A PROJECT REPORT

ON

“Different exact and approximation algorithms for Travelling-Sales-Person ”

SUBMITTED BY

Mr. Jagtap Prashant

Mr. Bhoite Shubham

Type your text

Mr. Mane Shubham

Mr. Machale Sham

UNDER THE GUIDENCE OF

Prof. Sayyed J.I

DEPARTMENT OF COMPUTER ENGINEERINGHSBPVT’s FACULTY OF ENGINEERING YEAROF SUBMISSION: 2023-2024 ENGINEERING,KASHTI.

CERTIFICATE

This is certified that **Mr. Machale Sham Ashok Roll No. 20** of VII Semester of Bachelor Of Computer Engineering of Institute HSBPVT'sFaculty of Engineering, Kashti (Code:5303) has completed the mini project satisfactorily in course **Blockchain Technology** for theacademic year 2023-2024 as prescribed in the curriculum.

Place**:-Kashti.**

Seat No.:-

Date **:-**

PRN No**:-72155665F**

Subject Teacher **H.O.D.** **Principal**

Contents

Abstract

This report provides an overview of exact and approximation algorithms for the Traveling Salesperson Problem (TSP). Exact algorithms explored include Brute Force. Dynamic Programming, and Branch and Bound, offering guaranteed optimal solutions. For approximations, the Nearest Neighbour Algorithm and Minimum Spanning Tree Algorithms are examined, providing faster solutions with trade-offs in optimality. The report aids in understanding algorithmic choices based on problem size and computational needs, offering insights for practical TSP problem-solving.

