



জৱাহরলাল নেহরু মহাবিদ্যালয়
JAWAHARLAL NEHRU COLLEGE

A Project Report

On

Dijkstra's algorithm for shortest path: Optimizing campus routes "

Master of Science in Mathematics



Submitted By

Dimpee Kalita

G.U. Roll No: PS-231-019-0028

G.U. Registration No: 19012468

Session: 2024-25

Submitted to

Department of Mathematics

Jawaharlal Nehru College, Boko

Acknowledgement

*I would like to express my sincere gratitude to my project - guide, **Sir Jayanta Kalita**, Department of Mathematics, Jawaharlal Nehru College, Boko for his constant support, inspiration, cordial advice and invaluable guidance throughout the course of this project.*

*I find myself spellbound to acknowledge **Dr. Dipankar Sarma Sir**, Head of the Department (Mathematics) Jawaharlal Nehru College, Boko for providing the required facilities from the department and all kind of academic help needed during the course of my project.*

I would also like to extend my thankful note to all the faculty members of department of mathematics, Jawaharlal Nehru College, Boko and my fellow batchmates for their help and support. Last but not the least I am indebted to my family for their support, patience and understanding

Dimpee Kalita

Date: 23/12/2024

Dimpee Kalita

GU Roll No. PS-231-019-0028

GU Registration No. 19012468

DECLARATION

I hereby declare that this project report has been prepared by me during the year 2024-25 under the guidance of Sir Jayanta Kalita in the partial fulfilment of the syllabus for 3rd semester of M.Sc (Mathematics) degree.

I also declare that this project report represents largely my own understandings and words in this topic. This report has been prepared without resorting to plagiarism and adhered to all principles of academic honesty and integrity. No falsified or fabricated data or information has been presented.

Finally I declare that this project report is the outcome of my own efforts and that it has not been submitted elsewhere for the award of any degree.

Dimpee Kalita

23/12/2024

DIMPEE KALITA

(PS-231-019-0028)

GU Registration No. 19012468

CERTIFICATE

*This is to certify that the project report entitled " **Dijkstra's algorithm for shortest path: Optimizing campus routes** " is an original piece of work completed by Dimpee Kalita of Jawaharlal Nehru College, Boko in partial fulfilment of the requirements for the degree of Masters of Science in Mathematics (University of Gauhati) under my guidance and supervision. No part of this project work has been submitted for the award of any degree at any other institution.*

Dimpee Kalita, completed the project work , fulfilling all the criteria and necessary requirements.

Date : 23/12/2024


Signature

Mr. Jayanta Kalita (Project Supervisor)
Assistant Professor (Department of Mathematics)
Jawaharlal Nehru College, Boko

CONTENTS

TOPICS

PAGE NO

1. <u>Abstract</u>	01
2. <u>Definition</u>	02
3. <u>History</u>	03-04
4. <u>Working Principle</u>	05-06
5. <u>Directed and Undirected Graphs</u>	
I. <u>Undirected Graph</u>	07-09
II. <u>Directed Graph</u>	10-11
6. <u>Application of Dijkstra's Algorithm</u>	12-14
7. <u>Limitations</u>	15-17
8. <u>Optimization of Dijkstra's algorithm</u> <u>in our College campus:</u>	18-20
9. <u>Project Result</u>	21
10. <u>Conclusion</u>	22
11. <u>Bibliography</u>	23

Abstract

Dijkstra's Algorithm, introduced in 1956 by Edsger W. Dijkstra, is a crucial tool in graph theory for finding shortest paths in the directed and undirected graphs with non-negative weights. This algorithm employs a priority queue to find the optimal path by iterative calculations.

This algorithm is highly respected in network routing, in GPS navigation, and in transportation planning: optimizing processes of data transmissions and route mapping. The problems with this algorithm will arise in routing with negative edge weights and dense graph scenarios.

This project applies Dijkstra's Algorithm in determining the shortest route to a campus by modelling roads as edges and indicating intersections as nodes, trying to optimize the commute while at the same time addressing various algorithmic challenges.

HISTORY:

Dijkstra's algorithm was one of the most influential contributions to computer science ever made by Edsger W. Dijkstra, the Dutch computer scientist, who developed it in 1956. It is used to determine the shortest path in weighted graphs, and its contribution places it as a landmark in graph theory and network optimization. The algorithm, published for the first time in formal form in 1959 under the title "A Note on Two Problems in Connexion with Graphs", greatly changed the approach to finding shortest paths.

Dijkstra's inspiration for this algorithm came from a problem at a practical level. It aimed to find the shortest path between two cities in the Netherlands. At that time, computing was at an embryonic stage, and techniques that could efficiently solve this sort of problem were quite absent. Dijkstra created his own method by iteratively determining the shortest path considering the closest unvisited node. This approach, currently regarded as a greedy algorithm, satisfies optimality by maximizing efficiency at every step.

The algorithm works by giving each node in the graph a tentative distance, which it then sets to zero for the starting node and infinity for all others. It repeatedly picks the node with the smallest tentative distance that hasn't been processed yet, marks it as visited, and then updates the tentative distances of its neighbours. That way, when it processes each node, it does so in a minimum-distance order. This continues until it has visited all nodes.

Dijkstra's algorithm was immediately used in transportation and navigation systems. It was used to map routing to help users

find the shortest paths between locations. In time, it spread into computer networks, from which it became the base for routing protocols such as OSPF (Open Shortest Path First) and IS-IS (Intermediate System-Intermediate System), which transmit data packets across networks. Moreover, it has been applied in robotics for navigation and in gaming for AI pathfinding.

The algorithm has some limitations despite the wide success it has garnered. It cannot handle graphs that have negative edge weights as this may lead to erroneous results. Algorithms such as Bellman-Ford correct this limitation. Additionally, the algorithm's memory usage can become a bottleneck for very large graphs.

Dijkstra's algorithm is not just a tool for computing, it marks a paradigm shift in solving optimization problems. Its introduction has been instrumental in demonstrating the power of algorithmic design and has inspired a multitude of developments in graph theory and computer science. Dijkstra himself, as one of the pioneers in structured programming, was awarded the Turing Award in 1972 for his overall contributions, including this landmark algorithm.

Today, Dijkstra's algorithm is part of the foundation of computer science education and a fundamental tool in applications ranging from logistics to artificial intelligence. Its legacy is a testament to the enduring impact of well-crafted algorithms in solving complex problems efficiently.

WORKING PRINCIPLE:

The Working Principle of Dijkstra's Algorithm:

Basically, Dijkstra's algorithm is used when we need to find the shortest path for each node from a specific node in a given weighted graph. The generality, closeness of the node, selection of the node in determining the shortest path is increasing incremental. Elaborations on its working include the following:

Step 1: Initialization

1. Give the starting node (source) a tentative distance of 0, with every other node receiving tentatively infinite (∞).
2. Mark all the nodes as unvisited.
3. Put the starting node as the current node.

Step 2: Examine Neighbours

1. From the current node, look closely at all the unvisited neighbours.
2. Calculate the tentative distance to each neighbour using:
Tentative Distance = Distance to Current Node + Weight of Edge.
3. If the calculated tentative distance is less than the current distance the neighbour is holding in memory, overwrite the neighbour's distance value.

Step 3: Mark Node as Visited

1. Mark current nodes as visited once all neighbouring nodes have been completely processed.
(Currently marked are none capable of being revisited.)

Step 4: Your Next Node

1. Select the unvisited node that has the smallest tentative distance."
2. Repeat Steps 2-4 until:
 - All nodes have been visited, or
 - The shortest path to the target node has been established.

Step 5: Termination

- The algorithm will stop once all the nodes are visited or the shortest path with respect to the given destination is prepared.

This searches the shortest path comprehensively and arrives at the most optimal solution through incremental selection, thereby giving the best solutions.

Directed and Undirected Graphs

1. Undirected Graph :

An undirected graph is a mathematical structure which is used to model the pairwise relationships between objects. An undirected graph consists of a set of vertices (nodes) and a set of edges (connections), where each edge connects two vertices without specifying a direction. Mathematically speaking, in an undirected graph, the edge connecting two vertices A and B is bidirectional; hence, it's allowed to traverse it accordingly in any direction. These graphs are often represented using adjacency lists or adjacency matrices. Such graphs are quite useful in many applications including social networks, transportation networks, and computer science algorithms where the relationships are inherently mutual or symmetric.

