PROGRAMMING PROJECT

COP5536 Advanced Data Structures

Abstract

Implemented a system to find the n most popular hashtags appeared on social media such as Facebook or Twitter.

Akshay Sharma

akshay.sharma@ufl.edu

UFID: 4195 9254

Contents

Project Description

We have implemented a system to find out n most popular hashtags appearing on social media like Facebook or Twitter. For the scope of this project, the input to the project will be given in a text file. The project has been implemented in java and the name of the input fill be supplied as a command line argument while running the program. The program will generate a text file output_file.txt with the most popular hash tags.

Working Environment

Minimum Hard Disk Space   : 4 GB

Minimum Memory   : 512 MB

Compiler Used   : java version "1.8.0_101"

                                                  : Java(TM) SE Runtime Environment (build 1.8.0_101-b13)

Compiling Instructions

The project has been compiled and tested on thunder.cise.ufl.edu.

For compiling and running the program(on supplied sample input), run the following commands

$ unzip Sharma_Akshay.zip

$ make

$ java hashtagcounter sampleInput.txt

The program will generate the output_file.txt which will contain the result.

Input Requirements

The program is required to take the input file as an argument. Following is an example of a program that read from a file named file_name.

Run command:

                    java hashtagcounter file_name

Output Requirements

Once a query is appeared the program writes down the most popular hashtags of appeared number in to

the output file in descending order. Output for a query is comma separated list without any new

lines. Once the output for a query is finished the program puts a new line to write the output for another

query. The program produces all the outputs in the output file named "output_file.txt".

Structure of the program and function descriptions

There are 4 classes that I have used to implement the programming assignment. Out of these 4 classes hashtagcounter.java has been mentioned in the requirements and is the class that contains the main function of the program. The other 3 classes FiboNode.java, FiboTree.java, HashTag.java are used for the implementation of the max Fibonacci heap data structure.

hashtagcounter.java:

Arguments: "inputfilename.txt" // the name of the input file containing the hashtags.

Throws: IOException

This file contains the main function which takes the input file name as an argument to populate the data structure.

Functions:

    1) **Static void pop()**

Return type: function is static and does not return anything.

Arguments: FiboHeap H, Int Count, BufferedWriter bw

Throws: IOException

Description: This function removes the top Count nodes from the data structure and writes them to output_file.txt

HashTag.java:

This class is a child of the class FiboNode.java documented below.

Variables:

    1) **String HashTagName:** The name of the hashtag to be stored in the data structure.

Functions:

    1) **public boolean equals(Object obj) :**

compares 2 hashtags to check if they are equal in the name and count and other pointers present in the hashtag. This function overrides java.lang.object's inbuilt .equals() function.

Arguments: Object Obj

Returns: Boolean true of the compared object(hashtag) and the called object (hashtag) are equal in value, otherwise returns false.

FiboNode.java:

The class FiboNode mainly describes the structure of a node

that is used in a Fibonacci heap.

Variables:

Int Degree: This variable is of type Integer and it signifies the number of nodes that a node can have in its next level.

Boolean childCut: This variable is of type Boolean and it signifies whether a child has already been removed from that node. A childCut of false means that no child has ever been removed from that node.

FiboNode child, leftSibling, rightSibling, parent: pointers to the left, right, parent and child Fibonacci nodes of this Fibonacci node.

Functions:

This class has getter and setter functions for all public variables.

Other functions in this class are:

    1) public boolean equals(Object obj) :

compares 2 FiboNodes to check if they are equal in the name and count and other pointers present in the hashtag. This function overrides java.lang.object's inbuilt .equals() function.

Arguments: Object Obj

Returns: Boolean true of the compared object(FiboNode) and the called object (FiboNode) are equal in value, otherwise returns false.

FiboHeap.java:

Variables:

FiboNode max: The max node of the Fibonacci tree. This node gets written to the output file when the removeMax() is called.

Functions:

    1) FiboHeap IncreaseKey():

Arguments: FiboHeap Tree, FiboNode child, integer increaseValue

Increases the key of any node and calls cascading cut if the key of the node exceeds the key of the parent.

