizeof(char*));
        for(int i=0;i<number_of_cities;i++)
        {
          arr[i]=(char*)malloc(10*sizeof(char));
        }
        printf("Enter The Name of Locations :\n");
        for(int i=0;i<number_of_cities;i++)
        {
```

```
        scanf("%s",arr[i]);
        }

        for(int k = 0; k < number_of_cities; k++){
                input_matrix[k] = (int*)malloc(number_of_cities * sizeof(int));
        }

        for(int i =0; i < number_of_cities; i++){
                for(int j = 0; j < number_of_cities; j++){
                        if(i == j){
                                *(*(input_matrix + i) + j) = 0;

                        }
                        else{
                        printf("Enter The Distance Between Location %s And %s
:\n",arr[i],arr[j]);
                        scanf("%d",&*(*(input_matrix + i) + j));
                }

                }
        }
        return input_matrix;
}

int next_min(int city)
{
        int min_cost = min;
        int next_city = 0;
        for(int i = 0; i < number_of_cities; i++)
        {
                if((input_matrix[i][city] != 0)  && (visited[i] != 1))
                        {
                                if(input_matrix[city][i] < min_cost)
                                        {
                                          min_cost = input_matrix[city][i];
```

```
//+ input_matrix[i][0];
                                        next_city = i;
                                    }
                        }
        }
        return next_city ;
}

void mincost(int root_city)
{
        int next_city = 0;
        visited[root_city] = 1;
        printf("%s ---> ", arr[root_city]);
        next_city = next_min(root_city );
        if(next_city < number_of_cities && count < number_of_cities - 1)
                {
                        count += 1;
                        mincost(next_city);
                }
}

int main()
{
   int c;
        int **input_matrix1;
        int root;
   printf("\nEnter 0 To Perform Input/Output Manually :\n");
   printf("Enter 1 To Perform Input/Output Using File Handling :\n");
   scanf("%d",&c);
   if(c==1){
      char finame[20];
      char foname[20];
      printf("Enter The Name Of The Input File :\n");
      scanf("%s",finame);
      printf("Enter The Name Of The Output File :\n");
      scanf("%s",foname);
```

```
FILE *f;
f=freopen(finame,"r",stdin);
f=freopen(foname, "w+", stdout);
input_matrix1 = input();
printf("\n\n\nDistance Matrix :\n\n");
for(int i = 0; i < number_of_cities; i++)
{
   printf("[ ");
   for(int j = 0; j < number_of_cities; j++)
   {
      printf("%d",input_matrix1[i][j]);
      if(j == number_of_cities - 1)
      {
         printf(" ]");
         printf("\n");
         continue;
      }
      printf("\t");
   }
}

while(1){
   for(int i=0;i<number_of_cities;i++)
      {
         printf("\nPress  %d TO Select Starting Location As :
%s",(i),arr[i]);
      }
   printf("\n");
   scanf("\n%d",&root);
   if(root>(number_of_cities-1))
      {
      printf("You Have Entered The Wrong Choice!!\n");
      printf("Please Try Again!!\n");
      continue;
      }
```

```
            else
                {
                printf("\nThe Best Path For Pizza Delivery :\n");
                mincost(root);
                printf("%s\n", arr[root]);
                printf("\n");

printf("**********************************************************************
**********************************************\n");
                break;
                }
        }
        fclose(f);
    }
    else if(c==0){
        input_matrix1 = input();
        printf("\n\n\nDistance Matrix :\n\n");
        for(int i = 0; i < number_of_cities; i++)
        {
            printf("[ ");
            for(int j = 0; j < number_of_cities; j++)
            {
                printf("%d",input_matrix1[i][j]);
                if(j == number_of_cities - 1)
                {
                    printf(" ]");
                    printf("\n");
                    continue;
                }
                printf("\t");
            }
        }

        while(1){
            for(int i=0;i<number_of_cities;i++)
                {
```

```
                printf("\nPress  %d TO Select Starting Location As :
%s",(i),arr[i]);
                }
            printf("\n");
            scanf("\n%d",&root);
            if(root>(number_of_cities-1))
                {
                printf("You Have Entered The Wrong Choice!!\n");
                printf("Please Try Again!!\n");
                continue;
                }
            else
                {
                printf("\nThe Best Path For Pizza Delivery :\n");
                mincost(root);
                printf("%s\n", arr[root]);
                printf("\n");

printf("**********************************************************************
**********************************************\n");
                break;
                }
        }
    }
        return 0;
}
```

Output

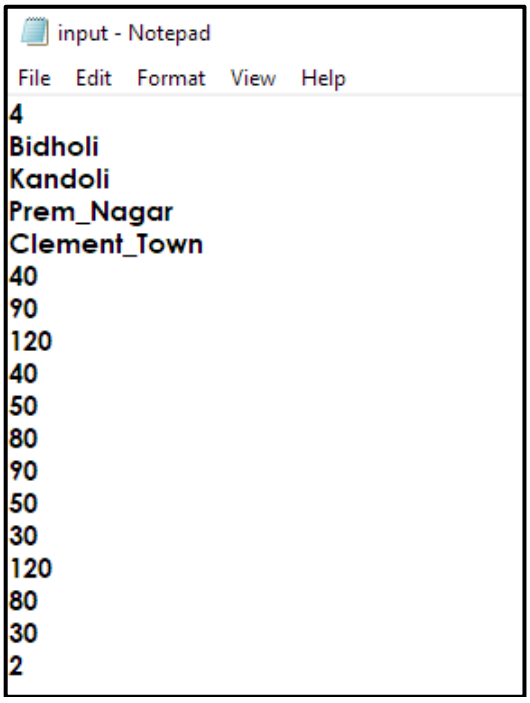
- **Without File Handling:**