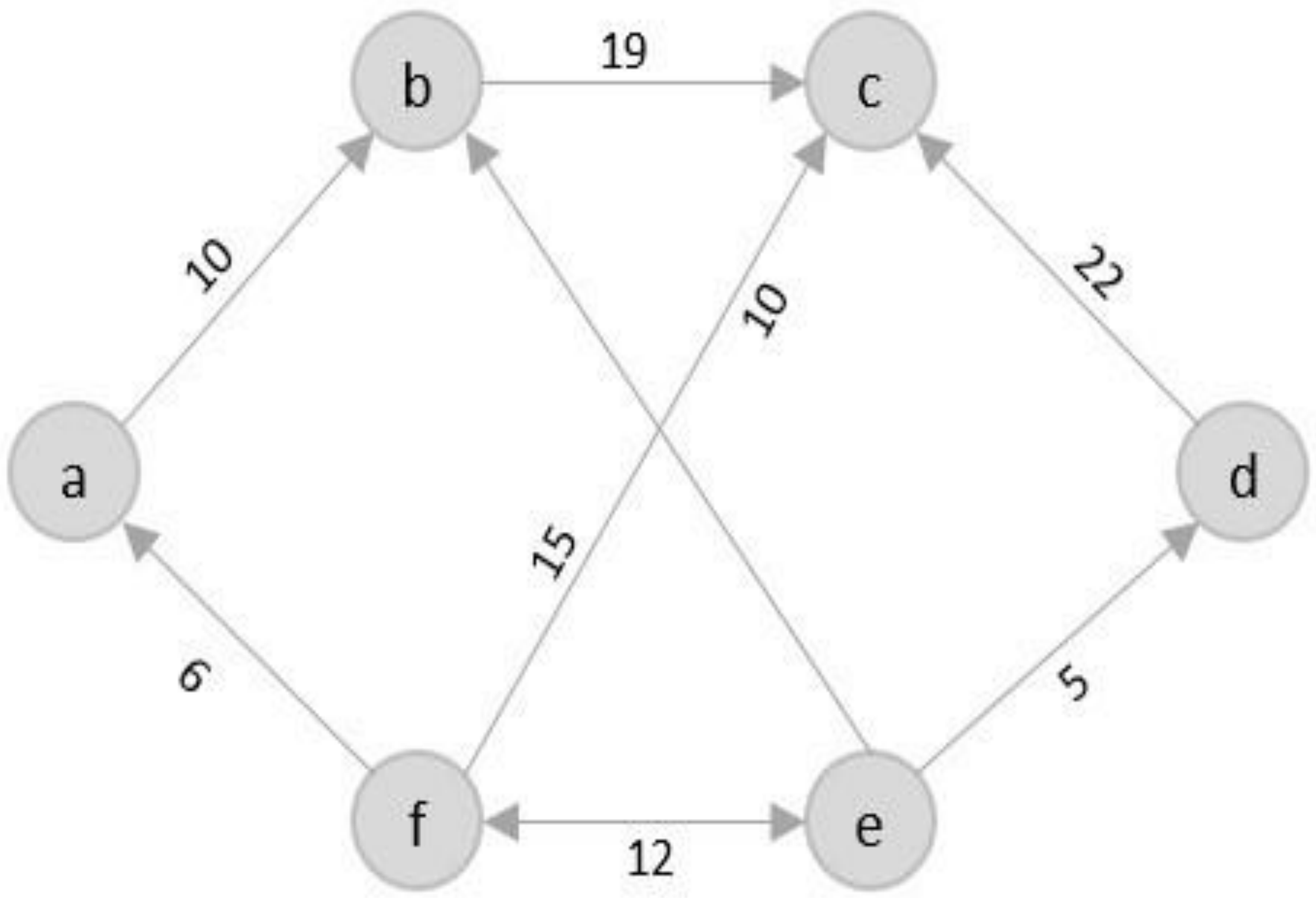
Problem Statement

The Traveling Salesperson Problem (TSP) is a fundamental optimization challenge, tasked with determining the most efficient route for a salesperson to visit a predetermined set of cities precisely once, concluding the journey by returning to the initial point of departure. This problem is formally represented as a complete graph, where cities are denoted as nodes, and edges between them represent the distances to be traversed. The core objective of TSP centres around minimizing the totaldistance travelled along a Hamiltonian cycle a cyclic path that visits eachcity exactly once

Introduction

The travelling salesman problem is a graph computational problemwhere the salesman needs to visit all cities (represented using nodes in a graph) in a list just once and the distances (represented using edges in the graph) between all these cities are known. The solution that is neededto be found for this problem is the shortest possible route in which the salesman visits all the cities and returns to the origin city.

If you look at the graph below, considering that the salesman starts from the vertex ‘a’, they need to travel through all the remaining vertices b, c, d,e, f and get back to ‘a’ while making sure that the cost taken is minimum.



There are various approaches to find the solution to the travelling salesman problem: naïve approach, greedy approach, dynamic programming approach, etc. In this report we will be learning aboutsolving travelling salesman problem using greedy approach.

Motivation

The Traveling Salesperson Problem (TSP) is a pervasive challenge with direct applications in logistics, network design, and beyond. Addressing TSP efficiently is crucial for cost reduction and resource optimization. This report is motivated by the practical relevance of TSP, aiming to explore algorithms that not only contribute to theoretical advancements but also offer tangible solutions for real-world optimizationproblems.

Objectives

- Explore principles of Brute Force, Dynamic Programming. Branch and Bound, Nearest Neighbour, and Minimum Spanning Treealgorithms for TSP.
- Evaluate computational efficiency, optimality, and scalability acrossvarious TSP instances.
- Highlight algorithmic strengths, weaknesses, and trade-offs for practical insights.
- Provide actionable recommendations for algorithm selection in real-world TSP problem-solving.

