CS210 SPRING 2024

Programming Assignment 3

Points Possible: 100

This assignment implements Prim's algorithm to find the minimum spanning tree.

Given the Adjacency Matrix (2D Array) of a Graph, find its minimum spanning tree using Prim's algorithm.

<u>What is needed?</u>

1. Given G[I][j], this is your graph. All vertex names are integers from 0-N-1.
2. The graph will be initialized and hardcoded into the main function.
3. Maintain 2 Lists (Array / LinkedList) to keep track of Visited and Unvisited vertices (also called Temporary & Permanent Vertices)
4. Pick an arbitrary start vertex set it to visited and check all the edges connected to it. Pick the lowest cost edge, add the destination vertex to the visited array and remove from unvisited array.
5. Add all the edges connected to the newly added visited vertex in the mix to compare for which is the next minimum edge. If the minimum most edge has both visited vertices, remove it from the mix and check the next minimum edge.
6. Implement a Priority Queue using Binary Heap for all the discovered edges to pick the minimum edge every time. When all vertices are visited and the unvisited List becomes empty, the program ends.

<u>Your program will have 3 classes.</u>

1. Node (Reusable Data Independent Class)
    a. This class should be a template
    b. Private variables are:
        i. T* data
        ii. Node<T>* leftChild
        iii. Node<T>* rightChild
        iv. Node<T>* parent
    c. This class must have a constructor, it can be overloaded if you choose to
    d. This class must have a print method.
    e. This class must have a compare method to compare Data values.
    f. Since all variables are private, to access them, you must write set & get methods for each variable:
        i. Set methods have one argument and perform assignment to the variable

ii. Get method returns the value of the variable to the method calling it. Set the appropriate return type.

2. BinaryHeap(Reusable Data Independent Class)
    a. This class should be a template
    b. Private variables are:
        i. Node<T>* root
        ii. int numberOfElements
        iii. int height
    c. Since all variables are private, to access them, you must write set & get methods for each variable:
        i. Set methods have one argument and perform assignment to the variable
        ii. Get method returns the value of the variable to the method calling it. Set the appropriate return type.
    d. This class must have a constructor, it can be overloaded if you choose to
    e. This class must have a destructor that is explicitly written.
    f. This class must also contain the following methods:
        i. insertElement(T* data) - this method inserts the data into the heap and heapifies. This method return nothing.
        ii. deleteMin() - this method finds the smallest element in the tree and returns it to the calling method and heapifies. It returns the data object.
3. Data
    a. Private Variable:
        i. int sourceVertex
        ii. int destinationVertex
        iii. int edgeCost
    b. Since all variables are private, to access them, you must write set & get methods for each variable:
        i. Set methods have one argument and perform assignment to the variable
        ii. Get method returns the value of the variable to the method calling it. Set the appropriate return type.
    c. This class must have a constructor, it can be overloaded if you choose to
    d. This class must have a print method.
    e. This class must have a compare method to compare value of edgeCost.

Your Program will have 1 global constant:

1. VERTEXCOUNT – this represents the number of vertices in the graph. Set this to 5.

Your Program will have 2 global methods:

1. runPrims (int G [VERTEXCOUNT] [VERTEXCOUNT], BinaryHeap* binHeap) - This method runs the prims algorithm on the graph and prints the output.
2. main () - Initialize graph, BinaryHeap and call runPrims method from this method.

Graph: Coded into the main

```
int G[i][j] = {{0, 3, 65, 0, 0},
          {3, 0, 85, 20, 45},
          {65, 85, 0, 41, 77},
          {0, 20, 41, 0, 51},
          {0, 45, 77, 51, 0}};
```

Expected Output: Printed by your program

```
Prim's MST is Edge -> Cost
0 - 1 -> 3
1 - 3 -> 20
3 - 2 -> 41
1 - 4 -> 45
```

Files to be submitted:

Submit a ZIP file (ZIP file must have your First and Last Name) containing the following:

1. C++ File

2. ReadMe.pdf

- Author Name
- Answer the following questions:
  - Explain what a Minimum Spanning Tree is in your own words
  - What are the preconditions to run the Prim's algorithm?
  - What is the input and output of the Prim's algorithm? Write it mathematically.
  - What are some real-world applications of Minimum Spanning Trees?