```
Enter 0 To Perform Input/Output Manually :
Enter 1 To Perform Input/Output Using File Handling :
0
********************************************************************************************************************
---------------------------------PIZZA DELIVERY OPTIMAL PATH FINDER---------------------------------
********************************************************************************************************************
Enter The Number of Locations :
4
Enter The Name of Locations :
Bidholi
Kandoli
Prem_Nagar
Clement_Town
Enter The Distance Between Location Bidholi And Kandoli :
40
Enter The Distance Between Location Bidholi And Prem_Nagar :
90
Enter The Distance Between Location Bidholi And Clement_Town :
120
Enter The Distance Between Location Kandoli And Bidholi :
40
Enter The Distance Between Location Kandoli And Prem_Nagar :
50
Enter The Distance Between Location Kandoli And Clement_Town :
80
Enter The Distance Between Location Prem_Nagar And Bidholi :
90
Enter The Distance Between Location Prem_Nagar And Kandoli :
50
Enter The Distance Between Location Prem_Nagar And Clement_Town :
30
Enter The Distance Between Location Clement_Town And Bidholi :
120
Enter The Distance Between Location Clement_Town And Kandoli :
80
Enter The Distance Between Location Clement_Town And Prem_Nagar :
30
```

```
Distance Matrix :

[ 0     40      90      120 ]
[ 40    0       50      80 ]
[ 90    50      0       30 ]
[ 120   80      30      0 ]

Press  0 TO Select Starting Location As : Bidholi
Press  1 TO Select Starting Location As : Kandoli
Press  2 TO Select Starting Location As : Prem_Nagar
Press  3 TO Select Starting Location As : Clement_Town
2

The Best Path For Pizza Delivery :
Prem_Nagar ---> Clement_Town ---> Kandoli ---> Bidholi ---> Prem_Nagar

********************************************************************************************************************

--------------------------------
Process exited after 200.5 seconds with return value 0
Press any key to continue . . .
```

- **With File Handling:**

```
Enter 0 To Perform Input/Output Manually :
Enter 1 To Perform Input/Output Using File Handling :
1
Enter The Name Of The Input File :
input.txt
Enter The Name Of The Output File :
output.txt

--------------------------------
Process exited after 28.59 seconds with return value 0
Press any key to continue . . .
```



input.txt

output - Notepad

File Edit Format View Help

```
**********************************************************************************************************
----------------------------------PIZZA DELIVERY OPTIMAL PATH FINDER----------------------------------
**********************************************************************************************************
Enter The Number of Locations :
Enter The Name of Locations :
Enter The Distance Between Location Bidholi And Kandoli :
Enter The Distance Between Location Bidholi And Prem_Nagar :
Enter The Distance Between Location Bidholi And Clement_Town :
Enter The Distance Between Location Kandoli And Bidholi :
Enter The Distance Between Location Kandoli And Prem_Nagar :
Enter The Distance Between Location Kandoli And Clement_Town :
Enter The Distance Between Location Prem_Nagar And Bidholi :
Enter The Distance Between Location Prem_Nagar And Kandoli :
Enter The Distance Between Location Prem_Nagar And Clement_Town :
Enter The Distance Between Location Clement_Town And Bidholi :
Enter The Distance Between Location Clement_Town And Kandoli :
Enter The Distance Between Location Clement_Town And Prem_Nagar :



Distance Matrix :

[ 0         40          90          120 ]
[ 40        0           50          80 ]
[ 90        50          0           30 ]
[ 120       80          30          0 ]

Press  0 TO Select Starting Location As : Bidholi
Press  1 TO Select Starting Location As : Kandoli
Press  2 TO Select Starting Location As : Prem_Nagar
Press  3 TO Select Starting Location As : Clement_Town

The Best Path For Pizza Delivery :
Prem_Nagar ---> Clement_Town ---> Kandoli ---> Bidholi ---> Prem_Nagar

**********************************************************************************************************
```

output.txt

References

[1] D. L. Bixby, a new algorithm for finding a fuzzy optimal solution for fuzzy transportation problems, Applied mathematical sciences, 4, 79-90, 2012

[2] Cormen T. H., Design and Analysis of Algorithm, Operation research theory and application, Third Edition, 2007. [7] Zadeh L.A., Fuzzy sets Information and Control, 8, 3, 338-353, 1965.

[3] (2016) W.J. Cook, Tropical Complexity, Sidon Sets, and Dynamic Programming. SIAM Journal on Discrete Mathematics 30:4, 2064-2085.

[4] The Travelling Salesman Problem and Minimum Matching in the Unit Square. SIAM Journal on Computing 12:1, 144-156

[5] https://www.researchgate.net/publication/341123222_Traveling_salesman_problem_a_perspective_review_of_recent_research_and_new_results_with_bio-inspired_metaheuristic

[6] https://protosity.wordpress.com/2015/12/12/travelling-salesman-problem-and-genetic-algorithms/#