E-COMMERCE DELIVERY ROUTING SYSTEM USING OPTIMIZED ALGORITHM FOR LOGISTICS STAFF

PROJECT REPORT: MINOR 1

submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with

Specialization in

Business Analytics and Optimization

Submitted by:

Name	Roll No.
Sugandh Agarwal	R103216101
Naman Jain	R103216133
Utkarsh Vikram Singh	R103216111

Under the guidance of

Mr. Ravi Prakash

Assistant Professor

(Selection Grade)

Department of Informatics

Department of Informatics, School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Bidholi, Via Prem Nagar, Dehradun, Uttarakhand

2018-19

UNIVERSITY WITH A PURPOSE

CANDIDATES' DECLARATION

I/We hereby certify that the project work entitled **E-Commerce Delivery Routing System Using Optimized Algorithm for Logistics Staff** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science And Engineering with Specialization in Business Analytics and Optimization, and submitted to the Department of Informatics at School of Computer Science, University of Petroleum And Energy Studies, Dehradun, is an authentic record of my/our work carried out during a period from **August, 2018** to **December, 2018** under the supervision of **Mr. Ravi Prakash, Assistant Professor (Selection Grade), Department of Informatics.**

The matter presented in this project has not been submitted by me/us for the award of any other degree of this or any other University.

Sugandh Agarwal	**(R103216101)**
Naman Jain	**(R103216133)**
Utkarsh Vikram Singh	**(R103216111)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(Date: 17th December 2018)

Mr. Ravi Prakash
(Project Guide)

Dr. TP Singh
Head of Department
Department of Informatics,
School of Computer Science,
University of Petroleum and Energy Studies, Dehradun- 248001 (Uttarakhand)

ACKNOWLEDGEMENT

We wish to express our deep gratitude to our guide **Mr. Ravi Prakash**, for all advice, encouragement, and constant support he has given us throughout our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, **Dr. TP Singh**, for his great support in doing our **project E-Commerce Delivery Routing System Using Optimized Algorithm for Logistics Staff** at **SoCS**.

We are also grateful to **Dr. Manish Prateek, Professor and Director, SoCS** for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

Sugandh Agarwal	**Naman Jain**	**Utkarsh Vikram Singh**
R103216101	**R103216133**	**R103216111**

School of Computer Science

University of Petroleum & Energy Studies, Dehradun

Minor I

PROJECT TITLE:

E-COMMERCE DELIVERY ROUTING SYSTEM USING OPTIMIZED ALGORITHM FOR LOGISTICS STAFF

ABSTRACT

E-commerce organizations have a separate department to handle logistics for timely delivery of products ordered by their customers. Each delivery has a separate customization and delivery address. The process, after the order has been accepted and is to be dispatched, begins with collection of goods from the warehouse and doorstep delivery via optimized path calculation, factoring in the essential components like fuel economy, load, and quantity. Cash on delivery and pre-processed payment options are considered, and automatic division to appropriate mode of transport for the goods is carried out. Hence, this automation leads to profitability, and reduction in cost of resources being used. An accurate way to calculate this path is via the Travelling-Salesman Problem by W.R. Hamilton.

Keywords: Travelling-Salesman, Algorithms, Hamiltonian Cycle, E-commerce, Optimization

TABLE OF CONTENTS

INTRODUCTION

E-commerce organizations like Amazon are known to deliver their products with efficiency, and in a good state. It is because of the logistical support they establish to cover their regions. There are two main ways to conduct this task:

1. By establishing a delivery system of their own
2. Outsourcing the logistics services

Each way has its own advantages and disadvantages, but the common ground for them is optimization of the route they will choose to have day-to-day deliveries being made at the doorstep of the customer. It is easy to understand that reduced costing with improved quality is one of the main aims of each organization. Since delivery includes manpower, resources like fuel, and transport maintenance, a major part of funds goes to logistics. With ever improving competition to be the best supplier of goods, effective systems become a need of the hour to facilitate this venture.

In the travelling-salesman problem, which is closely related to the Hamiltonian cycle problem, a salesman must visit n cities. Modeling the problem as a complete graph with n vertices, we can say that the salesman wishes to make a tour, visiting each city exactly once and finishing at the city he starts from. The salesman incurs a nonnegative integer cost c(A,B) to travel from city A to city B, and he wishes for this cost to be minimum, where the total cost is the sum of the individual costs along the edges of the tour. [1]

Modifying the problem to sectors in a city and cost as fuel, the staff for logistics shall have an optimized path with economical fuel expense. Given the functionalities of programming language C, a possible implementation is with the defined complexity of O (n^2*2^n).

PROBLEM STATEMENT

The main challenge in doing this project is to optimize the cost of the logistics system by devising an algorithm which involves minimum space and time. We will take orders from multiple users, from a specific city divided into sectors, from a list of products provided by us, and take the respective addresses to calculate the optimized path of delivery.