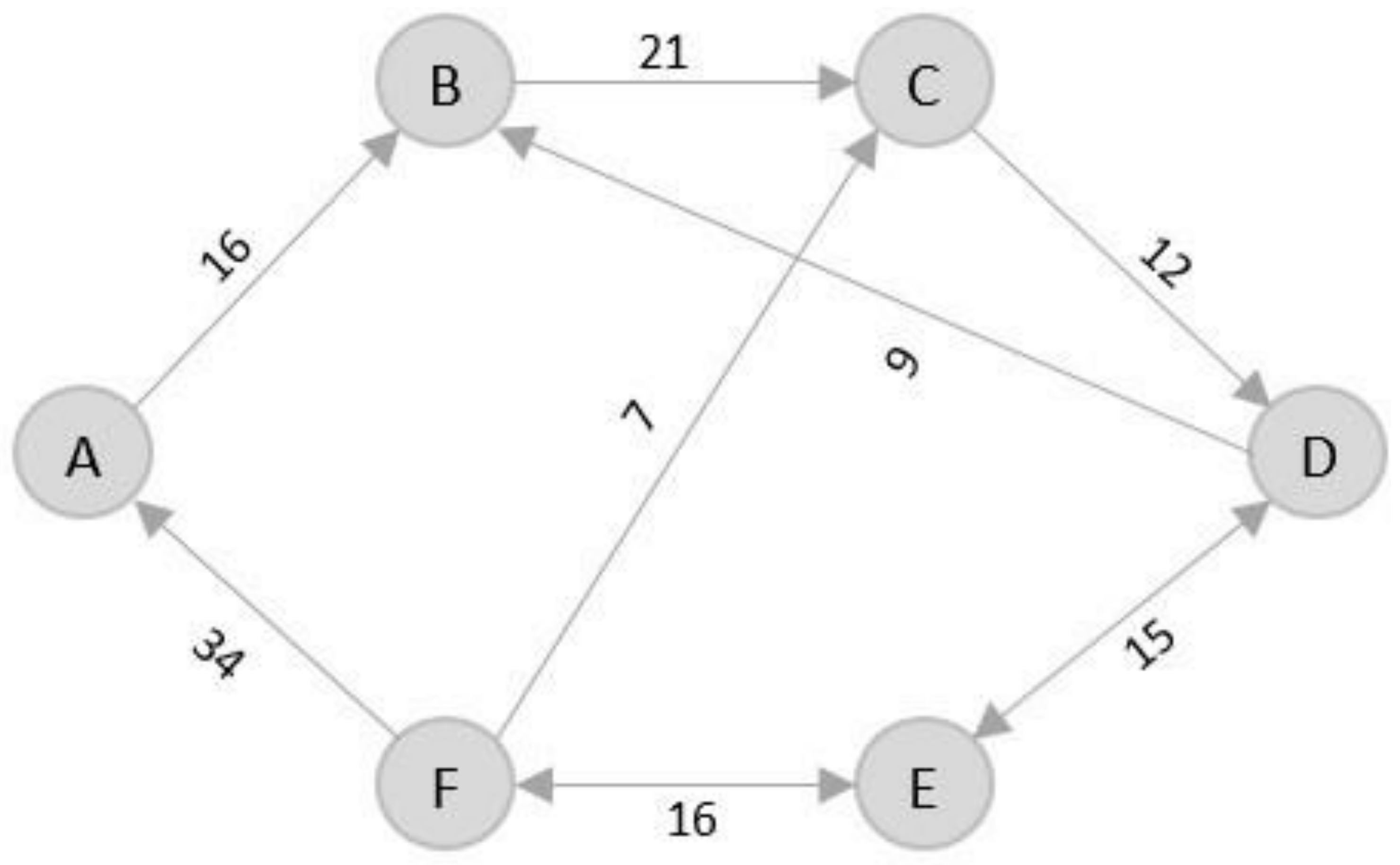
Algorithm

As the definition for greedy approach states, we need to find the best optimal solution locally to figure out the global optimal solution. Theinputs taken by the algorithm are the graph G {V, E}, where V is the set ofvertices and E is the set of edges. The shortest path of graph G startingfrom one vertex returning to the same vertex is obtained as the output.

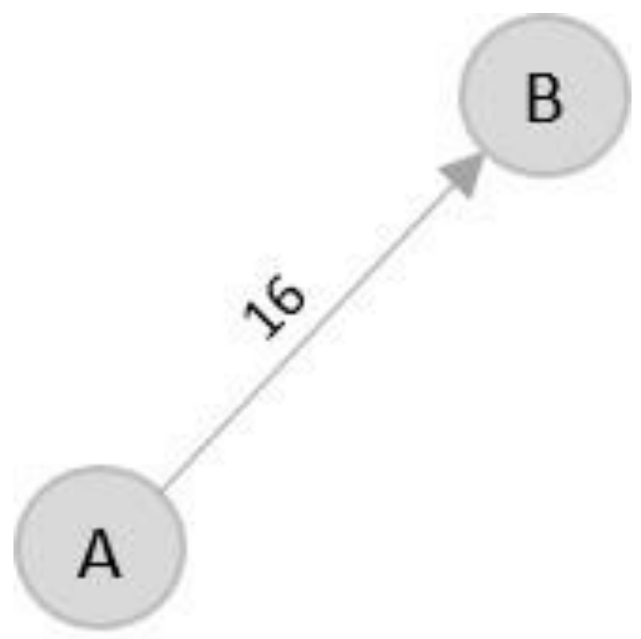
- Travelling salesman problem takes a graph G {V, E} as an input and declare another graph as the output (say G') which will record the path the salesman is going to take from one node to another.
- The algorithm begins by sorting all the edges in the input graph Gfrom the least distance to the largest distance.
- The first edge selected is the edge with least distance, and one ofthe two vertices (say A and B) being the origin node (say A).
- Then among the adjacent edges of the node other than the origin node (B), find the least cost edge and add it onto the output graph.
- Continue the process with further nodes making sure there are nocycles in the output graph and the path reaches back to the originnode A.
- However, if the origin is mentioned in the given problem, then the solution must always start from that node only. Let us look at someexample problems to understand this better.

