Advanced Data Structures (COP5536)

SPRING 2020

Programming Project Report

SAI DATTA VARA PRASAD VASIREDDY

UFID: 5529-6647

Vasireddys1@ufl.edu

PROJECT DESCRIPTION

This project aim is to develop a program to identify the n most popular hashtags on social media sites like Facebook or Twitter.

The hashtags for the purpose of this project are taken from an input script.

The basic concept for implementation is to use a max priority framework to figure out which hashtags are most common.

The project uses the following structures for implementation.

- Max Fibonacci heap: used to display hashtag frequencies.
- HashMap: The key for the HashMap is hashtag, and the value is the pointer to the corresponding node in the Fibonacci heap.

The code is released in JAVA. I implemented Max Fibonacci Heap in Java and stored all node addresses in a HashMap (Built in Data Structure) file. Max Fibonacci Heap is needed for increased key operation because it has better theoretical limits.

WORKING ENVIRONMENT

HARDWARE REQUIREMENT:

Hard Disk space: 4 GB

minimum Memory: 512 MB

CPU: x86

OPERATING SYSTEM:

LINUX/UNIX/MAC OS (If using other OS make command might not work)

COMPILER:

Javac

COMPILING & RUNNING INSTRUCTIONS:

This project was built on thunder.cise.ufl.edu and the local java compiler and checked on it. To run the application, one will use SSH link to access the file.

Following are the detailed steps for accessing the server and executing the program –

- Remotely access thunder.cise.ufl.edu using username@thunder.cise.ufl.edu and key in your password.
- Extract the contents of the hashtagcounter.zip file.
- Type ‘make’ without typing quotes.
- Type ‘java hashtagcounter <input_file_path> <output_file_path.txt>’. input_file_path is path to the input file while output_file_path is where we want the output of the program to be written to.

STRUCTURE OF THE PROGRAM AND FUNCTION DESCRIPTIONS

The program has a hashtagcounter file inside which there exists these following 3 classes.

1) hashtagcounter - The main class that reads the input and writes the output.

2) Node - This class is used to instantiate and object of node in memory.

3) Fibonoic_Heap – This class is used for the methods and functions of the Fibonacci Heap class

The basic workflow of the program is as follows:

1. The hashtagcounter class takes the data from an input file.
2. Based on the format defined in the input description file, it determines whether to insert nodes into the heap or delete the 'N' elements at the top of the heap.
3. When the top 'N' elements are retrieved, they will be written to a file according to the system parameters.
4. Whenever it encounters a "stop" string, it halts immediately.

The detailed view of class functions is given below –

hashtagcounter.class –

hashtagcounter has following class variables –

Filewriter printer – for writing the output to either the given output file or a standard output.
F_heap – object of Fibonoic_Heap which is used to access its class variables and methods.
map – a hashmap that maps each hashtag to the corresponding node in the Fibonacci heap.
BufferedReader b_read – for reading the input line by line from the given input file.
S_arr[] – an array used for storing current input lines as tokens. From the format of given input file, if the length of s_arr array is more than 2 then it is assumed that first token is hashtag and the second one is frequency of that hashtag. Otherwise, we will check if it is a count or look for "stop". If it is count, then

we perform remove_Max operation “num” times. Otherwise, we simply halt the program.

NODE CLASS:

Node class has following class variables –
deg – Number of children a node has in its immediate next level. Degree holds Integer value.
hashTag – contains the hashtag that the node object currently holds.
Child_Cutvalue– if the Child_Cutvalue is false, it means no child has been removed from it at that point.
key – frequency of the hashtag.
next– a pointer to the next node in the circular list.
prev – a pointer to the previous node in the circular list.
parent – a pointer to the parent node.
child – a pointer to the child node.

Node class has following functions:

Function	Return Type	Parameters
Node(int element ,String keyValue)	-	element, keyValue

Function	Description
Node(int element ,String keyValue)	Constructor that initializes a node with given hashtag and key.