LITERATURE REVIEW

Although the exact origins of the Traveling Salesman Problem are unclear, the first example of such a problem appeared in the German handbook Der Handlungsreisende - Von einem alten Commis - Voyageur for salesman traveling through Germany and Switzerland in 1832, as explained in [2]. This handbook was merely a guide, since it did not contain any mathematical language. People started to realize that the time one could save from creating optimal paths is not to be overlooked, and thus there is an advantage to figuring out how to create such optimal paths. This idea was turned into a puzzle sometime during the 1800's by W. R. Hamilton and Thomas Kirkman. Hamilton's Icosian Game was a recreational puzzle based on finding a Hamiltonian cycle in the Dodecahedron graph. [3]

The Traveling Salesman Problem, as we know it, was first studied in the 1930s in Vienna and Harvard as explained in [2]. Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of TSP. This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours. [3] [1] explains the mathematical computation with justification on TSP, and the algorithms identified for its implementation.

The modern logistics have become the most important means to improve the efficiency of material flow, reduce distribution costs in various industries; at the same time, the recent development of E-commerce also contributed to the expansion of the logistics market, promote the development of technologies related to logistics. Large numbers of practices have been carried out in the E-commerce logistics. [4] and [5] explain the importance, challenges, working, and the general implementation of the logistics and supply chain management.

[6] and [7] provides us valuable information on Software Engineering, and the Iterative Waterfall Model. This model is easy to understand, and reinforces the notion of "define before design" and "design before code." The model expects complete and accurate requirements early in the process. [6]

[8] is a direct resource for understanding Bellman-Held-Karp Algorithm, and the pseudocode that could be used.

[9] and [10] provide us an insight to TSP for multiple vehicles, and the previous work that has been done on the subject. It also gives us valuable information on the mathematical computations used for the purpose.

The literature allowed us to understand how travelling-salesman helps to optimize the cost in various scenarios, including practical implementations like DNA sequencing, and ant colony behavior.

PROJECT OBJECTIVES

OBJECTIVE 1:

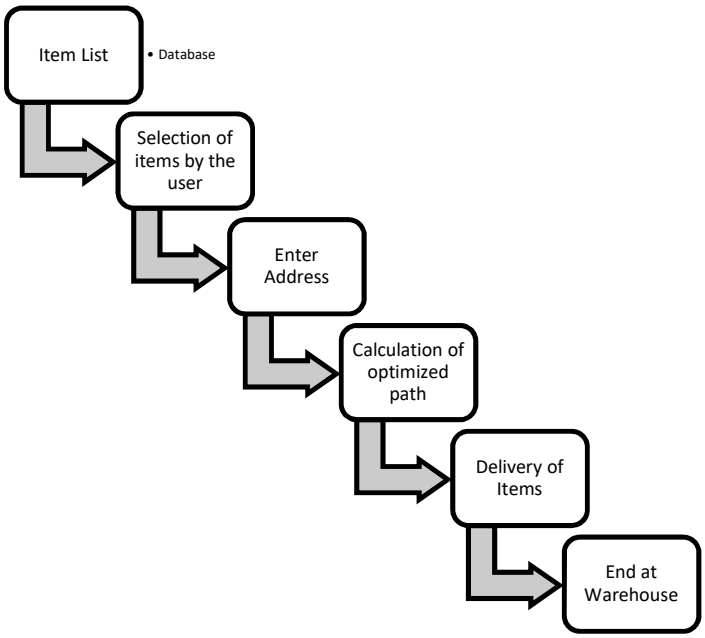
To have an understanding of algorithm used, its complexity analysis, and implementation for minimizing the cost of transportation.

OBJECTIVE 2:

To understand the working of delivery routing systems used by the logistics staff in an e-commerce organization, and to demonstrate the methodology used by this algorithm.

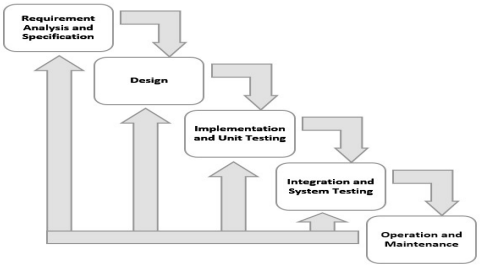
METHODOLOGY

The basic flow of software is given below:



This Project follows the Iterative Waterfall Software Process Model, wherein there are 5 stages, namely:

a. Requirement Analysis and Specifications- Capturing the requirements and expectations from the software
b. Design- Deciding upon the algorithms to be used for the software building
c. Implementation and Unit Testing- This phase deals with the actual coding part of the software
d. Integration and System Testing- Different test cases are tested in this phase. The software is tested for any faults or failures
e. Operation and Maintenance- Patching the issues, updating with new latest features are incorporated in this stage, along with checking in all possible environments



For the optimal planning, design, and implementation and achievement of the project objectives, this model is being used. It is a sequential model with each phase dealing with series of tasks and distinct objectives. By separating the set of concerns, the large and complex task of building the software is broken into smaller tasks which make it easier to handle. The output of each phase becomes the input for the next, this helps in keeping track of the work done at each stage. [6]

When errors are detected at some later phase, these feedback paths allow correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases.