Example

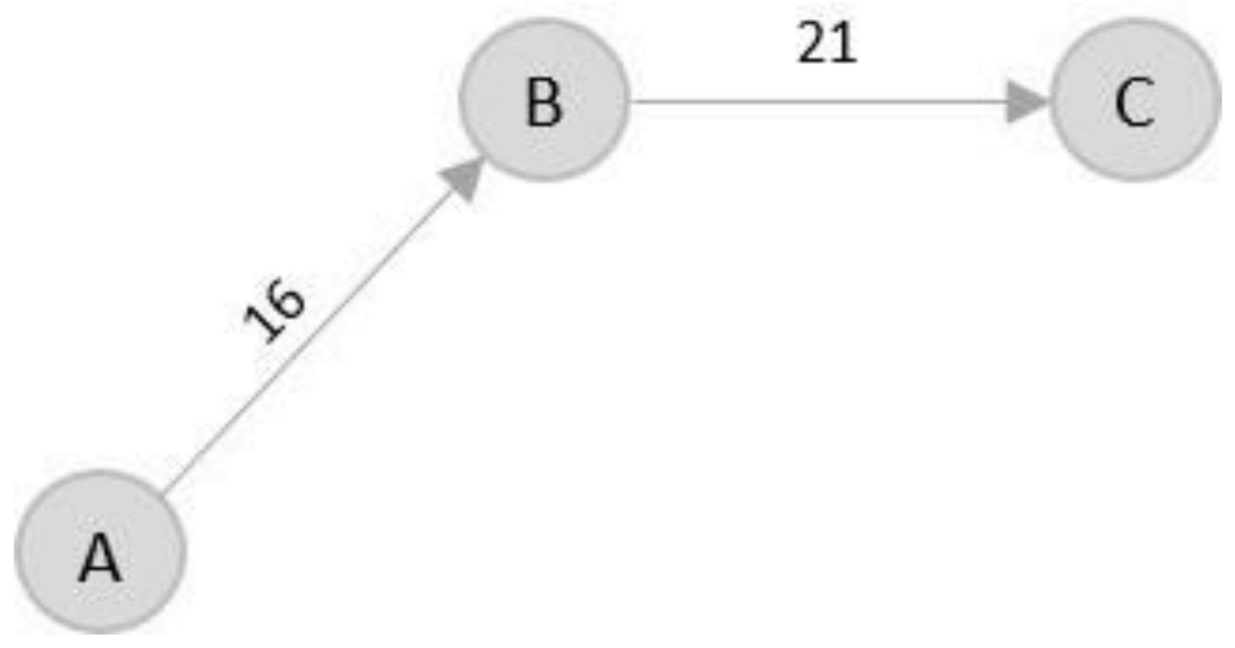
Consider the following graph with six cities and the distances between them –



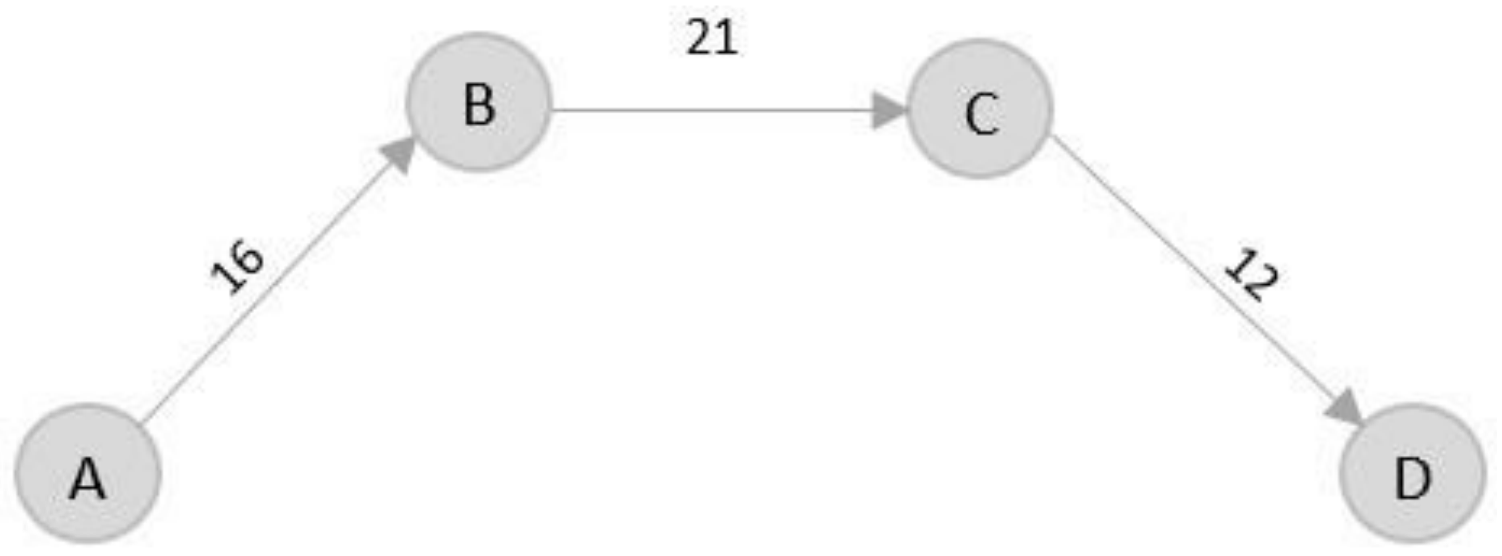
From the given graph, since the origin is already mentioned, the solutionmust always start from that node. Among the edges leading from A, A →B has the shortest distance.



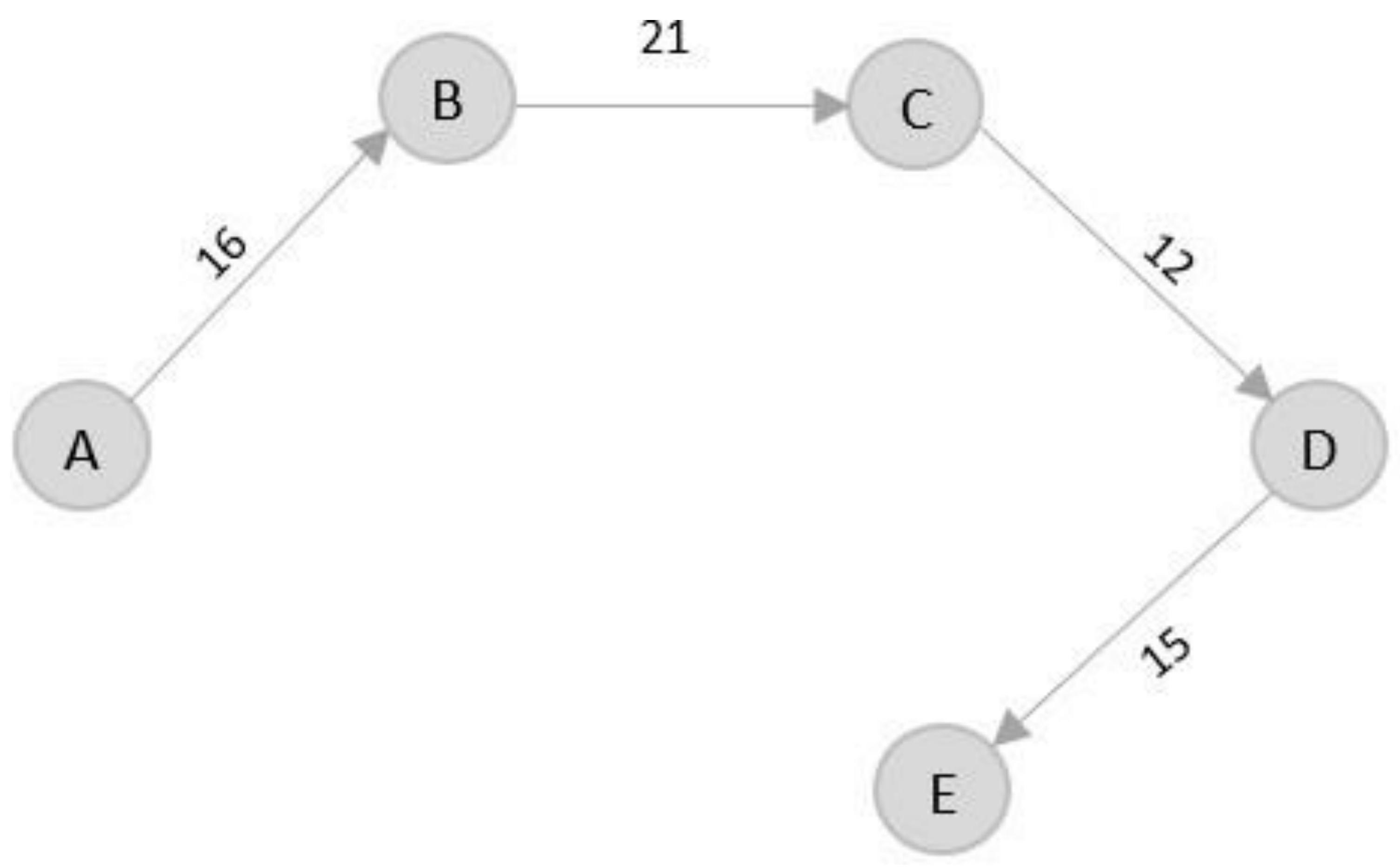
Then, B → C has the shortest and only edge between, therefore it isincluded in the output graph.



