Hugh A. Miles II

UFID: 78686913

Section: 13A8

09/21/16

COP3530

Minimum Spanning Tree

Summary

- My MST program shows 3 different algorithms that delivers the best MST in a graph.

- Structure
  - Graph
    - Consist of vector of Edges
    - Adjacency list representation
  - Node<type>
    - Constructor (type initValue, int strIndex)
      - In the constructor you need to indicate the value to be set, and where the Node is being placed with the strIndex.
    - Type value
      - The data value stored in side the node.
    - Node<type>* rightChild and leftChild
      - Address for the children of this Node.
  - Edge

        - Constructor(int src, int destination, int weight)
            - Set the objects properties intially
        - Insert(int job)
            - This method add the job to the sum and logs
        - Print()
            - Prints the log and sum
    - Node
        - Constructor(int size)
            - Gives the max number jobs that can be assigned to
        - Insert(int job)
            - This method add the job to the sum and logs
        - Print()
            - Prints the log and sum
    - arrayHeap
        - push()
            - Allows you to add a machine to the heap
        - compare(int parentIndex, int childIndex)
            - checks if the parents is < then its child if not it bubbles down till it finds its position
        - realign()
            - After a pop this starts a bubble down with the last element in the heap
        - top()

        - returns the lowest machine
    - pop()
        - removes the lowest machine
    - isEmpty()
        - bool that tells you if the heap is empty
    - size()
        - gives the amount of machines in the heap
    - printTree()
        - print out the heap in Tree form
    - getMax()
        - returns the biggest machine's sum
    - getRightChild(int parentIndex)
        - gives you right child of current index
    - getLeftChild()
        - gives you right child of current index
    - getParent(int childIndex)
        - gives parent of current childIndex
    - swap(int indexOne, int indexTwo)
        - swaps index positions in place
- minTree
    - push(type value)
        - push Machine in to heap
    - pop()

        - removes root and melds left over subtrees
- top()
    - returns machine at the root of tree
- print()
    - machines in Queue
- printTree()
    - print elements of queue in tree form
- printSubTree(Node<type>* root)
    - print tree from Node given
- meld(Node<type>* currentRoot, Node<type>* toMeld)
    - takes roots of 2 trees and melds toMeld in to the current root
- checkRanks()
    - checks if the tree is leftist and recursively fixs the trees alignment
- getRank(Node<type>* n)
    - get the rank of the current Node n
- swap(Node<type>* parent)
    - swaps the parents children

Test Cases:

Input:

```
4 5
0 1 5
1 3 4
3 2 2
2 0 7
1 2 10
0
```

Output

```
hmiles23@HAM ~/Google Drive/Programming/C++/COP3530 - Data Structs/Turnin/PA4
 master • ./MST < input.txt
Enter number of Node and Edges:
Enter Node A and Node B and Undirected Edge Weight(s):
enter the start Node:
Prim's MST:
[0,1] 5
[1,3] 4
[3,2] 2
Total Weight:
11
Kruskal's MST:
[3,2] 2
[1,3] 4
[0,1] 5
Total Weight:
11
Boruvka's MST:
[0,1] 5
[1,3] 4
[2,3] 2
Total Weight:
11
```