IMPLEMENTATION

- The project has been divided into 4 modules for implementation of TSP using Dynamic Programming, each corresponding to a part of the project

    - MODULE 1: Creation of Data File in C
    - MODULE 2: User Entry (Inputs by the user)
    - MODULE 3: Edit Product Record
    - MODULE 4: Computation of Optimal Route (Minimum Cost)

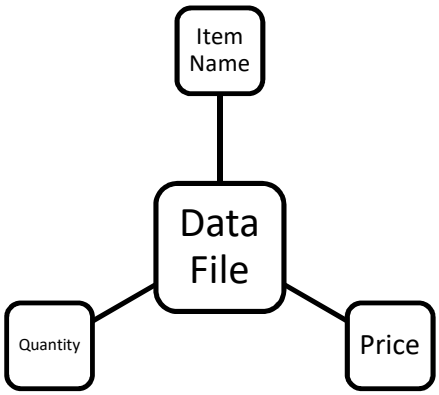
- The project is implemented in two ways for Comparative Study of time and space complexity

    - Dynamic Programming: Bellman-Held-Karp Algorithm
    - Brute Force: Johnson and Trotter Algorithm

- Need of Travelling-Salesman Problem is identified for the scenario using multiple references, including DNA sequencing, and Ant Colony Behavior.

MODULES

- MODULE 1: Creation of data file in C



  - A Data File simulates the items available for purchase using file handling
  - The file consists of the items, their quantity, and price
  - The file could be manipulated as per the requirements by the programmer

- MODULE 2: User Entry (Inputs by the user)

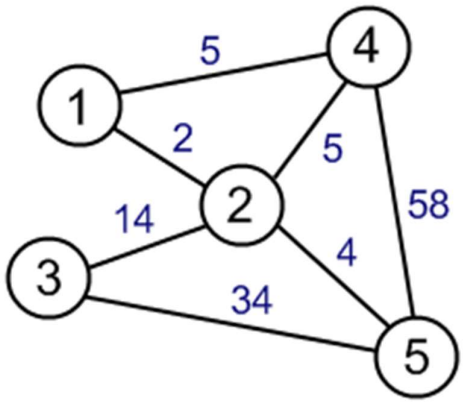
  - The module aims at a simulation of items being ordered by a user on an E-Commerce platform
  - Items being ordered by the user shall be fetched from the data file created in Module 1
  - The number of users could be n, where n is any natural number
  - Each order shall be stored in a file for use in other corresponding modules along with the address

- MODULE 3: Edit Product Record

  - The module focuses on editing the database of available items
  - The functionalities accessible to the user are:

    - Append
    - Read
    - Edit
    - Delete

- MODULE 4: Computation of optimal route (minimum cost)



- Inputs from node table will create the minimum route
- The path shall be displayed for the use of delivery staff which could further be shared to the delivery personnel

PSEUDOCODE

The pseudocode for Bellman-Held-Karp is as follows:

```
function algorithm TSP (G, n)
  for k := 2 to n do
    C({k}, k) := d1,k
  end for

for s := 2 to n-1 do
    for all S ⊆ {2, . . . , n}, |S| = s do
      for all k ∈ S do
        C(S, k) := minm≠k, m∈S [C(S\{k}, m) + dm,k]
      end for
    end for
  end for

 opt := mink≠1 [C({2, 3, . . . , n}, k) + dk,1]
  return (opt)
end
```

ALGORITHM USED

The Bellman-Held-Karp algorithm belongs to Dynamic Programming in TSP, and is the most efficient algorithm to be implemented for solving it:

- Take one starting vertex x arbitrarily
- For the set of vertices S belonging to V\{S}, vertex v is in S, let:
    - B(S,V) = minimum length of path, that:
    - Starts in S
    - Visits S and all vertices in S (and no other vertices)
    - Ends in V

1. $B(\{v\},v) = d(s,v)$
2. For j = 2 to |V| - 1:
    1. For all sets S, |S| = j for all v belonging to S:
    2. $B(s,v) = \min_{w\text{-}S\setminus(v)}\{B(S\setminus\{V\},w\}) + d(w,v)\}$
3. Return $\min\{B(V\setminus\{s\},v) + d(v,s) \mid v \text{ belongs to } V\}$

The total running time is therefore $O(n^2*2^n)$.