Let us understand more with the help of an example. Consider an undirected graph with 5 nodes: A,B,C,D,E, and the following edge weights :

A $\leftrightarrow$ B : 4

A $\leftrightarrow$ C : 2

B $\leftrightarrow$ C : 1

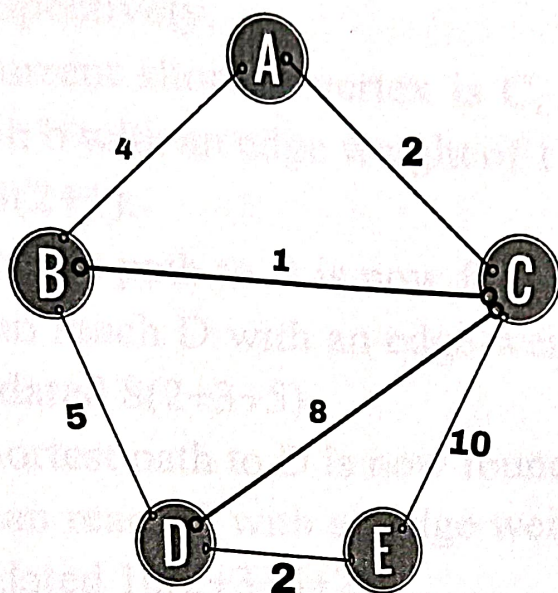
B $\leftrightarrow$ D : 5

C $\leftrightarrow$ D : 8

C $\leftrightarrow$ E : 10

D $\leftrightarrow$ E : 2

Here, we want to find the shortest path from node A to all the other nodes. We first draw a graphical representation of the above situation.



Further executions are performed in the table below.

Step	Current node	Distance updates to neighbour	Tentative Distance	Nodes
1	A (Start)	B=4, C=2	A=0, B=4, C=2, D= ∞ , E= ∞	Started at A, Nearest neighbours updated
2	C	B=2+1=3 (via C) D=2+8=10, E=2+10=12	A=0, B=3, C=2, D=10, E=12	(Shortest path to B)
3	B	D=2+1+5=8	A=0, B=3, C=2, D=8, E=12	Shorter path to C
4	D	E=2+1+5+2=10	A=0, B=3, C=2, D=8, E=10	Shorter path to E
5	E	No updates	A=0, B=3, C=2, D=8, E=10	All nodes visited

Explanation:

1. Start at A: The distance to A is 0, and the distances to its neighbours (B and C) are updated. The distances to b and C become 4 and 2, respectively.
2. Move to C: The current shortest vertex is C, with distance 2. From c we can reach b with an edge weight of 1. So, the distance to B is updated to $3(2+1)$.
3. Move to B: the shortest path to B is now found with a distance of 3. From b we can reach D with an edge weight of 5. So, the distance to D is updated $8(2+3+5)$.
4. Move to D: The shortest path to D is now found with a distance of 8. From D we can reach E with an edge weight of 2. So, the distance to E is updated $10(2+3+5+2)$.
5. Move to E: The algorithm finally reaches E, and the shortest path to E is 10.

Therefore the shortest path from node A to other nodes are :

A $\rightarrow$ B = 3 (via C)

A $\rightarrow$ C = 2

A $\rightarrow$ D = 8 (via B)

A $\rightarrow$ E = 10 (via D)

2. Directed Graph:

A directed graph (diagram) is a graph where each edge has a direction, meaning that each edge is an ordered pair of vertices. The edge indicates a one-way relationship from one vertex to another.

Dijkstra's algorithm finds the shortest path from a source to all the other vertices in a weighted directed graph with non-negative edge weights.

Let us consider a directed graph with 4 nodes A, B, C, D and the following edge weights:

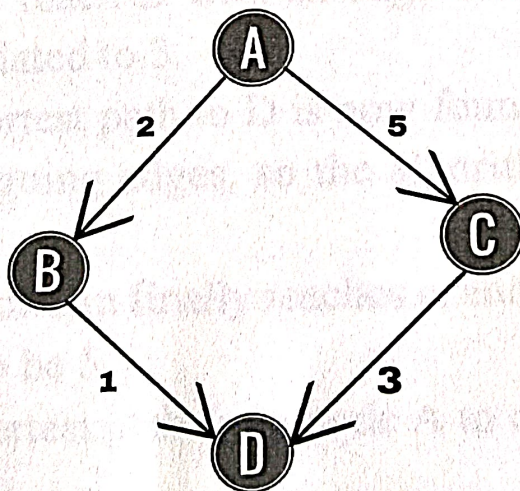
A $\rightarrow$ B : 2

A $\rightarrow$ C : 5

B $\rightarrow$ D : 1

C $\rightarrow$ D : 3

We represent the above situation graphically as-



We will use Dijkstra's algorithm to find the shortest path from vertex a to all other vertices.

