Advanced Data Structures (COP5536)
Fall 2016
Programming Project Report

Divya Yadla

UFID: 94134866

dyadla@ufl.edu

COMPILING INSTRUCTIONS:

In order to compile the file type,

javac HashTagCounter.java

To run the file,

java HashTagCounter inputFile.txt

inputFile.txt is passed as a command line argument to the program.

CLASSES AND FUNCTION PROTOTYPES:

There are 2 classes that have been used to implement this project. They are:

- FiboHeapNode
- FiboHeap
- HashTagCounter

Here's a short description of the functionality of each class.

FiboHeapNode

The class FiboHeapNode mainly describes the structure of the node that is used in the implementation of the Fibonacci Heap.

The variables each represent the attributes of a node. They are:

- **element:** element stores the hashtag.
- **key:** key stores the Fibonacci node number.
- **degree:** Stores the number of nodes that are present in the next level.
- **childCut:** Specifies whether the node has already lost a child or not. It is of Boolean type.

This class has other variables such as parent, child, right and left which are used for the circular doubly linked list that is created at each level.

The member functions of this class are:

- **int getKey():** It returns the value of the Fibonacci Node.
- **String getTag():** It returns the Hashtag.

FiboHeap

The methods to implement the Fibonacci heap are defined in this class. All operations in this class are performed on instances of FiboHeapNode class.

The member functions of this class are:

- **FiboHeapNode insert(FiboHeapNode n):** Used to insert a new node into the Fibonacci heap.
- **void increaseKey(FiboHeapNode n, int k):** Used to increase the value of n to k which is passed as an argument to the function.
- **void cascadingCut(FiboHeapNode t):** Checks the childCut value and performs necessary changes and removes the node and adds it to the root level if necessary.
- **void cut(FiboHeapNode a, FiboHeapNode b):** Removes the node from the doubly linked list.
- **FiboHeapNode removeMax():** Removes the node with the maximum frequency from the Fibonacci heap.
- **setNewMaxPointer():** Updates the pointer to the node with maximum value in the Fibonacci heap.
- **void pairWiseCombine():** Combines two nodes of same degree into a single heap. pairWiseCombine() is performed every time an increaseKey() operation is performed.
- **void link(FiboHeapNode a,FiboHeapNode b):** Takes care of all the links in the doubly linked list whenever a cut is performed.

HashTagCounter

The HashTagCounter class contains the main function that runs the application.

The input is taken from a file and the operations on a heap are performed. This function writes output to a file after performing removeMax operation. This class is the encapsulating class for all other classes of the program. It contains the instances of all other classes to access their member functions.

PROGRAM STRUCTURE:

```
public class HashTagCounter{
        class FiboHeapNode(){
                FiboHeapNode(String tag, int frequency);
                String getTag();
                int getKey();
        }
        class FiboHeap(){
                FiboHeapNode insert(FiboHeapNode node, int key);
                void increaseKey(FiboHeapNode p, int value);
                void cascadingCut(FiboHeapNode p);
                void cut(FiboHeapNode c, FiboHeapNode p);
                FiboHeapNode removeMax();
                void setNewMaxPointer();
                void pairWiseCombine();
                void link(FiboHeapNode p, FiboHeapNode q);
        }
        public static void main(String args[]){
                FiboHeapNode node;
                FiboHeap heap;
        }
}
```

CONCLUSION:

The order of the retrieval of hashtags varies when the hashtags have the same frequency after performing the increaseKey operation. The program runs for a million inputs and produces the output in less than 60 secs. It accurately retrieves the n most popular hashtags from a given input of hashtags and their frequencies.