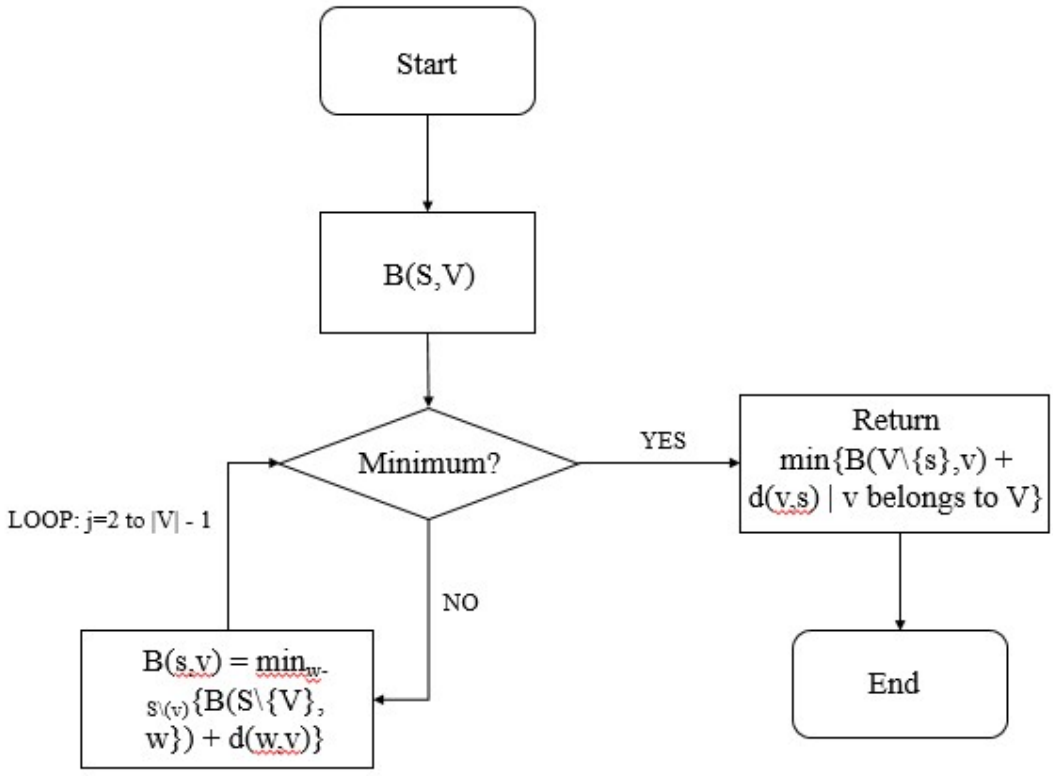
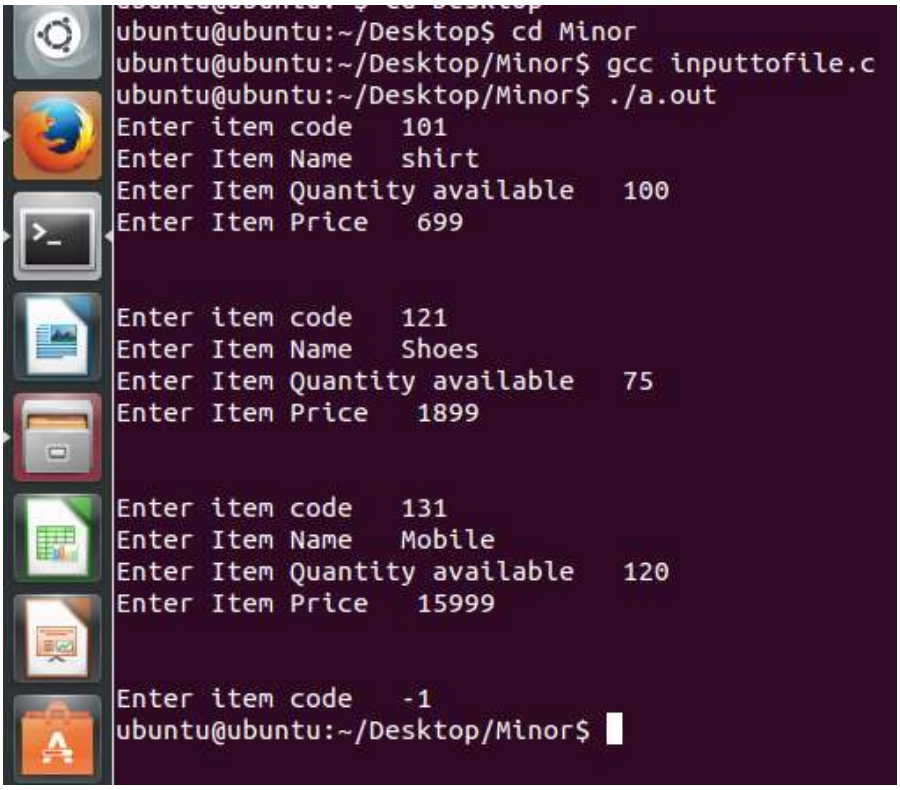