Step	Current node	Distance updates to neighbour	Tentative Distance	Nodes
1	A (Start)	B=2, C=5	A=0, B=2, C=5, D= ∞	Started at A
2	B	D=2+1=3	A=0, B=2, C=5, D=3	Shortest path to D
3	D	No neighbours	A=0, B=2, C=5, D=3	A->B->D
4	C	D=3	A=0, B=2, C=5, D=3	C is visited, so more updates

Explanation:

1. Start at A: The distance to A is 0, and the distance to its neighbours(B and C) are updated. The distance to B and C become 2 and 5, respectively.
2. Move to B: The current shortest known vertex is B, with distance 2. From B, we can reach D with an edge weight of 1. So, the distance to D is updated to 3.
3. Move to D: the shortest path to D is now found with a distance of 3. D has no outgoing edges, so the algorithm moves to the next vertex.
4. Move to C: The algorithm finally reaches c, and the shortest path to C is confirmed to be 5.

Therefore the shortest path from node A to other nodes are :

A-> B = 2

A-> C = 5

A-> D = 3

Application of Dijkstra's algorithm:

Dijkstra's algorithm has a very widespread application in the field of technology, logistics and engineering. Some of the major applications are pointed below:

1. Shortest Path in Graphs:

This algorithm is used in graphs to find the shortest path from one node to another, primarily when the edges have non-negative weights. Its application in various problems and systems emphasizes an efficient traversable path for navigation.

2. Network Routing:

Dijkstra's algorithm is widely used in computer networks in routing protocols such as OSPF (Open Shortest Path First) to determine the best or most efficient path for a packet to travel from one node to another node.

3. GPS and Navigation Systems:

Dijkstra's algorithm in the GPS devices and mapping applications like Google Maps is to determine the shortest or fastest route between any two geographical points with respect to distance or travel time.

4. Transportation and Logistics:

It finds applications in transportation for route optimization, allowing for the planning of a logistics company's routes to be as time- and fuel-efficient as possible.

5. Telecommunication Networks:

The algorithm is also used in telecommunication systems to improve the routing of signals or manage traffic, thereby enabling data transmission with minimal delays.

6. Robot Navigation:

Robot navigation involves utilizing the algorithm uniquely so that the robot navigates with the least distance globally while traversing through already marked obstacles to a destination point.

7. Urban Planning and Infrastructure:

The algorithm assists the urban planners in planning an efficient road network system, water line network, and electric line network designs in order to optimize costs by minimizing the construction expenditures.

8. Gaming and Artificial Intelligence:

Dijkstra's algorithm is something often adopted in video games for pathfinding, the passage from point A to point B while avoiding obstacles by a character or NPC (non-player character) in a game.

9. Emergency Response Systems:

The algorithm is used by ambulance, fire, and police services to determine the shortest route to a scene of the incident so they can save time.

10. Electrical Circuit Design:

In the design of electronic circuits, Dijkstra's algorithm helps determine the shortest paths in wiring between components to reduce material consumption and complexity.

11. Traveling Salesman Problem: Heuristic Solutions:

Although Dijkstra's algorithm cannot directly resolve the Traveling Salesman Problem (TSP), it is applied as part of heuristic methods that achieve approximate optimal routes in simpler cases.

12. Social Network Analysis:

Application of Dijkstra's algorithm identifies those shortest degree of separation among users in social media, which could help understand connections or recommend friends.

13. Air Traffic Management:

Airlines and air traffic controllers will rely upon Dijkstra's algorithm to calculate the optimal flight route if factors such as distance travelled, weather conditions, and air traffic are all taken into consideration.

14. Energy Distribution Systems:

Dijkstra's algorithm determines a way of efficiently routing electricity from sources to consumers to minimize power losses in power grid systems.

15. Healthcare Systems:

The algorithm will also find application in hospital networks to route patients, staff, or medical supplies within and between the hospitals.

These applications provide an insight into how Dijkstra's algorithm is helpful in the entire optimization problem-solving process within technology, engineering, and logistics.