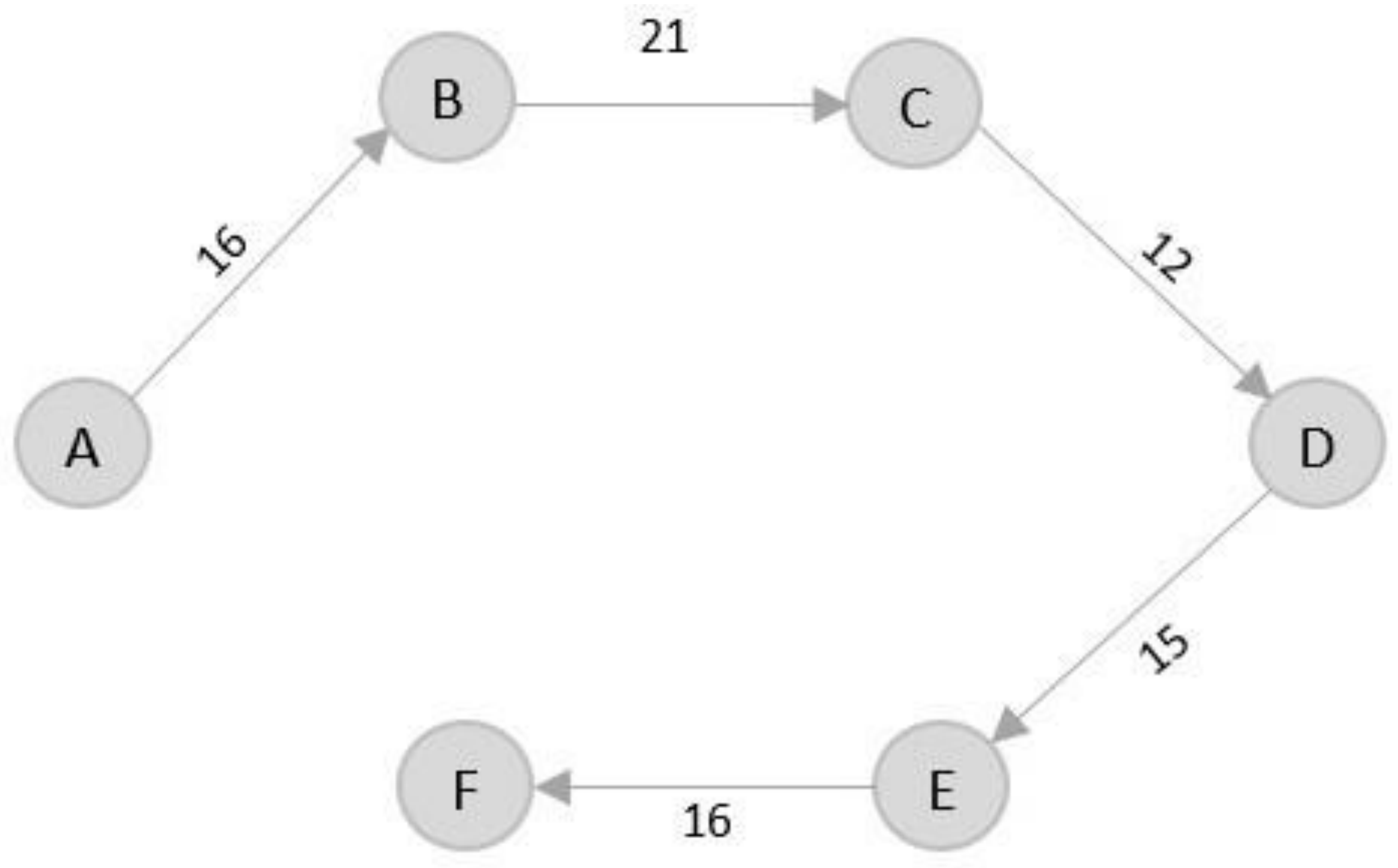
There’s only one edge between C → D, therefore it is added to the outputgraph.