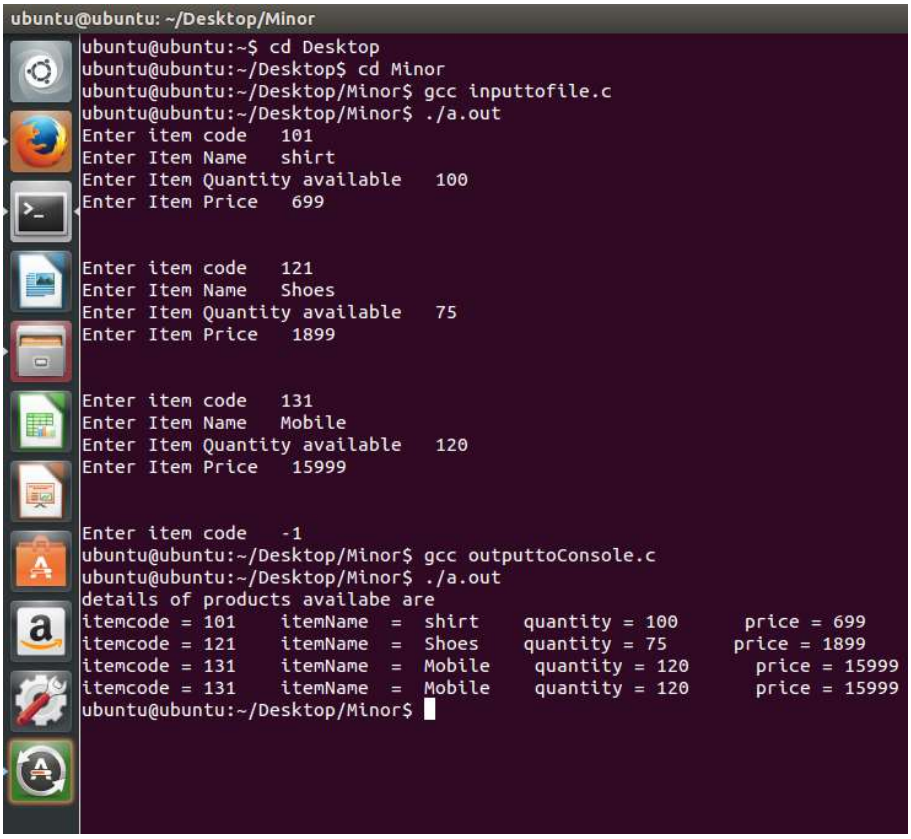
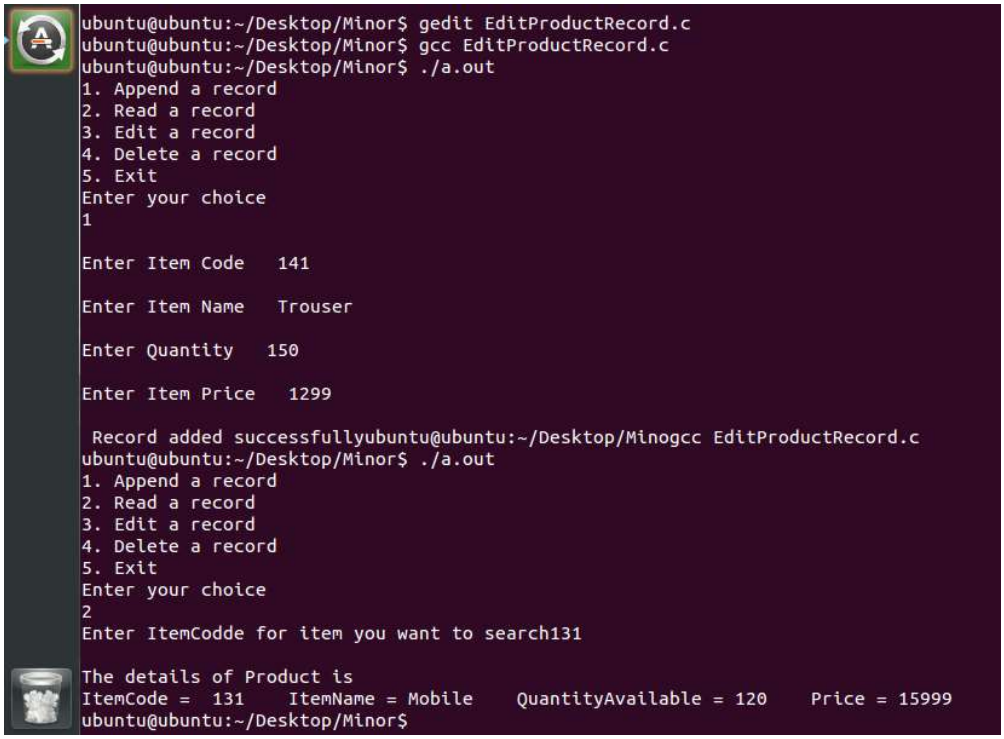