    2) FiboHeap cut()

Arguments: FiboHeap Tree, FiboNode child, FiboNode parent

Cuts a Fibonacci node from its parent. Resets the parents child pointer. And resets left and right pointers of the siblings of cut node.

3) FiboHeap cascadingCut():

Arguments: FiboHeap Tree, FiboNode parent

Performs cascading cut on the passed parent node. Calls cascading cut on the grandparent node.

4) FiboHeap removeMax()

Arguments: FiboHeap H

Removes the max node of the passed Fibonacci tree and reassigns a new max node.

5) FiboHeap insert()

Arguments: FiboHeap h, Node x

Inserts the node x in the tree h and returns the tree.

6) FiboHeap pairwiseCombine()

Arguments: FiboHeap H

Performs a pairwise combine on the top-level nodes (siblings of the max node) and re inserts them as the

7) Fiboheap meld()

Arguments: FiboNode x, FiboNode y

Melds the 2 nodes and inserts the node with the lesser kay as the child of the node of the grater key.

8) removeFiboNode()

Arguments: FiboNode x

Removes the node x from the tree. Adjusts the left and right sibling pointers of its neighbors accordingly.

9) getMax()

Returns the pointer to the max node of the FiboHeap.
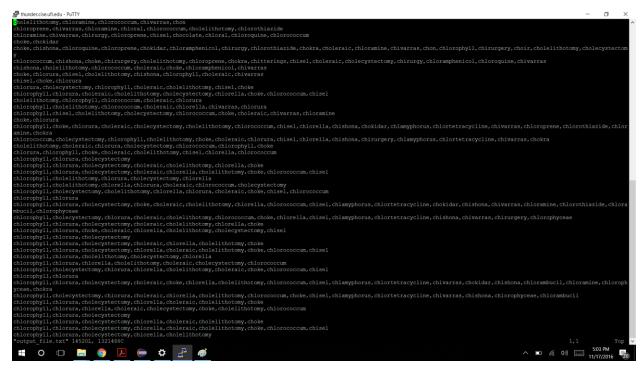
10) setMax()

Arguments: FiboNode max.

Sets the Max pointer of the given FiboHeap.

Sample Output

1) Running the project for a million inputs

```
thunder.cise.ufl.edu - PuTTY
login as: assharma
Authorized Access only

UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED.
You must have explicit permission to access this
device. All activities performed on this device
are logged. Any violations of access policy can
result in disciplinary action.

assharma@thunder.cise.ufl.edu's password:
Last login: Thu Nov 17 16:57:31 2016 from ip70-171-46-131.ga.at.cox.net
thunderx:1% ls
Desktop/  Documents/  Downloads/  Maildir@  Music/  Pictures/  Public/  Sharma_Akshay.zip  Templates/  Videos/
thunderx:2% unzip Sharma_Akshay.zip
Archive:  Sharma_Akshay.zip
  inflating: FiboHeap.java
  inflating: FiboNode.java
  inflating: HashTag.java
  inflating: hashtagcounter.java
  inflating: makefile
  inflating: project_report.docx
  inflating: sampleInput_Million.txt
thunderx:3% make
javac hashtagcounter.java
Error occurred during initialization of VM
java.lang.OutOfMemoryError: unable to create new native thread
make: *** [hashtagcounter] Error 1
thunderx:4% java -version
Error occurred during initialization of VM
java.lang.OutOfMemoryError: unable to create new native thread
thunderx:5% java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)
thunderx:6% make
javac hashtagcounter.java
Note: ./FiboHeap.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
javac FiboHeap.java
Note: FiboHeap.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
javac HashTag.java
javac FiboNode.java
thunderx:7% java hash
hashtagcounter.class  hashtagcounter.java
thunderx:7% java hashtagcounter.java hashtagc
hashtagcounter.class  hashtagcounter.java
thunderx:7% java hashtagcounter sampleInput_Million.txt
thunderx:8% vi output_file.txt
```

2) Output_file.txt

Conclusion

The project has been successfully implemented and tested to output n most popular hashtags from inputs containing up to a million entries. The program uses Max Fibonacci heap to create the output within acceptable time limits.