Limitations :

Dijkstra's algorithm is a well-known and well-accepted algorithm that finds shortest paths from the source vertex to other vertices in a weighted graph. It has some limitations that do impede its usefulness in some situations, although these limitations come with no question to maul its effectiveness in lots of other applications. These limitations maybe summarized as performance issues, assumptions about the graph, and appropriateness concerning specific problem classes.

1. Non-negative Weights:

Perhaps one of the major drawbacks of Dijkstra's algorithm is that it presumes all edge weights in the graph to be non-negative. This omission in itself disallows the algorithm from handling graphs such that certain edge weights stand negative. Hence, should the graph contain negative weight edges, the algorithm may fail to give correct answers as it works with the assumption that once a vertex is reached, its shortest path is forever, determined. Finding a shorter path before processing a vertex with negative weights violates this assumption. The Bellman-Ford algorithm is better set to manage situations with negative weights, so as to be able to handle these properly and also detect negative weight cycles.

2. Greedy Nature:

Dijkstra's algorithm is based on a greedy approach, meaning it makes the locally optimal choice at each point with the hope of finding a global optimum. Though it works in many cases, where Dijkstra's algorithm may not be appropriate is when

the graph does not properly lend itself to a greedy-style approach. For certain graph topologies, especially with particular edge weights being very complex or varied, those greedy decisions taken earlier on might lead to poor solutions. Again, this might not always be the case; however, in some edge instances, wrong greedy decisions may indeed cause the algorithm to miss the overall best path.

3. Memory Usage:

Because of keeping up with distance values for all vertices within the graph, Dijkstra's algorithm tends to be memory-intensive, especially when it comes to larger graphs. Moreover, the algorithm opts for priority queues or heaps to efficiently choose the next vertex to visit, thereby consuming more memory. For very large graphs, it could be a serious drawback, particularly when resources are constrained.

4. No Path Reconstruction:

Dijkstra's algorithm does not provide any mechanism to reconstruct the shortest paths from the source vertex; instead, it will just compute the shortest distances from the source. This other method, such as maintaining a predecessor array in memory, would require additional data structures, hence which could make the algorithm more complicated to write and also could use more memory.

5. Single Source Shortest Path Only:

Dijkstra's algorithm has existed to find the shortest path from one single vertex to all other vertices of the graph; it is not designed to efficiently deal with a multitude of source vertices.

The Floyd-Warshall algorithm might be more suitable for problems that need travel times between more than one sources.

Even if Dijkstra's algorithm is a well-known and commonly used algorithm, he has some noticeable weaknesses. They altogether include: handles only positive edge weights, is greedy by nature, not applicable to dense graphs due to performance issues, high space requirements in large graphs, does not allow for reconstruction of paths, and the limited use to single-source shortest path problems. While Dijkstra's algorithm is certainly a good fit in many situations, the understanding of its limitations is vital in selecting the best algorithm for a given problem context.

Optimization of Dijkstra's algorithm in our College campus:

Practically, let us use Dijkstra's algorithm to find the shortest path between some of the prime locations of our college from a fixed source path.

Let us select,

J N College canteen as our source vertex and we need to find the shortest distance from it to the other four locations, i.e. J N College, Boko, Science building, Mini stadium, Boko, Jawaharlal Nehru College(second gate), and NSS unit, JNC, Boko.

We shall now assign edge names to all the locations:

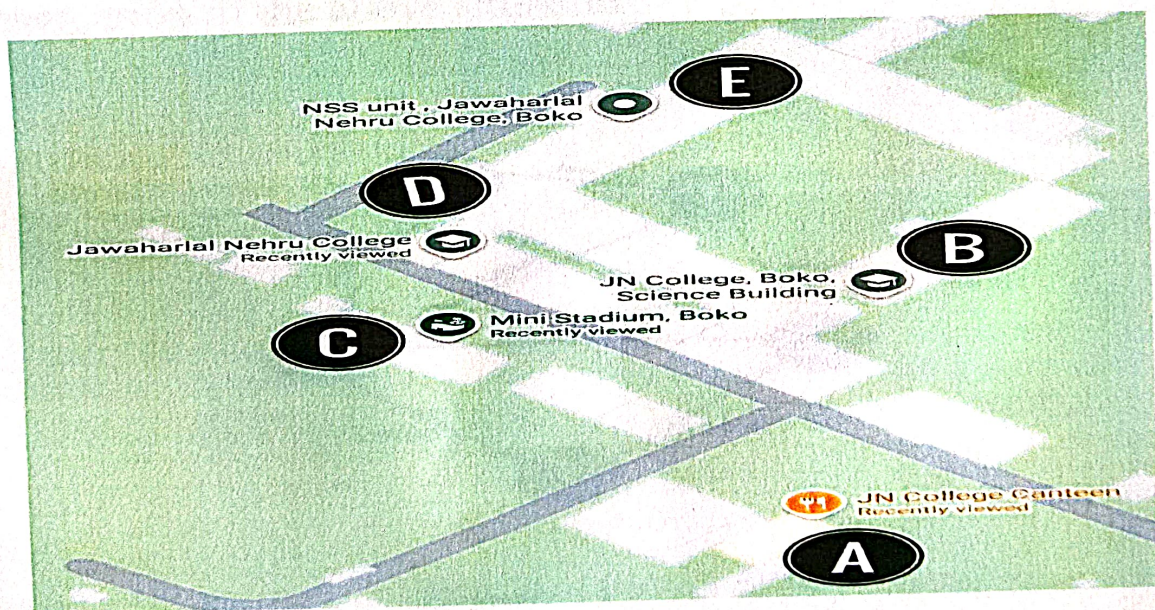
JN College Canteen : A

JN College, Science building : B

Mini stadium : C

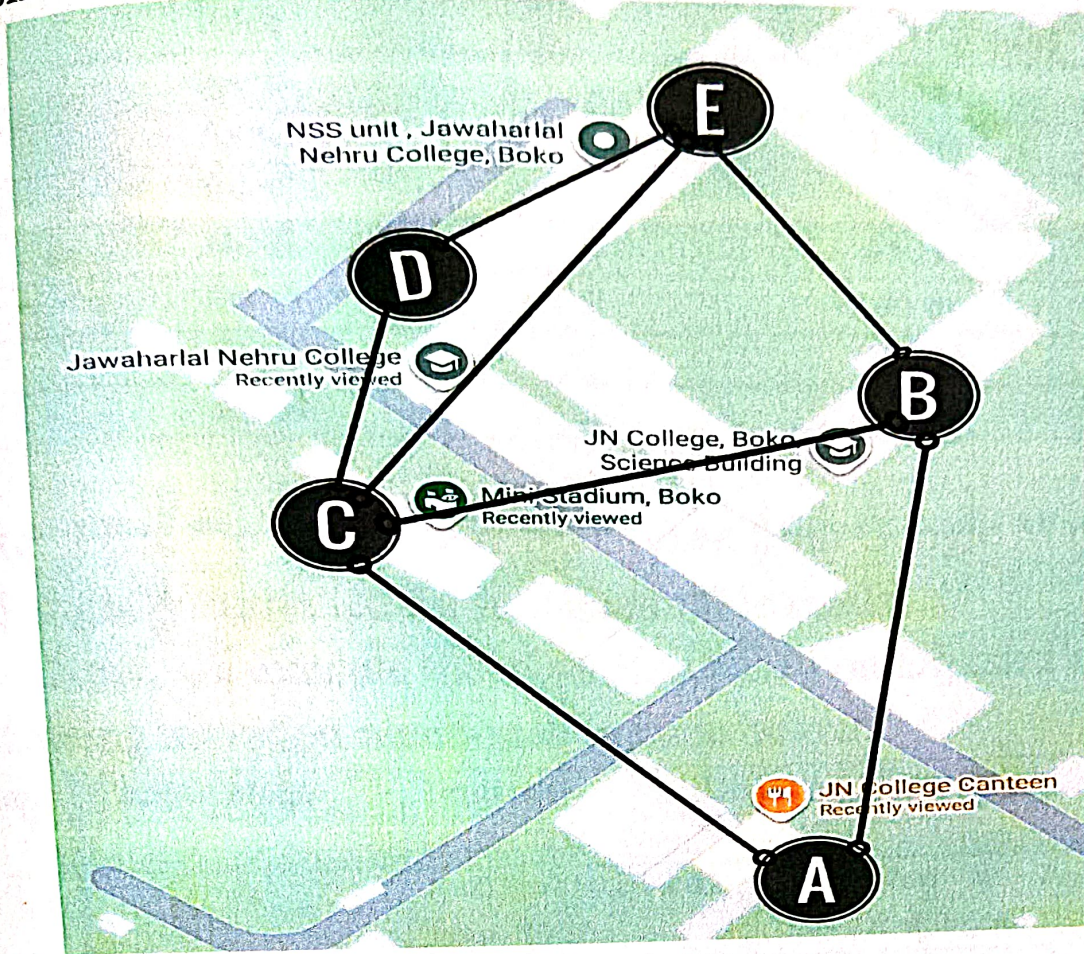
JN College, Second Gate : D

NSS Unit, JN College : E



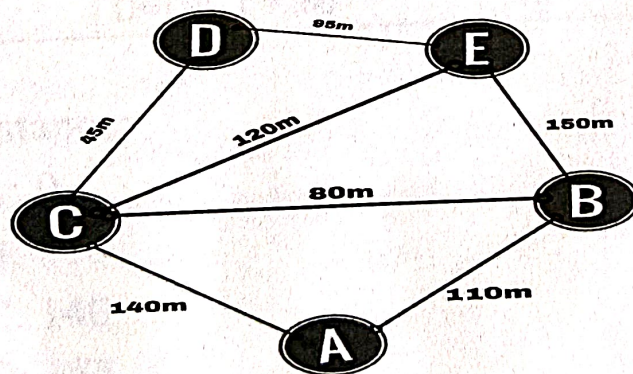
(Figure a)

We shall now join all the vertices with edges as shown below :



(Figure B)

For simplicity, let us draw a more simpler graphical representation of the above situation.



(Figure c)

Now, the edge weights are tabulated below:

Edge	Weight (Distance)
A->B	110 m
A->C	140 m
B->C	80 m
B->E	150 m
C->D	45 m
C->E	120 m
D->E	95 m

Note: Here the values have been taken from Google Maps India and slightly corrected for non-motorable roads.

Now, we execute the algorithm based on Dijkstra's for the following situation mentioned in table below:

Step	Current Node	Tentative Distance (A,B,C,D)	Notes
1	A	A=0,B=110,C=140, D= ∞ , E= ∞	Start at A. Initialize distances.
2	B	A=0,B=110,C=140, D= ∞ , E=260	Process B, update E=110+150.
3	C	A=0,B=110,C=140, D=185, E=260	Process C. Update D=140+45.
4	D	A=0,B=110,C=140, D=185, E=260	Process D. E is not improved.
5	E	A=0,B=110,C=140, D=185, E=260	All nodes processed.

Shortest Distance:

- A to A: 0
- A to B: 110
- A to C: 140
- A to D: 185
- A to E: 260

Project Result:

Dijkstra's algorithm for shortest path: Optimizing campus routes.

Overview

This project explored Dijkstra's algorithm, its history, working principles, and applications in graph theory, particularly in optimizing route-finding within a college campus.

Outcomes

Edge	Path	Shortest Distance
A->B	Canteen to Science building	110 m
A->C	Canteen to Mini Stadium	140 m
A->D	Canteen to JN Collge (second gate)	185 m
A->E	Canteen to NSS Unit	260 m

Results and Insights:

The project demonstrated that Dijkstra's algorithm effectively calculates optimal paths in directed and undirected graphs. Applying it to our college campus highlighted its ability to simplify route-finding, improving overall campus accessibility.

Conclusion :

Dijkstra's Algorithm is a fundamental work in the area of computer science and graph theory that solves the single-source shortest path problem efficiently. Developed by Edsger W. Dijkstra in 1956, it finds the shortest path from a source node to any other node in a graph, both directed and undirected.

In this project, we explored the definition, historical background, and working of Dijkstra's Algorithm and applied it to both directed and undirected graphs. Its limitations, such as not being able to handle negative edge weights, were also discussed. We have noted some of the most interesting applications, like optimizing college route planning where the algorithm makes sure the most efficient travel paths exist between various campus locations.

Through this exploration, it becomes apparent that, despite the limitations of Dijkstra's Algorithm, it remains an important tool for the practicality of different domains, including navigation systems and network optimization. The understanding and implementation of this algorithm also help improve computational problem-solving skills and give prominence to algorithms in addressing everyday problems.

Bibliography

1. Dijkstra, E. W. (1959). "A Note on Two Problems in Connexion with Graphs." Numerische Mathematik,
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
3. Kleinberg, J., & Tardos, E. (2005). Algorithm Design. Pearson Education
4. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
5. <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
6. <https://brilliant.org/wiki/dijkstras-short-path-finder/>
7. <https://maps.app.goo.gl/U98bvEw2VLETQFuz5> (Google Maps , India)