Figure 1Control Flow

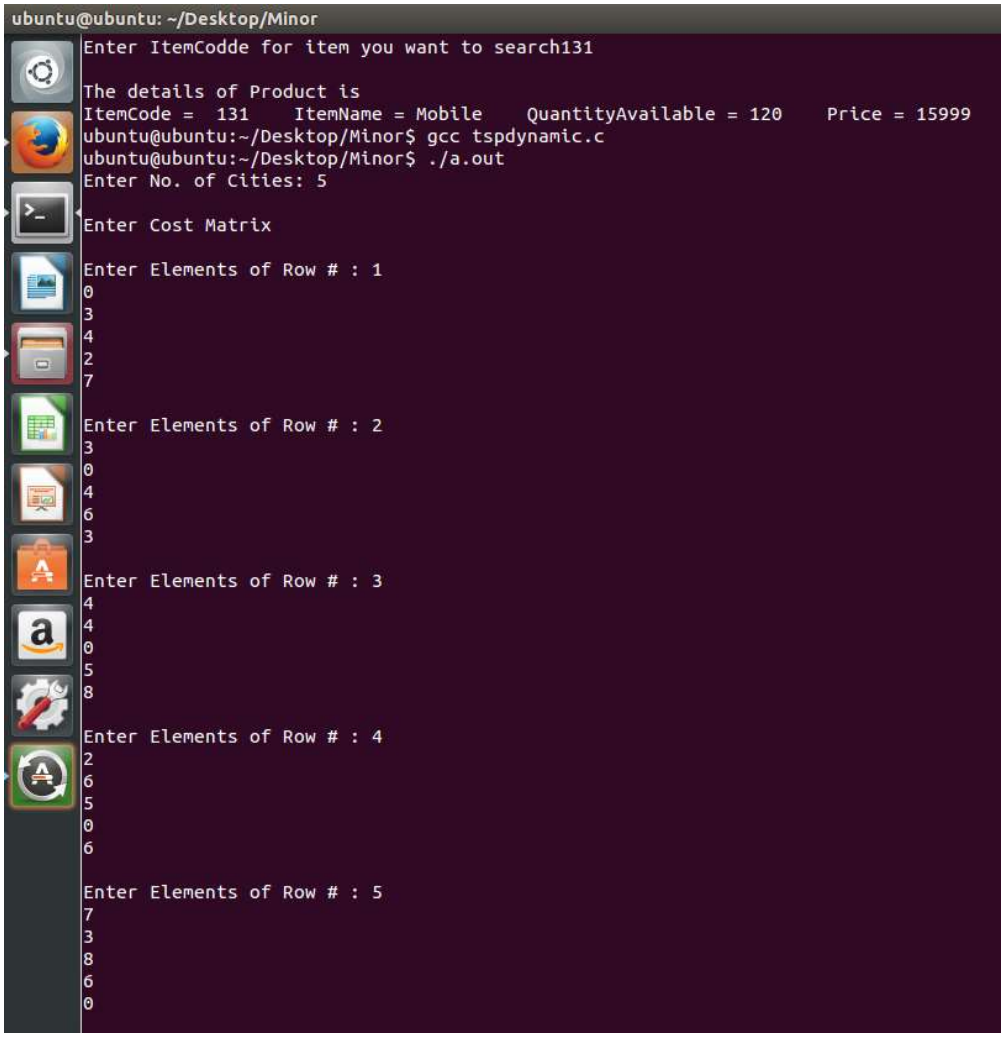
OUTPUT SCREENS



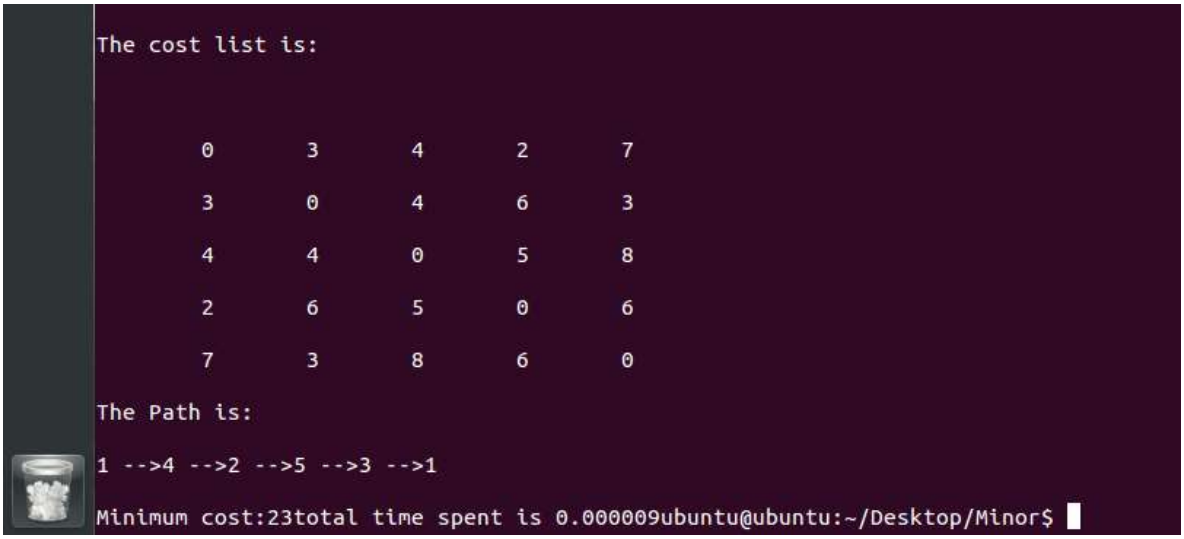




```
ubuntu@ubuntu:~/Desktop/Minor$ gedit EditProductRecord.c
ubuntu@ubuntu:~/Desktop/Minor$ gcc EditProductRecord.c
ubuntu@ubuntu:~/Desktop/Minor$ ./a.out
1. Append a record
2. Read a record
3. Edit a record
4. Delete a record
5. Exit
Enter your choice
1

Enter Item Code   141

Enter Item Name   Trouser

Enter Quantity   150

Enter Item Price   1299

 Record added successfullyubuntu@ubuntu:~/Desktop/Minogcc EditProductRecord.c
ubuntu@ubuntu:~/Desktop/Minor$ ./a.out
1. Append a record
2. Read a record
3. Edit a record
4. Delete a record
5. Exit
Enter your choice
2
Enter ItemCodde for item you want to search131

The details of Product is
ItemCode =  131    ItemName = Mobile    QuantityAvailable = 120    Price = 15999
ubuntu@ubuntu:~/Desktop/Minor$
```



```
ubuntu@ubuntu: ~/Desktop/Minor
Enter ItemCodde for item you want to search131

The details of Product is
ItemCode =  131    ItemName = Mobile    QuantityAvailable = 120    Price = 15999
ubuntu@ubuntu:~/Desktop/Minor$ gcc tspdynamic.c
ubuntu@ubuntu:~/Desktop/Minor$ ./a.out
Enter No. of Cities: 5

Enter Cost Matrix

Enter Elements of Row # : 1
0
3
4
2
7

Enter Elements of Row # : 2
3
0
4
6
3

Enter Elements of Row # : 3
4
4
0
5
8

Enter Elements of Row # : 4
2
6
5
0
6

Enter Elements of Row # : 5
7
3
8
6
0
```



