Advanced Data Structures

COP5536

Chen Yang

UFID: 5210-9967

cyang3@ufl.edu

Problem Description

You 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. Basic idea for the implementation is to use a max priority structure to find out the most popular hashtags.

You must use the following data structures for the implementation.
1. Max Fibonacci heap: use to keep track of the frequencies of hashtags.
2. Hash table: The key for the hash table is the hashtag, and the value is the pointer to the corresponding node in the Fibonacci heap.

You will need to perform the increase key operation many times as keywords appear in the input keywords stream. You are only required to implement the Fibonacci heap operations that are required to accomplish this task.

Input and Output Requirements

Your program is required to take the input file and output file names as arguments.
With Java $java hashtagcounter <input_file_name> [output_file_name]

Input Format

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. The following is an example of an input file.

Output Format

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.

Execution Instructions

1. Unzip Chen_Yang.zip
2. Locate the unzipped file and using 'cd' on the terminal
3. Use '$make' to make file.
4. Run the program by using java command: $java hashtagcounter <input_file_path_and_name>

Class Structures and Description

The whole project consists of 3 classes: hashtagcounter, FibonacciNodeStructure, FibonacciHeap.

hashtagcounter:

This is the main part of program and this is the beginning of the program. It contains the main method. This class includes the reading input file and executing the corresponding query. The function in this class are listed as follows:

1) Take file as input
2) Parse file and save file in an ArrayList
3) Execute Query function over the ArrayList
4) Return an output file as result

MaxFibonacciHeap:

This class consists all the operations of a normal Max Fibonacci Heap.
The methods are listed as follows:

1) insertion: insert a new element into the Fibonacci max heap, and keep its feature.
2) removeMax: remove the max node in the Fibonacci heap, then pairwise according to the situation to keep the heap's feature.
3) compareandCombine: pairwise and merge those trees of the same degree.
4) increaseKey: increase the node data value
5) cascadingCut: inserts the node into top level doubly linked list if parent's childCut flag is false, then set it to true, else do cascading cut on its parent.

FibonacciNode:

This class contains the basic structure of a Fib-heap.
The fields in this class are listed as follows:

1) degree: the degree of a node
2) hashtag: the hashtag stored in the node
3) data: the frequency of the hashtag
4) childpointer: the pointer to child of the node
5) leftSibling: the pointer to the left sibling of the node
6) parentNode: the pointer to the parent of the node
7) rightSibling: the pointer to the right sibling of the node
8) childCut: flag for the cascading cut check

Function Prototypes:

Class: hashtagcounter

Function	Description	parameters	Return
public static void main	The main function that the program starts. This function will receive a file as input and then parse the file and save the queries in a ArrayList. Finally return the result of program to output_file.txt	String[] args	void

Class: MaxFibonacciHeap

Function	Description	parameters	Return
Public void insertion	This function will insert a new element into the Fibonacci heap. The location of insertion is root level. If there is no root in the heap or the node to be inserted has greater frequency than the current max frequency, then replace the max frequency node.	FibonacciNode node	Node inserted in the current heap
public FibonacciNode removeMax	This function will remove the max frequency node in the Fibonacci heap. If there is only one node in the heap, then remove this node and update the max pointer to null. Else, insert the children of the node to be removed at the bottom level then remove this node. After that, pairwise the heap and update the max pointer.	integer	Node to be removed
private FibonacciNode compareandCombine	This function will pairwise and merge those trees of the same degree. First, traverse all the nodes in the root level, then keep a hash which keeps the degree of tree till now. For each node do so: check if there are any same degree nodes, if yes merge the two nodes by checking which of them has a greater frequency, set the greater one as parent and set the less one as child. Finally, log the parent node with new degree in hash.	FibonacciNode node1, FibonacciNode node2	void
public void increaseKey	This function will increase the node date value, if the node to be inserted has greater frequency than its parent, then swap them to keep the max Fibonacci heap feature.	FibonacciNode node, int d	void
public void cascadingCut	This function will inserts the node into top level doubly linked list if parent's childCut flag is false, then set it to true, else perform cascading cut on parent node.	Fibonacci Node node	void

Class: FibonacciNode

Function	Description	parameters	Return
public int getData	This function will return the data	null	data
public void setData	This function will update the data	Int data	void
Public FibonacciNode getChildptr()	This function will return the children of the node	null	childpointer
public void setChildptr	This function will update the child node of the target node	FibonacciNode childpointer	void
public FibonacciNode getLeftSibling	This function will return the left sibling node of the target node	null	FibonacciNode leftSibling
public void setLeftSibling	This function will update the left child node of target node	FibonacciNode leftSibling	void
Public FibonacciNode getParent	This function will return the parent node of the target node	null	FibonacciNode RightSibling
public void setParent	This function will update the parent node of the target node	FibonacciNode parentNode	void
public FibonacciNode-getRightSibling	This function will return the right node near by the target node, if the target is the last one of its level, then return the first node in the next level.	null	FibonacciNode RightSibling
public void setRightSibling	This function will update the right node near by the target node, if the target is the last one of its level, then update the first node in the next level.	FibonacciNode rightSibling	void
public boolean getChildCut	This function will return the childcut of the node	null	boolean
public void setChildCut	This function will update the childcut of the node	FibonacciNode ChildCut	void
public int getDegree	This function will return the degree of the node.	null	Degree
public void setDegree	This function will update the degree of the node	int degree	void
public String getHashtag	This function will return the data in the target node.	null	hashtag
public void setHashtag	This function will update the data in the target node.	String hashtag	void