Advances Data Structures (COP 5536)
Hashtag Counter Implementation

Spring 2020

Kratika Singhal

Email: singhalk@ufl.edu

UFID: 6953-5971

Contents

Project Description

The aim of this project is to implement a system to find the n most popular hashtags that appear on the social media. For the scope of the project, the hashtags are given from an input file. The output is written to an output file or the console as per the arguments given.

The program uses the following data structure for the implementation of the project.

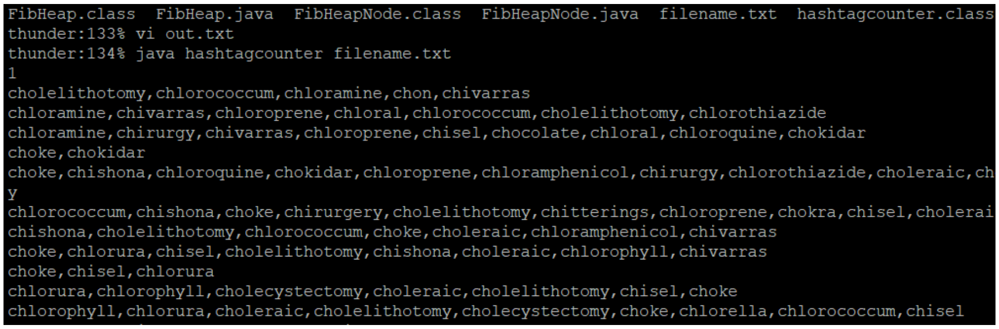
1. Max Fibonacci heap – keeps the track of the frequencies of hashtag.
2. Hash table – stores the hashtags as keys and the pointer to the corresponding node as the value.

The program is designed in such a way that it can run for a million hashtag inputs with accuracy and in efficient running time. Max Fibonacci heap data structure is used because it has an amortized complexity of O(1) for the increase key operation which is performed number of time when any repeated hashtag is encountered.

Input and Output Requirements

The program code takes an input text file. It reads the input file name as an argument. Example
$ java hashtagcounter input_file.txt
If the output file name is not mentioned, the program writes the output to the console.

Example: $ java hashtagcounter <input_file.txt>



```
FibHeap.class  FibHeap.java  FibHeapNode.class  FibHeapNode.java  filename.txt  hashtagcounter.class
thunder:133% vi out.txt
thunder:134% java hashtagcounter filename.txt
1
cholelithotomy,chlorococcum,chloramine,chon,chivarras
chloramine,chivarras,chloroprene,chloral,chlorococcum,cholelithotomy,chlorothiazide
chloramine,chirurgy,chivarras,chloroprene,chisel,chocolate,chloral,chloroquine,chokidar
choke,chokidar
choke,chishona,chloroquine,chokidar,chloroprene,chloramphenicol,chirurgy,chlorothiazide,choleraic,ch
y
chlorococcum,chishona,choke,chirurgery,cholelithotomy,chitterings,chloroprene,chokra,chisel,cholerai
chishona,cholelithotomy,chlorococcum,choke,choleraic,chloramphenicol,chivarras
choke,chlorura,chisel,cholelithotomy,chishona,choleraic,chlorophyll,chivarras
choke,chisel,chlorura
chlorura,chlorophyll,cholecystectomy,choleraic,cholelithotomy,chisel,choke
chlorophyll,chlorura,choleraic,cholelithotomy,cholecystectomy,choke,chlorella,chlorococcum,chisel
```

If the output file name is specified as an argument, the output text file is created in the same folder where the source code is present.
Example: $ java hashtagcounter <input_file.txt> <output.txt>

Programming Environment

The project is implemented in JAVA. The program is compatible with standard java compliers. The program has also been tested on thunder.cise.ufl.edu server.

Hardware Requirement

Hard Disk space: 4 GB
Memory: 512 MB minimum
Operating System: Windows, LINUX/UNIX/MAC OS

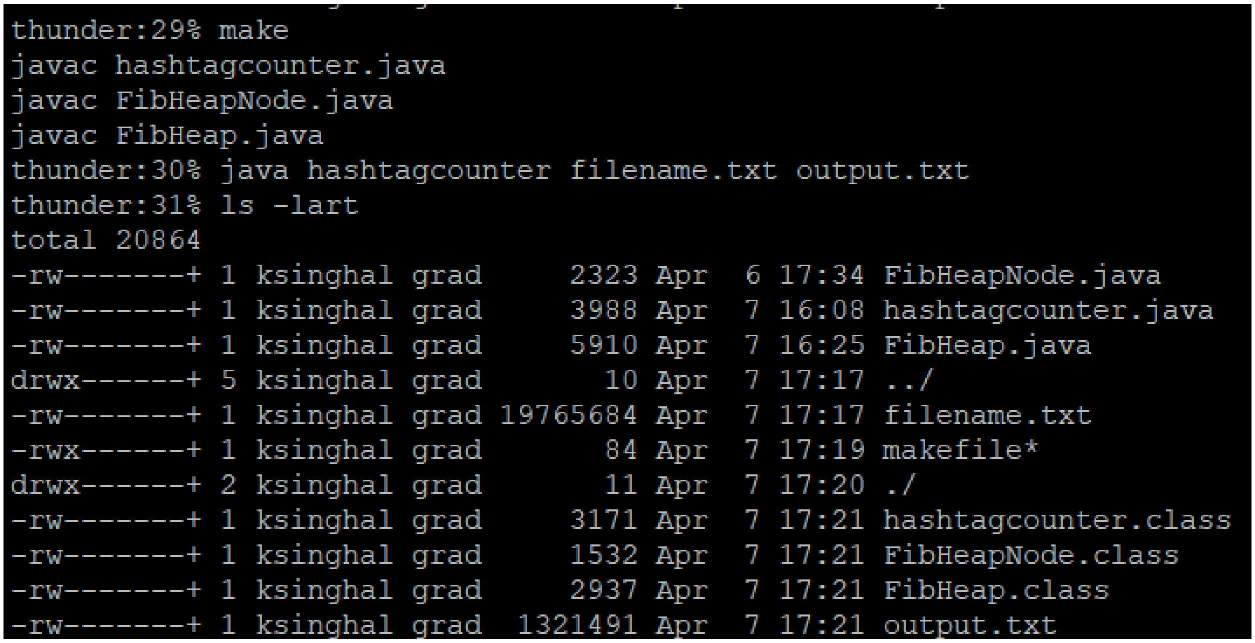
Procedure to Compile and Execute on Server

The program source code has been tested and executed intensively using standard java compiler on local machine as well as on thunder.cise.ufl.edu server. The program compiles and runs smoothly on both the platforms.

Steps to execute the program on thunder

1. Connect to gatorlink vpn and connect to thunder using SSH server.
2. Transfer and unzip the source code onto thunder.
3. Execute the makefile using command – make
4. It will compile the java files.
5. Now run the java program using command –
   Java hashtagcounter <inputfile.txt> <outputfile.txt>

Please refer to the below image showing the procedure to run the Java code. The output file is generated as shown below.



```
thunder:29% make
javac hashtagcounter.java
javac FibHeapNode.java
javac FibHeap.java
thunder:30% java hashtagcounter filename.txt output.txt
thunder:31% ls -lart
total 20864
-rw-------+ 1 ksinghal grad     2323 Apr  6 17:34 FibHeapNode.java
-rw-------+ 1 ksinghal grad     3988 Apr  7 16:08 hashtagcounter.java
-rw-------+ 1 ksinghal grad     5910 Apr  7 16:25 FibHeap.java
drwx------+ 5 ksinghal grad       10 Apr  7 17:17 ../
-rw-------+ 1 ksinghal grad 19765684 Apr  7 17:17 filename.txt
-rwx------+ 1 ksinghal grad       84 Apr  7 17:19 makefile*
drwx------+ 2 ksinghal grad       11 Apr  7 17:20 ./
-rw-------+ 1 ksinghal grad     3171 Apr  7 17:21 hashtagcounter.class
-rw-------+ 1 ksinghal grad     1532 Apr  7 17:21 FibHeapNode.class
-rw-------+ 1 ksinghal grad     2937 Apr  7 17:21 FibHeap.class
-rw-------+ 1 ksinghal grad  1321491 Apr  7 17:21 output.txt
```

Program Structure

The source code consists of three classes

1. FibHeapNode
2. FibHeap
3. HashtagCounter

The main class hashtagcounter.java reads the input file line by line and writes the output in the output file or the console. It creates a hash table which stores the hashtag as the key for the hash table and the value as the pointer to the corresponding node in the Fibonacci heap. As the input is read, various patterns of the hashtags, query and stop are taken into account. When new hashtags appear, the corresponding key and value is stored in the hash table. When a hashtag repeats, the count is increased, the function increase key is invoked. When a query is encountered, the method extractMax() calculates the maximum occurring hashtag. Finally, when a stop is encountered, the processing terminates and the output is written.

FibHeapNode.java

FibHeapNode class defines the structure of the max Fibonacci heap node. It instantiates the objects of a node in the memory. It defines various attributes of a node defined the following table.