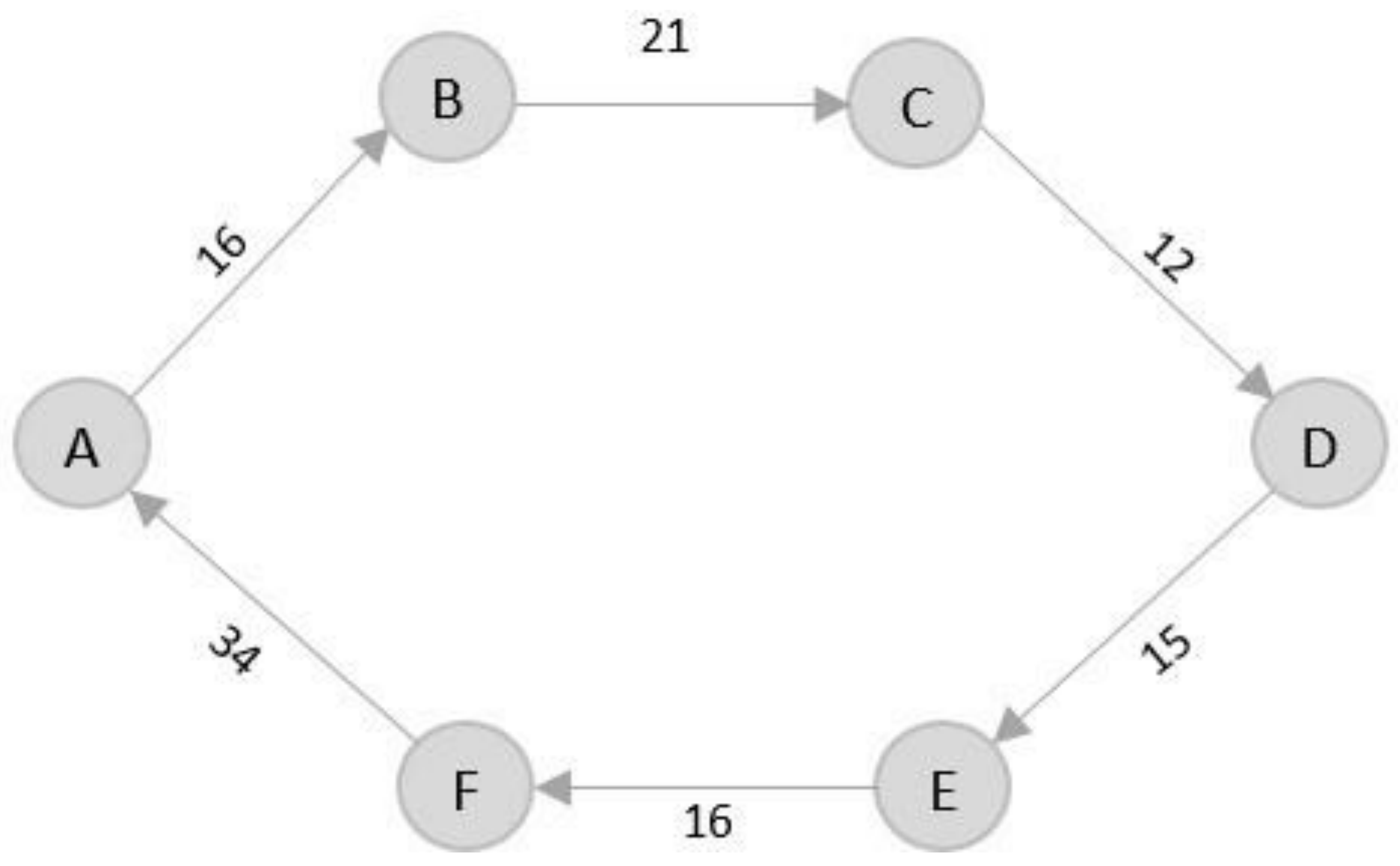
There’s two outward edges from D. Even though, D → B has lower distance than D → E, B is already visited once and it would form a cycle ifadded to the output graph. Therefore, D → E is added into the output graph.



There’s only one edge from e, that is E → F. Therefore, it is added into theoutput graph.



Again, even though F → C has lower distance than F → A, F → A is addedinto the output graph in order to avoid the cycle that would form and C is already visited once.



The shortest path that originates and ends at A is A → B → C → D → E → F → A

The cost of the path is: 16 + 21 + 12 + 15 + 16 + 34 = 114.