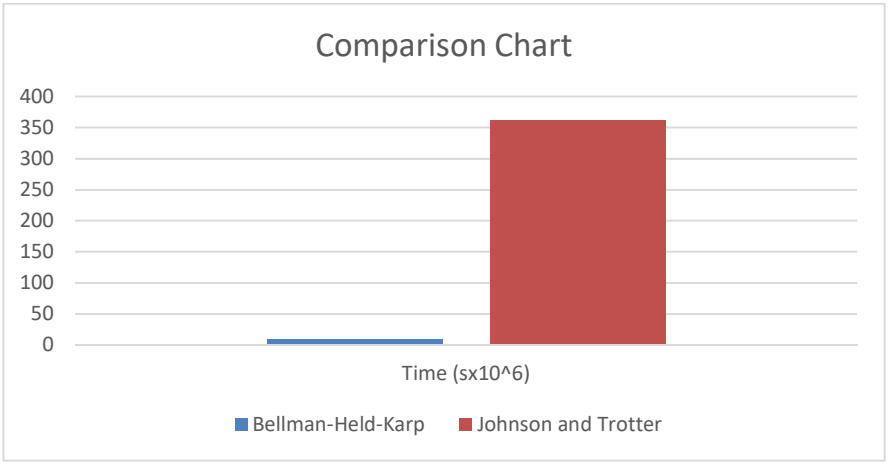
RESULT ANALYSIS

The key points of analysis of Bellman-Held-Karp Algorithm are as follows:

- It is the most efficient algorithm to solve the TSP dynamically
- The time complexity of this algorithm is O (n^2*2^n)
- The space complexity of this algorithm is O ($n*2^n$)

Comparative study highlighted the following key point:

Parameters\Algorithm	Bellman-Held-Karp	Johnson and Trotter
Time	0.000009 s	0.000362 s



OTHER ALGORITHMS AND THEIR COMPLEXITY ANALYSIS

Some algorithms are:

1. BRUTE FORCE ALGORITHM: The most direct solution would be to try all permutations (ordered combinations) and see which one is cheapest (using brute force algorithm). The running time for this approach lies within a polynomial factor of O((n-1)!).

2. NEAREST NEIGHBOR (NN) ALGORITHM: The algorithm (a greedy algorithm) lets the salesman choose the nearest unvisited city as his next move. This algorithm quickly yields an effectively short route. For N cities randomly distributed on a plane, the algorithm on average yields a path 25% longer than the shortest possible path.

3. MATCH TWICE AND STITCH (MTS) ALGORITHM: It performs two sequential matchings, where the second matching is executed after deleting all the edges of the first matching, to yield a set of cycles. The cycles are then stitched to produce the final tour.

4. CHRISTOFIDES' ALGORITHM: The Christofides' algorithm follows a similar outline but combines the minimum spanning tree with a solution of another problem, minimum-weight perfect matching. This gives a TSP tour which is at most 1.5 times the optimal.

CONCLUSION AND FUTURE SCOPE

The project could be updated for the additions of Address Indexing and Latitude and Longitude

-based addressing system for the users using web-based solutions. The problem could also be implemented for multiple vehicle routing, along with its allocation and implementation. The project has successfully implemented Traveling Salesman Problem using Bellman-Held-Karp Algorithm.