Fibonoic_Heap.class –

Function	**Return Type**	**Parameters**
insert(Node node)	Node	Node node
removeMax()	Node	-
increaseKey(Node a, int val)	Void	Node heapNode, int addVal
Meld(Node,Node)	Node	Node tree1,Node tree2
childCut(Node)	Void	Node heap_Node
remove_nMax(int)	Void	int(numb)
m_ax()	Node	-

Function	**Description**
insert(Node node)	Insert() inserts a new node into the heap. Current implementation of this function add the element next to the maxNode
removeMax()	removeMax function removes the maxNode from the list and fills the gap by adjusting the next and the previous pointers to it. Merge By Degree follows removeMax.
increaseKey(Node a, int val)	increaseKey increases the frequency of a hashtag by certain value. Child_Cut function follows increaseKey. They check if the node value is greater than its parent node. If so, they cut the element and add it to the root list

Meld(Node,Node)	Merges two doubly linked lists in 0(1) time
childCut(Node)	Recursively cuts the marked parents of a node.
remove_nMax(int)	Removes the first N max elements from the fibnoacci heap.
m_ax()	Returns the maximum value of the heap.

PROGRAM EXECUTION

```
thunder:11% ls
Fibonaic_Heap.java  hashtagcounter.java  Maildir@  Makefile  Node.class  Node.java  sampleInput.txt
thunder:12% cp Fibonaic_Heap.java Fibonoic_Heap.java
thunder:13% ls
Fibonaic_Heap.java  Fibonoic_Heap.java  hashtagcounter.java  Maildir@  Makefile  Node.class  Node.java  sampleInput.txt
thunder:14% rm Fibo
Fibonaic_Heap.java  Fibonoic_Heap.java
thunder:14% rm Fibonaic_Heap.java
thunder:15% ls
Fibonoic_Heap.java  hashtagcounter.java  Maildir@  Makefile  Node.class  Node.java  sampleInput.txt
thunder:16% cat M
Maildir@  Makefile
thunder:16% cat Makefile

all:
        javac Node.java
        javac Fibonoic_Heap.java
        javac hashtagcounter.java

thunder:17% make
javac Node.java
javac Fibonoic_Heap.java
Note: Fibonoic_Heap.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
javac hashtagcounter.java
thunder:18% java hashtagcounter sampleInput.txt
thunder:19% ls
Fibonoic_Heap.class  hashtagcounter.class  Maildir@  Node.class  output_file.txt
Fibonoic_Heap.java   hashtagcounter.java   Makefile  Node.java   sampleInput.txt
thunder:20% cat output_file.txt
choirmaster,chokefull,chlamys,chlorination,chirr,chirurgy,chloramphenicol,chloramine
chokefull,chlorococcales,chlorophthalmidae,chitterings,chlorination,chokra,cholinergic,chiseling,chloramphenicol
chirr,chlamys,chokefull,chlorophyta,chiseling,chirurgy,chlorophyllose,chloramphenicol,chlorophthalmidae,chlorococcales
chokefull,chiseling,chloramphenicol,chirr,chitterings,chirurgy,chisel
chiseling,chloramphenicol,chloroform,chokefull,chlamys,chlorococcales,cholinergic,choline,chloranthaceae,chisel,chirr,chirurgy,chitterings,chlorophthalmidae
chloramphenicol,chiseling,chisel,chlorination,chirr
chiseling,chloramphenicol,chlorination,chloranthaceae,chlorococcales,chisel,choleric,chloroform,choeronycteris,chirr,chirurgical,chlorosis,chlortetracycline,chlorophyta
chiseling,chloramphenicol,choeronycteris,chokra,chlorococcales,chlorination,chloranthaceae,choleric
choeronycteris
chloramphenicol,choice,chitinous,choeronycteris
chondrosarcoma,chitinous,choice,choeronycteris,chloramphenicol,chiseling,chokra,choirmaster,chlorination,chlorosis
thunder:21%
```

RESULTS

This code was tested on the provided sampleInput.txt file. The output with the most popular hashtags was generated and stored in the specified file. Output might be different from the actual output because ties were broken arbitrarily. Also, it entirely depends on the implementation of the Fibonoic_Heap.

CONCLUSION

With the proper implementation of Fibonoic_Heap and successful execution of the program on the given sample_Input file, the objective of the assignment has been met. Furthermore, we can improve the efficiency by adjusting the degreeTable size and Faster IO, but that is beyond the scope of the project.