Advanced Data Structures (COP5536)

PROJECT REPORT

SUBMITTED BY:

SYED MUHAMMAD ALI
3816-1305
ali.s@ufl.edu

Contents

Introduction

A system is implemented to find the “n” most popular hashtags that appear on social media such as Facebook or Twitter. In this system the hashtags are given as input from a file and output will be either displayed on the console or in an output file depending on if the user specified an output file name or not while executing the command.

The main structures used for this implementation are a Max Fibonacci Heap and a Hash Table. The Max Fibonacci Heap allows a max priority structure to be implemented and the Hash Table takes the hashtag as a key to access the Fibonacci Heap node containing that hashtag.

Max Fibonacci Heap has an amortized complexity of O(1) for the Increase key, meld, Insert operations and O(logn) for the delete maximum operation. All Fibonacci heap functions discussed in class are implemented however some are integrated as part of other operations instead of having their own method. This allowed an easier implementation for the needs of this system. For the hash table, the HashMap library implementation is used of Java.

Structure

The program starts with the main method of the hashTagCounter class and it can be given either 1 argument, containing the input file name or 2 inputs containing the input as well as the output file name. The input file is then read line by line and if the line starts with a ‘#’ it means the value is to be inserted or perform increase key on it. This is done by calling the insert method the Fibonacci heap class that takes care of this by seeing if the value exists already in the heap and performs increase key, otherwise normal insertion.

If the input file line starts by a number, it means the top n hashtags need to be printed where n is the number on the line. In this case the getTop method is called in the Fibonacci heap. It performs deleteMaximum operation n times to get n top elements and then reinserts them back. This allows the top n hashtags to be obtained and printed to the console or the output file. Each time the getTop method is called, it populates 1 line in the output.

If the input file line starts by a 'stop' keyword, the whole program stops and execution is terminated.

Classes and Methods

There are 3 classes:

1. hashTagCounter
2. Node
3. FibonacciHeapMax

hashTagCounter

This class is the main class of the program.

The method prototype is:

public static void main(String[] args)

The main method is the only method and it starts the whole program. It takes upto 2 string arguments, 1 input file and 1 output file.

Node

This class implements the nodes that will be used by the heap.

The variables are:

1. parent of type Node, it stores the parent node of this node.
2. child of type Node, it stores the child node of this node.
3. left of type Node, it stores the node on the left of this node in the same level linked list.
4. right of type Node, it stores the node on the right of this node in the same level linked list.
5. hashtag of type String, it stores the hashtag name.
6. childLostBefore of type Boolean, it stores the childCut value.
7. frequency of type Integer, it stores the frequency of occurrence of this hashtag.
8. degree of type Integer, it stores the degree of the node.

FibonacciHeapMax

This class implements the actual Max Fibonacci heap.

The variables are:

1. **Root** of type Node, it stores the root node of the Fibonacci Heap.
2. **maxIndicator** of type Node, it stores the maximum value node.
3. **HashTable** of type HashMap, it keeps track of what values are already present in the Fibonacci Heap. If given the hashtag as key, it will return the node containing the hashtag from the HashTable.

The prototypes are:

1. public ArrayList<String> getTop(int topHastagNum)
   This method returns the top n hashtags when provided with an n. It achieves this by checking the maximum node pointer and then deleting max, repeating this n times to get the top n hashtags. After getting the top n hashtags, it reinserts the deleted nodes again into the heap.
   Its input is the integer value of the number of top hashtags needed and it returns a string list of the hashtags.

2. void reinsert(Node CurrentNode)
   This method allows reinsertion of nodes into the heap when deleted temporarily. It takes the node as the input and outputs nothing. Since reinsertion is done at the top level, it inserts the node left of root value.

3. void deleteMaximum()
   This method allows deletion of maximum node and then merges the nodes with same degree to create the new heap. It takes no input and has no output.

4. void joinList(Node CurrentNode)
   This method allows deletion process of node at the same level by removing it from the linked list. It takes the node as input and has no output.

5. void removeNode(Node CurrentNode)

This method allows deletion of a node. It calls the joinList during the process of deletion to complete the process. It takes the node as input and has no output.

6. void insert(String hashtag, int frequency)
   This method performs insertion of a node, if it doesn't already exist or increase key operation if the node exists already. It takes the hashtag of type string as input along with frequency of type integer. It has no output.

7. void CompleteCascadingCut(Node CurrentNode, Node ParentOfCurrentNode)
   This method performs the complete process of cascading cut including removal of a node as well as any further removal of parent nodes that have true childCut value. It takes the node we want to remove as input as well as its parent node. It has no output.