

PROJECT TITLE: ALGORITHM TO SOLVE TRANSPORTATION PROBLEM ON UG CAMPUS

GROUP MEMBERS:

David Livingstone Hini - 10853538

Yankah Kenneth Topp - 10825209

Emmanuel Sekyi - 10818784

Calvin Debrah - 10810118

David Mensah Bonsu - 10843168

Zanita Kukua Morgan - 10848766

Attah Raymond - 10840913

Nana Yaw Safo - 10842582

Michael Oduro - 10852744

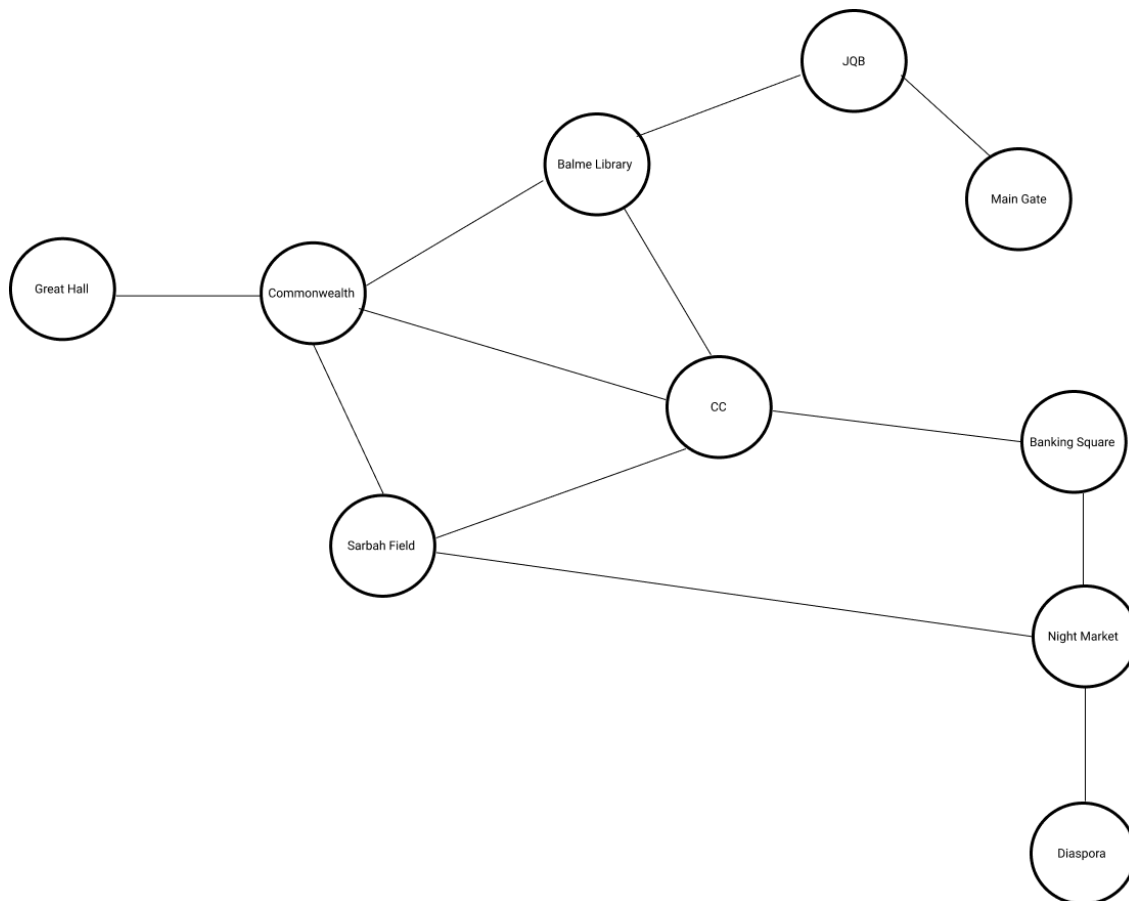
NAME	ID	ROLE	ACTIVITIES PERFORMED	PERCENTAGE CONTRIBUTION
David Livingstone Hini	10853538	Coder	Participated in the coding and design process.	15%
Yankah Kenneth Topp	10825209	Coder	Participated in the coding and design process.	12%
Emmanuel Sekyi	10818784	Tester	Testing of codes during and after development process.	10%
Calvin Debrah	10810118	Advisor	Organized meetings	10%
David Mensah Bonsu	10843168	Tester	Testing of codes during and after development process.	10%
Zanita Kukua Morgan	10848766	Reporter	Reviewing of codes and report writing	12%
Attah Raymond	10840913	Coder	Participated in the coding and design process.	11%
Nana Yaw Safo	10842582	Tester	Testing of codes during and after development process.	10%
Michael Oduro	10852744	Tester	Testing of codes during and after development process.	10%

Summary of Project Description

Our Project is an interactive routing and map system that leverages the power of some specific kind of algorithms(Vogels and Critical path method algorithms) to provide users the shortest possible path or route available to them based on the distance and the time it will take for them to go from their current destination to their final destination. Comparatively it weighs the distances and provides the shortest route that could be cost-effective and fast for the users.

Group's Objectives

- We used Google maps in obtaining the actual distances between nodes or destinations.
- We used the Vogel's approximation method to estimate the cost and the distance between nodes or vertices to find the best route and the critical path method to find the best route in terms of time of arrival, thus before implementing the Dijkstra's algorithm in our code for the shortest path.
- We drew a rough sketch of the unidirectional graph between nodes or landmarks that we chose and implemented it in our code for the map, this is shown below;



Key Features of the System Developed

- The java Swing GUI we used for our UI design provides a real-time interface for users to interact with efficiently.

- The interface comes with a drop-down list that allows users select their source or starting location and also their final destination.
- There's a "Get Direction" button that will provide the best possible route and total distance for travel once a user clicks it.

Algorithms Used

We used the vogels approximation method algorithm for calculating the costs and penalties in order to arrive at a first feasible solution to this problem. The critical path method algorithm was used to estimate or identify the time or task it will take to complete each move from one node to the other. Thus, the source node and neighboring nodes in the graph.

When an initial location is chosen, the user will also have to input their destination and our will find the shortest path between these two locations by implementing all the algorithms stated above.

Algorithm implemented in our java program for the shortest path.

Algo2 Pseudocode:

Algorithm Algorithm1(input, nodeArray, nodeObjects):

Input: input is a String for the user input letter for starting node, nodeArray is a String[] array containing all the node names, nodeObjects is an ArrayList of the node Objects of the Node class.

Output: Prints the path taken to get to the final node Z (including backtracks) and the shortest path to get there.

Create a new ArrayList "path", and a new Stack "shortestPath".

Set the user chosen node as the current node.

While current node is not equal to Z node **do**:

Add current node to "path" and "shortestPath".

Set current node is visited to true.

Initialize new list "compareDD"

For every entry in current node's edges **do**

If edge node is already visited **then**

Continue

Add current edge to list "compareDD"

If "currentDD" is empty **then**

Pop last entry in our stack "shortestPath"

Set the popped node to the current node

Continue

For each element in "compareDD" **do**

If element directDistance is less than minimum directDistance **then**

Set node with minimum directDistance to current node

Print the list "path" and the Stack "shortestPath"

Algorithm we used in obtaining our undirectional graph.

The algorithm used in implementing our graph above is as follows;

function graph(Edge, edge)

 if EDGES \leftarrow edge

 return

 add new edge to init array EDGES

 new Edge \leftarrow edge[destination, distance, source]

 for node in graph

 if node \leftarrow source, add destination to new edge

 for node in graph

if node \leftarrow destination, add source to new edge
return edge [], return node

What we Learnt

Developing this project broadened our understanding of solving transportation problems. We learnt how to implement Dijkstra's algorithm and also get the best arrival time when moving from one location to another.

The Project Development Process

First off, research and planning were done to acknowledge the problem and requirements. After this, we analyzed the problem to determine how we were going to go about it. An algorithm was written to aid us in developing the codes. Coding and design were next and finally, testing of the code was done to ensure quality assurance.

User Interface

Designed with the Java Swing User Interface, it is shown below;

Great Hall

Main Gate

Get Direction

Best Route:
Great Hall -> Commonwealth Hall -> Balme Library -> JQB -> Main ...
Estimated Distance of Travel: 2.440km

7:31 am ✓

7:33 am ✓

it return the

7:35 AM
11/30/2021

Zanita Kukua Morgan

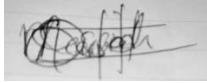
Calvin Debrah

David Livingstone Hini

Emmanuel Sekyi

Michael Oduro

Attah Raymond

A handwritten signature in black ink, appearing to read 'David Mensah Bonsu', with a horizontal line underneath.

David Mensah Bonsu

A handwritten signature in black ink, appearing to read 'Nana Yaw Safo', with a horizontal line underneath.

Nana Yaw Safo

A handwritten signature in blue ink, appearing to read 'Yankah Kenneth Topp', with a horizontal line underneath.

Yankah Kenneth Topp