Assignment 2 Part 1

David Gray s2947315

Contents

1. Problem Statement

The goal of Part 1 of this assignment was to rewrite the Prim's Algorithm function that is of the order O(n^3) to order O(n^2), significantly reducing the time complexity to run the algorithm. Along with this, a few other methods had to be created that set up the graphs and arrays to test both algorithms abilities to create a minimum spanning tree from the data.

2. User Requirements

There are no user requirements for this program apart from running the executable.

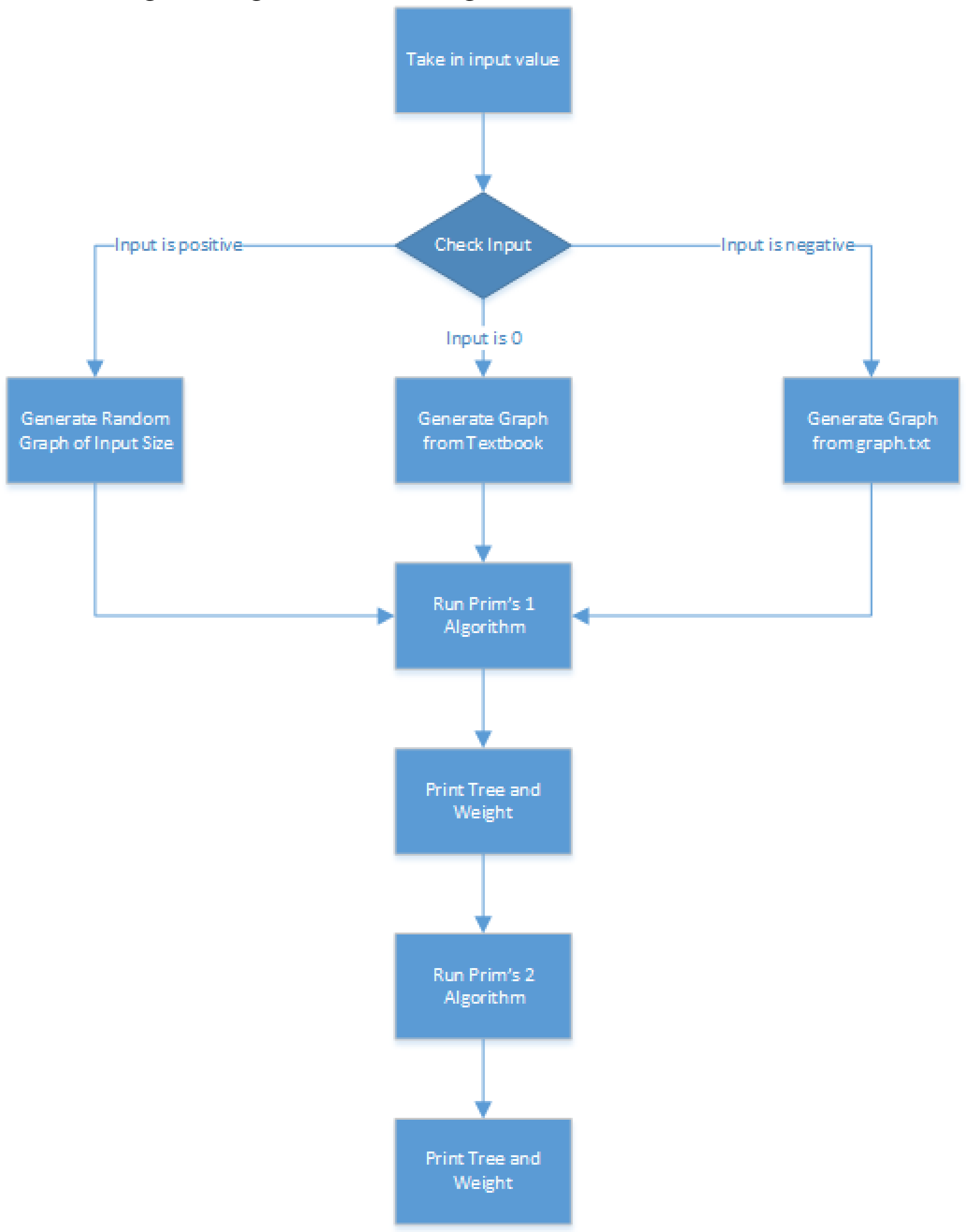
3. Software Requirements

The following outlines the software requirements for the program:

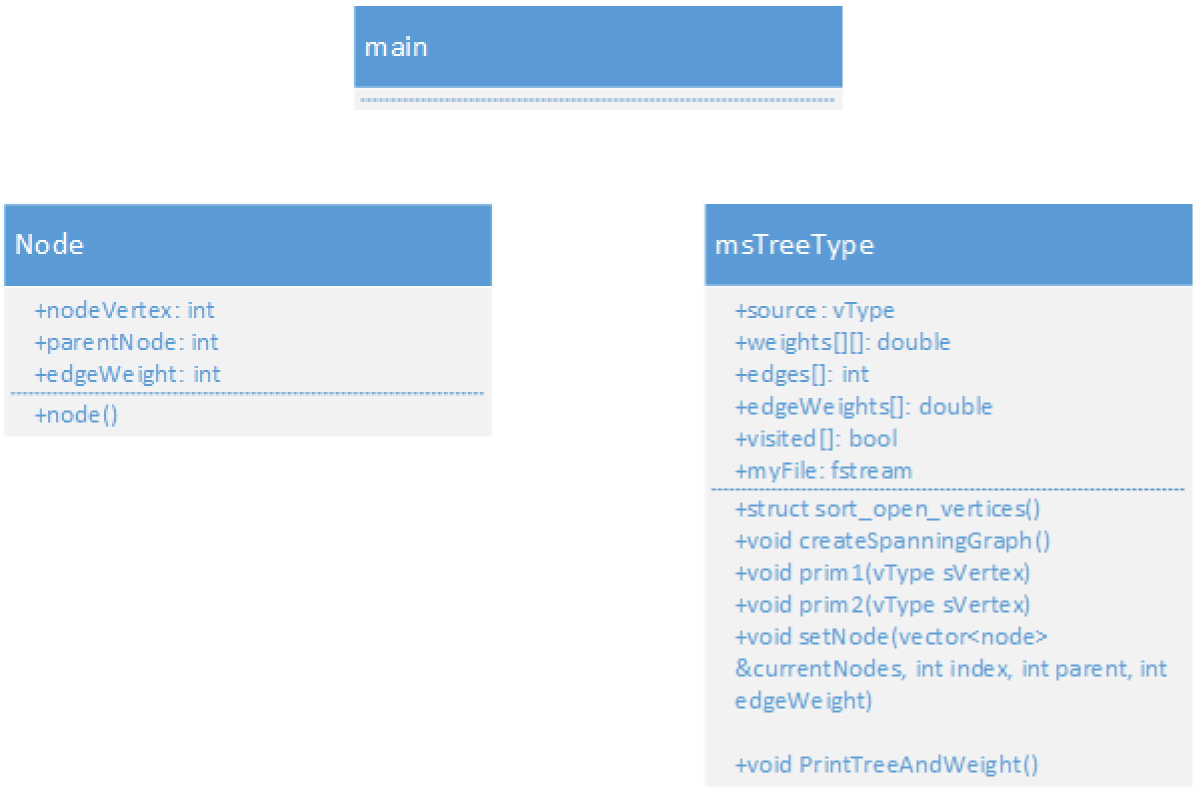
- The program shall take in multiple inputs for whether or not it will use a preset graph or a newly generated graph.
- If the input is 0, the program will create the preset graph in figure 12-16 from the textbook.
- If the input is a negative number, the program will create the preset graph from graph.txt
- If the input is a positive number, the program will randomly generate a graph of the size of the input.
- The program shall read in preset graphs from given text files
- The program shall use both Prim’s Algorithms to create a minimum spanning tree from a graph
- The program shall display the results from the created minimum spanning tree including the edge weights and edge connections

4. Software Design

High Level Design – Logical Block Diagram



Structure Chart


main
Node
+nodeVertex: int
+parentNode: int
+edgeWeight: int
+node()
msTreeType
+source: vType
+weights[][]: double
+edges[]: int
+edgeWeights[]: double
+visited[]: bool
+myFile: fstream
+struct sort_open_vertices()
+void createSpanningGraph()
+void prim1(vType sVertex)
+void prim2(vType sVertex)
+void setNode(vector<node> &currentNodes, int index, int parent, int edgeWeight)
+void PrintTreeAndWeight()

List of main functions in the software.