SYSTEM REQUIREMENTS

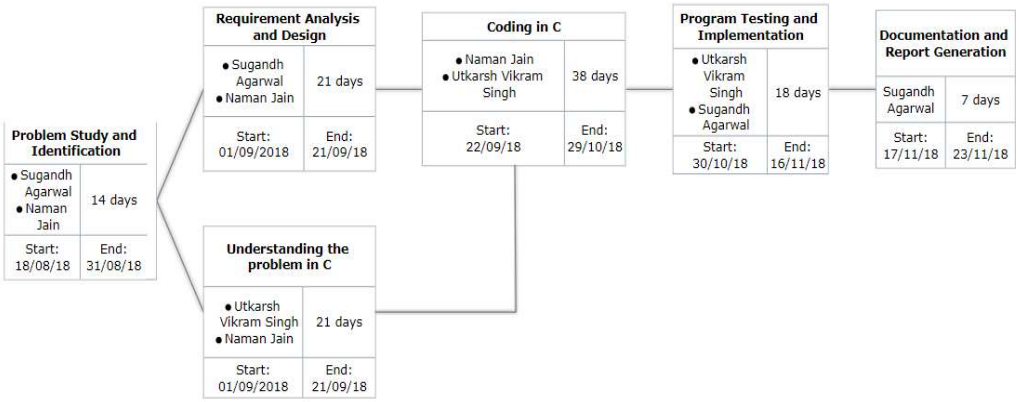
SOFTWARE REQUIREMENTS

Operating System	:	MS-DOS 2.0 or higher
Programming Language	:	C
Compiler	:	GCC

HARDWARE REQUIREMENTS

Processor	:	Pentium IV or higher
Disk Drive	:	Floppy or Hard Disk Drive
RAM	:	512 MB or higher

SCHEDULE (PERT CHART)



REFERENCES AND BIBLIOGRAPHY

[1] T.H. Cormen, C.E. Leiserson, R.R. Rivest, C. Stein. *Introduction to Algorithms: Third Edition.* The MIT Press, Cambridge, Massachusetts, London, England

[2] Cook, William J. *In Pursuit of the Traveling Salesman: Mathematics at the Limit of Computation*. Princeton, NJ: Princeton UP, 2012 [Print]

[3] C. Brucato. "The Travelling Salesman Problem," 2013, pp. 10-14

[4] C. Woodford. "E-Commerce." Internet: https://www.explainthatstuff.com/ecommerce.html, Sept. 14, 2016 [Sept. 06, 2018]

[5] Y. Yu, X. Wang, R.Y. Zhong, G.Q. Huang. "E-commerce Logistics in Supply Chain Management: Practice Perspective," 2016, pp. 179-185

[6] K.K. Aggarwal, Y. Singh. *Software Engineering: Third Edition.* New Age International Publishers, Dariya Ganj, New Delhi, Delhi

[7] S.K. Pal. "Software Engineering | Iterative Waterfall Model." Internet: https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model, Aug. 26, 2016 [Sept. 06, 2018]

[8] Wikipedia. "Held-Karp Algorithm" Internet: https://en.wikipedia.org/wiki/Held%E2%80%93Karp_algorithm#Dynamic_programming_approach, Apr. 1, 2018 [Oct. 28, 2018]

[9] Q. Wang. "Improving TSP Problem Allowing Multiple Vehicle Distribution," 2016

[10] M. Assaf, M. Ndiaye. "Multi Travelling Salesman Problem Foundation," 2017, pp. 292-295

APPENDIX 1: CODE

```
/*This Program will calculate minimum weight Hamiltonian cycle using Dynamic Programming approach
when matrix is symmetric.
This is a code snippet.
*/

#include<stdio.h>
#include<time.h>

int node_table[10][10];
int traversed[10];
int n,cost=0;

void get_table(){
int i,j;
printf("Enter No. of Cities: ");
scanf("%d",&n);
printf("\nEnter the distance(cost) between nodes\n");
for(i=0;i < n;i++){
printf("\nEnter Elements of Row # : %d\n",i+1);
for( j=0;j < n;j++){
        scanf("%d",&node_table[i][j]);
        traversed[i]=0;
}

printf("\n\nThe cost list is:\n\n");
for( i=0;i < n;i++){
        printf("\n\n");
        for(j=0;j < n;j++){
                printf("\t%d",node_table[i][j]);
        }
}

int mindist(int c){
        int i,nextcity=1000;
        int min=1000,costmin;
        for(i=0; i<n; i++){
                if((node_table[c][i]!=0)&&(traversed[i]==0)){
                        if(node_table[i][c] + node_table[c][i] < min){
                                min = node_table[i][0]+node_table[c][i];
                                costmin = node_table[c][i];
                                nextcity=i;
                        }
                }
        if(min!=1000)
                cost+=costmin;
        return nextcity;
}

float mincost(int city){
        double time_spent = 0.0;
        clock_t begin = clock();
        int i,ncity;
        traversed[city]=1;
        printf("%d -->",city+1);
        ncity=mindist(city);
        if(ncity==1000){
```

```
                ncity=0;
                printf("%d",ncity+1);
                cost+=node_table[city][ncity];
                return;
            }

             mincost(ncity);
        clock_t end = clock();
        time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
        return time_spent;
}

void display_result(){
        printf("\n\nMinimum cost:");
        printf("%d",cost);
}

int main(){
        float time_spent;
        get_table();
        printf("\n\nThe Path is:\n\n");
        time_spent=mincost(0);
        display_result();
        printf("total time spent is %f", time_spent);
return 0;
}
```