COP 5536
Advanced Data Structures

Programming Project, Spring 2020

HASH-TAG COUNTER

Submitted by:

Sai Venkatesh Kadiyala

UFID: 3938-5216

saikadiyala@ufl.edu

Project Description:

We are required to implement a system 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. The basic idea for the implementation is to use a max priority structure to find out the most popular hashtags.

The project is written in JAVA.

Implementaion:

The following data structure concepts have been used for the implementation keyword counter system:

• Max Fibonacci heap: To keep track of the frequencies of keywords
• Hash Table: To store keywords as keys and pointer to the corresponding node in the Fibonacci heap as values.

Program and Function Description:

The program has 3 JAVA class files and a Makefile as follows:
hashtagcounter: This class contains the main function to read the input file and writes out the required output.
Node: This class contains the Node which is used in Max Fibonacci Heap and also its methods
MaxFibonacciHeap: This class contains all the functions and methods of Max Fibonacci Heap.
Makefile: This file is used to compile the .java through command line and to create "hashtagcounter" executable.

Structure of Node:

Each node in the max Fibonacci heap has the following class variables:
frequency - To store the frequency of a hashtag, the data type is int
nodeDegree - To store the degree of the node to determines the number of child nodes corresponding to the node. The data type of this variable is int.
parent- Node object which points the parent of the corresponding Node.
child- Node object which points the child of the corresponding Node.
leftSib,rightSib– Node objects that point the node's left sibling and right Sibling int the circular list.
Childcut- Boolean data type, to store the child cut value of the node.

The following constructor and functions are present in Node.java:

Node()

Description	Node constructor to initialize the Node. It stores the frequency of corresponding hashtag in it and has pointers to left and right sib
Parameters	Int – frequency of hashtag
Return Type	

int getValue()

Description	This method gets the returns the Frequency of corresponding Node
Parameters	-
Return	int type, returns the frequency stored in the node.

setValue()

Description	This method sets/updates the returns the Frequency of corresponding Node
Parameters	int type, takes in frequency value that is to be stored
Return	-

getDegree()

Description	This method get the returns the degree associated with the Node
Parameters	-
Return	int type, degree of the node

setDegree()

Description	This method update/sets the returns the degree associated with the Node
Parameters	int type, the desired new degree
Return	int type, degree of the node

Structure of MaxFibonacciHeap:

Contains the class variables – maxNode which is node object and also stores the heapSize ,int type which is nothing but the number of nodes in the heap.

The following functions are present in MaxFibonnaciHeap.java:
createNode()

Description	This method creates a Node in the MaxFibonacciHeap and stores the frequency.
Parameters	int type, value to be stored
Returns	Node object

increaseKey()

Description	This method updates the frequency of existing node, perform cut operation followed by cascade cut operation if the element is not the root and new frequency is greater than its parent's frequency. It also sets updates max pointer if needed.
Parameters	Node - the node storing the frequency of a hashtag which is to be incremented. int – Frequency of hashtag by which it has to be incremented.
Returns	-

cascadingCut()

Description	This method examines the childCut value of node and if is true, it will be moved to top level and its childCut value will be set to false.
Parameters	Node- to the node which is being examined.
Returns	-

removeMax()

Description	This method will remove the current maxNode from the MaxFibonaccHeap. If the maxNode has any child nodes, they all will be moved to the top level. This has the pairwise combining embedded into it
Parameters	-
Returns	Returns the Max Node of the MaxFibonnaciHeap

merge(Node A, Node B)

Description	This method sets the Node with less frequency as a child to the Node with higher frequency and increments the degree of the Node which will become the parent.
Parameters	The nodes to be Merged
Returns	Returns the node with higher frequency set as parent

size()

Description	This method is to find the size of the heap of a MaxFibonacciHeap
Parameters	-
Returns	int type, returns the heap size

Structure and functioning of hashtagcounter:

The hashtagcounter.java file contains the main() function which takes input file and outputfile as arguments check if there is an output file specified in the command line.

It prints the output to file/console depending on the input.

It also checks for errors in the input arguments such as the filename, any error in the reading or writing.

It also uses the MaxFibonacciHeap data structure to store frequencies in nodes, Hashtable to store the appropriate hashtags as key and the pointer to frequency in the MaxFibonacciHeap in the value.

Input:

As specified , Hashtags appear one per line in the input file and start with # sign. After the hashtag an integer will appear and that is the count of the hashtag (There is a space between hashtag and the integer). You need to increment the hashtag frequency by that count. Queries will also appear in the input file and once a query appears you should append the answer to

the query to the output file. A query appears as an integer number (n) without # sign in the beginning. The answer to the query n is n hashtags with the highest frequency. These should be written to the output file. An input line with the word “stop” (without hashtag symbol) causes the program to terminate.

Output:

For each query n, you need to write the n most popular hashtags (i.e., highest frequency) to the output file in descending order of frequency (ties may be broken arbitrarily). The output for a query should be a comma separated list occupying a single line in the output “output_file.txt”. There should be no space character after the commas.

Result: