Advanced Data Structures
Spring 2020
Project Report

Rishabh Aryan Das

UFID: 0536-9273

rishabh.das@ufl.edu

PROJECT DESCRIPTION

Implement a hashtagcounter to find the n most popular hashtags that appear on social media such as Facebook or Twitter. For the scope of this project hashtags will be given from an input file. Basic idea for the implementation is to use a max priority structure to find out the most popular hashtags.

Two data structures are used as follows:

1. Max Fibonacci heap: to keep track of frequencies of hashtags
2. Hash table: The key for the hashtable is the hashtag and the value is the pointer to the corresponding node in the Fibonacci heap

INSTALLATION

Steps are as follows:

1. cd das_rishabh
2. make
3. java hashtagcounter input_filename.txt

Output file will be generated as output_file.txt if no output filename is specified.

STRUCTURE OF THE PROGRAM

1. The driver class (hashtagcounter) has the main method which accepts two command line arguments in the order input_filename.txt output_filename.txt

2. It processes each line of the input file invoking the Fibonacci heap's addNode routine or increaseKey routine or deleteMax routine

3. The addNode routine inserts a node into the heap.

4. The increaseKey routine is invoked when we try to insert a hashtag that is already present in the heap. It increases the frequency of the node passed by the amount. If the resulting node's key gets increased more than that of it's parent then this routing calls cascadeCut routine to preserve the max heap properties.

5. The deleteMax routine is invoked when there is a query in the input file. It removes the maximum node from the heap and performs a pairwiseCombine to combine the child nodes in the upper circular list.

6. A 'stop' keyword terminates the code.

DOCUMENTATION

There are three classes as follows:

1. hashtagcounter
2. FibonacciHeap
3. Node

1.1 **hashtagcounter**

***Description*:** This is the driver module of the program

Methods:

public static void main(String[] args)

Description	This is the main method which takes an input file from the commandline arguments and invokes the routines of the Fibonacci heap accordingly
Arguments	args- accepts a input filename and a output filename
Return	void

public static void process()

Description	This routine is invoked when the program encounters a query in the input file. This routines removes the maxNode from the Fibonacci heap and creates the output string
Arguments	o/p filename,list of nodes removed,Fibonacci heap,

	queryString,hashtable containing the nodes
Return	void

public static void writeFile()

Description	This method writes the output string to the specified output file
Arguments	Output filename, output string
Return	Void

4.2 **FibonacciHeap**

***Description*:** This is the implementation of max Fibonacci heap in java. Supports addNode O(1), deleteMax O(logn), increaseKey O(1) routines in amortized time complexity.

Methods:

public Boolean isHeapEmpty()

Description	This method checks if the heap is empty or not.
Arguments	NIL
Return	Boolean

public void addNode()

Description	This method inserts a node into the max fibonacci heap
Arguments	newNode, key of the new node
Return	Void

Public node increaseKey()

Description	This node increases the frequency of the existing node in the fibonacci heap by the value provided in the input
Arguments	Node , newKey
Return	Current node

public node deleteMax()

Description	This method removes the max node from the fibonacci heap and merges its children in the upper circular linked list
Arguments	NIL
Return	Removed Node

protected void cut()

Description	This method removes the child node from the child list of parent node
Arguments	Child, parent
Return	Void

Protected void cascadeCut()

Description	This method cuts off a child node from its parent till a parent with childcut value false is encountered
Arguments	Child node

Return	Void

Protected void pairwiseCombine()

Description	This method combines the trees with equal degrees
Arguments	NIL
Return	Void

4.3 **Node**

Description: This is the implementation of the node which has hashtag,key,degree,childcutValue, parentNode,childNode,leftNode and rightNode

Methods:

public final int getKey()

Description	Returns key of the node
Arguments	NIL
Return	Key

public final string getHashtag()

Description	Returns the value of the node
Arguments	NIL
Return	hashtag