CreateSpanningGraph(): takes an integer as a parameter. Depending on the input integer value, the program creates; a randomly generated graph the size of the input integer, a graph from graph.txt or a graph from textbookGraph.txt. The method adds to a graph linked list the nodes in a 2D matrix and sets the connection weight values for each node.

PrintTreeAndWeight(): prints out the edges, edge weights, and path total of the minimum spanning tree found using Prim's 1 Algorithm and Prim's 2 Algorithm, along with the 2D matrix of the graph.

prim2(): uses the created graph from the CreateSpanningGraph() to find the minimum spanning tree using the connections of the graph. The algorithm sets up a vector of node objects which contains the node's vertex, its current edge weight and its parent node that it is connected to. The algorithm starts from the source node and checks the connections between it and all the other nodes and updates the values for the edge weight and parent node if required. From that, the vector of nodes is sorted according to the edge weights of each node and the node at the front of the vector is removed and is used as the next start node in the next iteration of the path finding process.

setNode(vector<node> &currentNodes, int index, int parent, int edgeWeight): searches through the vector queue that stores the nodes that have yet to be connected in the minimum spanning tree to find the node that has the vertex value that is passed through to the function. When it has found the correct node, it updates the values within the node object with the values passed into the function.

sort_open_vertices(): is a sort method that is used to sort the vector queue by the edgeWeights value. This is important as the node with the current lowest edge weight is removed from the queue and is used as the next starting node during each iteration of the path finding process in Prim's 2 Algorithm.

List of all data structures in the software. (eg linked lists, trees, arrays etc)

The main data structures this program uses are linked lists, arrays and vectors.

The linked lists are used to store the nodes of the graph that will be used with both Prim's Algorithms.

Arrays are used to store the information of edges, edge weights, node connection weights and whether or not a node has been visited.

Vectors are used as a priority queue system to find the minimum spanning tree of the graph in the Prim's 2 Algorithm. It stores objects of node type which contains the parent of the current node, the index of the current node and the edge weight of the current node. The vector is sorted by the edge weights of the nodes. For each iteration in finding the next path when running Prim's 2 Algorithm, the node with the lowest edge weight is erased from the vector and is used as the next starting node.

5. Requirement Acceptance Tests

Software Requirement No	**Test**	**Implemented (Full /Partial/ None)**	**Test Results (Pass/ Fail)**	**Comments (for partial implementation or failed test results)**
1	Correctly takes in input	Full	Pass	
2	Correctly recognizes input value and creates graph accordingly	Full	Pass	
3	Correctly processes graph and creates minimum spanning tree using Prim’s 1 Algorithm and Prim’s 2 Algorithm	Full	Pass	
4	Correctly prints out edges and edge weights of created minimum spanning tree and 2D matrix of the graph used	Full	Pass	.
5	Correctly finds the minimum spanning tree	Full	Pass	

6. Detailed Software Testing

Test	Expected Results	Actual Results
Tree Creation		
Tested input registration by setting an input value of -1	Program should detect that the input value was negative and the graph.txt example should be used	As expected
Tested random graph creation by setting an input value of 5 and printing out the 2D matrix of the graph	Program should print out the 2D matrix of the randomly generated graph	As expected
Tested graph.txt creation by setting an input value of -1 and printing out the 2D matrix of the graph	Program should print out the 2D matrix of the graph.txt generated graph	As expected
Tested testbookGraph.txt creation by setting an input value of 0 and printing out the 2D matrix of the graph	Program should print out the 2D matrix of the textbookGraph.txt generated graph	As expected
Tested minimum spanning tree of textbookGraph.txt by hand calculating the final path weight and comparing it to the results of Prim's 1 and Prim's 2	Program should return a final weight value of 38	As expected
Tested PrintTreeAndWeight method to correctly print out the edges, edge weights and path total for textbookGraph.txt	Prints out the edges, and edge weights for the path and the path total of 38	As expected

7. User Instructions

If running the program using the Visual Studio 2010, open the project and then run the program in debug mode.

If using the executable, double click on the file and let it run.

When prompted, enter an input value to perform the desired action.