Implementation

Program:

```
import java.util.Arrays;
//Different exact and approximation algorithmfor Travelling-sales-person problempublic
class TravelingSalesperson {
    static final int V = 6;

    static int findMinKey(int key[], boolean mstSet[]) {int
        min = Integer.MAX_VALUE;
        int minIndex = -1;
        for (int v = 0; v < V; v++) {
            if (!mstSet[v] && key[v] < min) {
                min = key[v];
                minIndex = v;
            }
        }
        return minIndex;
    }

    static void primMST(int graph[][], int parent[]) {int
        key[] = new int[V];
        boolean mstSet[] = new boolean[V]; Arrays.fill(key,
        Integer.MAX_VALUE);Arrays.fill(mstSet, false);
        key[0] = 0;
        parent[0] = -1;
        for (int count = 0; count < V - 1; count++) {int u
            = findMinKey(key, mstSet); mstSet[u] = true;
            for (int v = 0; v < V; v++) {
                if (graph[u][v] != 0 && !mstSet[v] && graph[u][v] < key[v]) {
                    parent[v] = u;
                    key[v] = graph[u][v];
                }
            }
        }
    }
    // Function to print the preorder traversal of the Minimum Spanning Treestatic void
    printPreorderTraversal(int parent[]) {
        System.out.print("The preorder traversal of the tree is found to be  ");for (int i
        = 1; i < V; i++) {
            System.out.print(parent[i] + " -> ");
        }
        System.out.println();
    }
    // Main function for the Traveling Salesperson Approximation Algorithmstatic void
    tspApproximation(int graph[][]) {
        int parent[] = new int[V];
        int root = 0;

        primMST(graph, parent);
        // Print the preorder traversal of the Minimum Spanning Tree
        printPreorderTraversal(parent);
        System.out.print("Adding the root node at the end of the traced path ");for (int i =
        1; i < V; i++) {
            System.out.print(parent[i] + " -> ");
        }
        System.out.println(root + "  " + parent[0]);
        // Calculate and print the cost of the Hamiltonian pathint cost
        = 0;
        for (int i = 1; i < V; i++) {
            cost += graph[parent[i]][i];
        }
        // The cost of the path would be the sum of all the costs in the minimum spanning tree.
        System.out.println("Sum of all the costs in the minimum spanning tree: " + cost);
    }
    public static void main(String[] args) {
        // Example graph represented as an adjacency matrixint
        graph[][] = {
             {0, 3, 1, 6, 0, 0},
            {3, 0, 5, 0, 3, 0},
            {1, 5, 0, 5, 6, 4},
            {6, 0, 5, 0, 0, 2},
            {0, 3, 6, 0, 0, 6},
            {0, 0, 4, 2, 6, 0}
        };


       /*      */
        tspApproximation(graph);
    }
}
```

Output:

```
The preorder traversal of the tree is found to be 0 -> 0 -> 5 -> 1 -> 2 -> Adding the
root node at the end of the traced path 0 -> 0 -> 5 -> 1 -> 2 -> 0  -1Sum of all the costs
in the minimum spanning tree: 13
```

Result

The Beard wood–Halton–Hammersley theorem provides a practicalsolution to the travelling salesman problem. The authors derived an asymptotic formula to determine the length of the shortest route for a salesman who starts at a home or office and visits a fixed number of locations before returning to the start.

Conclusions

In conclusion, the Traveling Salesman Problem is a well-studied optimization problem with a wide range of real-world applications. The problem is defined as finding the shortest route that visits a set of cities exactly once and returns to the starting point.

References

- https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_travelling_salesman_problem.html
- https://cran.r-project.org/web/packages/TSP/vignettes/TSP.pdf
- https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/
- https://www.google.com/search?q=travelling+salesman+problem+introduction&oq=&gs_lcrp=EgZjaHJvbWUqCQgAECMYJxjqAjIJCAAQIxgnGOoCMgkIARAjGCcY6gIyCQgCECMYJxjqAjIJCAMQIxgnGOoCMgkIBBAjGCcY6gIyCQgFECMYJxjqAjIJCAYQIxgnGOoCMgkIBxAjGCcY6gLSAQkxNjkzajBqMTWoAgiwAgE&sourceid=chrome&ie=UTF-8
- https://en.wikipedia.org/wiki/Travelling_salesman_problem#:~:text=The%20Beardwood%E2%80%93Halton%E2%80%93Hammersley%20theorem,before%20returning%20to%20the%20start.
- https://www.google.com/search?q=travelling+salesman+problem&oq=&gs_lcrp=EgZjaHJvbWUqCQgCECMYJxjqAjIJCAAQIxgnGOoCMgkIARAjGCcY6gIyCQgCECMYJxjqAjIJCAMQIxgnGOoCMgkIBBAjGCcY6gIyCQgFECMYJxjqAjIJCAYQIxgnGOoCMgkIBxAjGCcY6gLSAQkyMTc2ajBqMTWoAgiwAgE&sourceid=chrome&ie=UTF-8