Defined Variables

Variable Name	Data type	Description
key	integer	Key value which is to be stored in the node
tag	string	Stores the hashtag
deg	integer	Stores the degree of a node. Degree is the count of number of children the node contains.
leftptr	FibHeapNode	It is a pointer to the to the node's left sibling in the doubly linked list
rightptr	FibHeapNode	It is a pointer to the node's right sibling in the doubly linked list
parent	FibHeapNode	It is a pointer to the parent node of the corresponding node.
Childptr	FibHeapNode	The variable is a pointer to any one of the children nodes.
childCut	boolean	A child cut value indicates if the node has lost any child.
nodeCount	integer	Keeps the count of the node

Defined Methods

Method Name	Return type	Parameters	Description
FibHeapNode	-	key	constructor to create a Fibonacci Node structure by initializing key, child cut indicator and degree
Getkey	integer	-	Returns the child pointer of the node
setKey	void	key	Sets the key of the node
getchildPtr	FibHeapNode	-	Returns the child pointer of the node
setchildPtr	void	childPtr	Sets the child pointer of the node
getleftPtr	FibHeapNode	-	Returns the left pointer of the node
setleftPtr	void	leftPtr	Sets the left pointer of the node
getParent	FibHeapNode		Returns the pointer to the parent
setParent	void	Parent	Sets the parent of the node
getRightPtr	FibHeapNode	-	Returns the right pointer of the node
setRightPtr	void	rightPtr	Sets the right pointer of the node
getChildCut	boolean	-	Returns the child cut value. Returns either true or false
setChildCut	void	childCut	Sets the child cut value for the node
getDeg	integer	-	Returns the degree of the node
setDeg	void	deg	Sets the degree of the node
getTag	string	-	It returns the tag of Node that invoked this method.
setTag	void	tag	Sets the hashtag string

FibHeap.java

FibHeap.java class defines various operations that can be performed on the max Fibonacci heap. The methods described in this class performs the insert, increase key, remove max, consolidate and other operations which can be performed on the Fibonacci heap data structure. The list of variables and methods defined in the class are as follows.

Defined Variables

Variable Name	Data type	Description
Max	FibHeapNode	Max points to the node in the root list whose key value is maximum

Defined Methods

Method name	Return type	Arguments	Description
insert()	Void	FibHeapNode n	This function inserts a new node in the max Fibonacci heap. It initializes the structural attributes of node and checks weather the heap is empty, if yes, it makes new node the only node of the heap. Otherwise the new node is inserted accordingly at its pointer. The max pointer is updated if required.
extractMax()	FibHeapNode	-	The function removes the max element node from the heap and returns it. It starts by saving pointer to the maximum node and returns it at the end. While the max node may have children and may need restructuring hence puts the children in the root list and calls the consolidate() method.
consolidate()	FibHeapNode	FibHeapNode n1, FibHeapNode n2	The consolidate function combines the trees with similar degree until every tree has a distinct degree. It combines the same degree trees by comparing the key values and returns the combined tree. It updates the max pointer according to the requirement.
increaseKey()	Void	FibHeapNode n, int k	The function increases the key value of the node by a given value. If the increased key is greater than the parent key, cascadeCut() method is called for further processing.
cascadeCut()	Void	FibHeapNode n	This method checks the child cute flag. If the child cut value is false for the parent, make it true else recurses its way up the tree until it finds either a root or an unmarked node.

hashtagcounter.java

hashtagcounter.java is the main class reads the inputs line by line and produces the output. It makes call to various methods in other two classes as per requirement.

Defined Variables

Variable name	Data type	Description
fr	FileReader	It reads the contents of a file as a stream of characters
tag	String hashtable	This is the has table in which the key is the hashtags and the value is the pointer to the corresponding nodes
deletedNotes	FibHeabNode arraylist	It is a dynamic array to store deleted or removed nodes
br	BufferedReader	It is an object of BufferedReaderJava class to reads the text from an Input stream
wr	PrintWriter	Writes a formatted string as the output.
outputFile	string	Takes the argument as input. Stores the name of the output file given in the argument.
temp	string	Local temp variable required for the processing
p	Int	Local temp variable required for the processing
str	String	Local temp variable required for the processing
node	FibHeapNode	Stores the max element, extract by invoking the extractMax method

Result

The program can write the output both to the console or the output file as per the requirement. For each query n, the program writes the n most popular hashtags (the one with highest frequency) to the output file in descending order of frequency. Ties(similar hashtag encounter) in this project are taken care arbitrarily. The output for a query is a comma separated list occupying a single line in the output. The code is capable of running for a million input hashtags in efficient time.

Conclusion

The objective of the project has been met. The project has successfully implemented the max Fibonacci heap data structure for the calculation of popular hashtags on the social media.

References

- *Introduction to Algorithms* by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Third edition.
- *Stack overflow* https://stackoverflow.com/