Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

ADS Project Report

Introduction

We had to implement a maximum Fibonacci Heap where we count the hashtags of a given input file. So for this, I had four classes. First class was hashtagcounter which reads the file with the hashtags. Second class was of the main FibonacciHeap which has all the basic functionalities of the maximum Fibonacci heap. Third class is of the FibonacciNode which stores all the attributes of the node in the Fibonacci heap. Fourth class is the Hashtag class which stores the attributes of the Hashtag. This system has been implemented to find out the most popular hashtags which appears on social media websites.

Machine Details

1) Processor
- 2.7GHz dual-core Intel Core i5 processor with Turbo Boost up to 3.1GHz
  3MB shared L3 cache

2) Memory
- 8GB of 1866MHz LPDDR3 (RAM)
  256GB PCIe-based flash storage (Internal Memory)

3) Graphics
- Intel Iris Graphics 6100

Compiler Used

- Java Compiler version : 1.8.0_101
- Java Runtime Environment Version : 1.8.0_101-b13

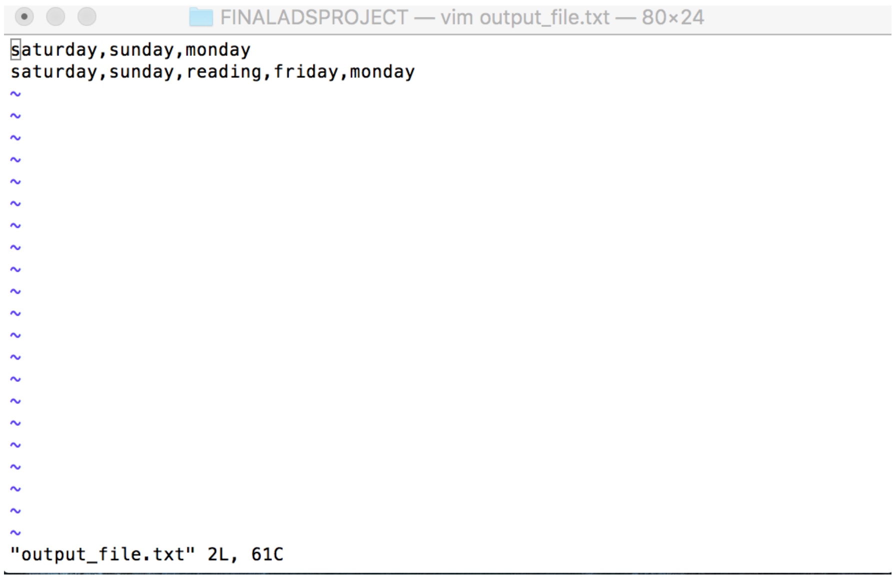
Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

Compiling and Running

I ran my code on the thunder server of the UFL website as well. Unzipping the file should be done first and then the following commands should be run.

```
make
java hashtagcounter inputfile.txt
```

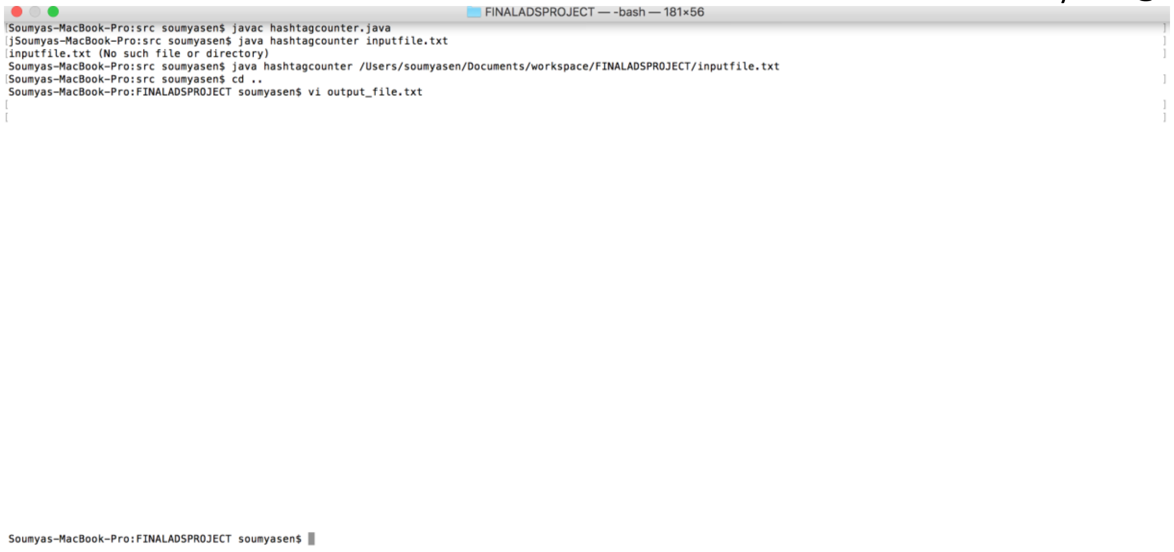
Results



```
saturday,sunday,monday
saturday,sunday,reading,friday,monday
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"output_file.txt" 2L, 61C
```

This was the result (i.e. output file) when the input file was the one given in the PDF.

Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

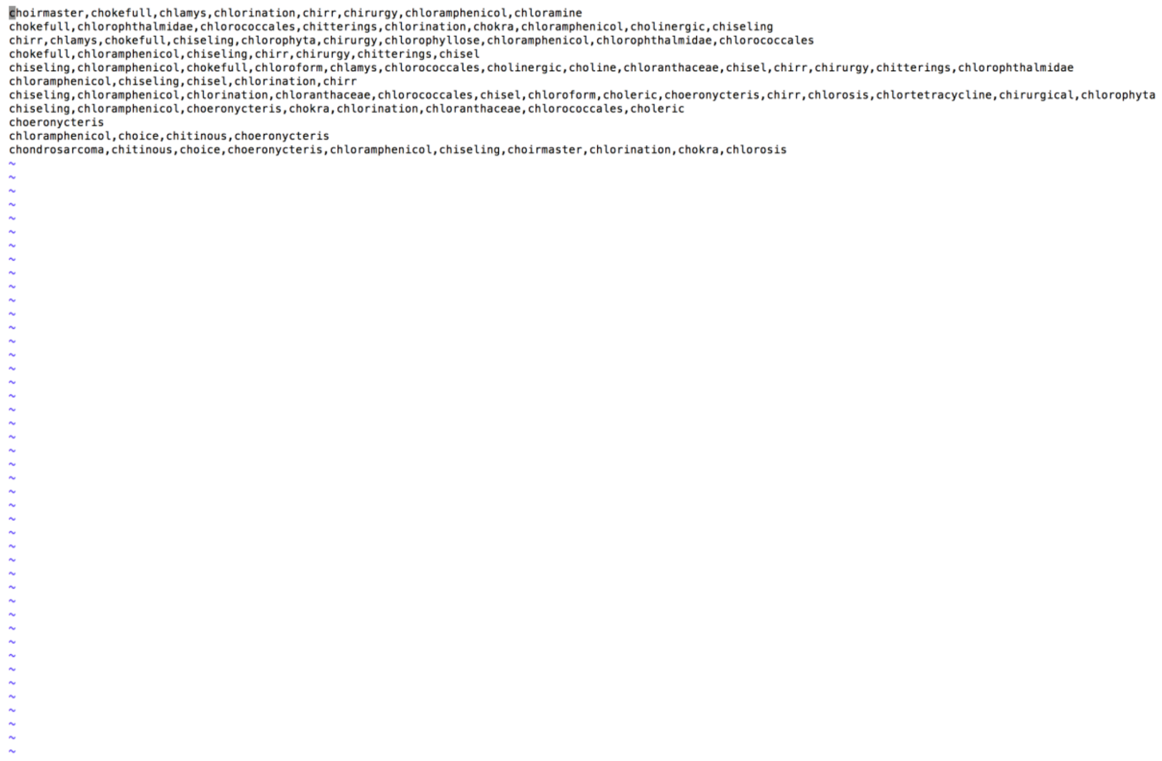


```
FINALADSPROJECT — -bash — 181×56
Soumyas-MacBook-Pro:src soumyasen$ javac hashtagcounter.java
jSoumyas-MacBook-Pro:src soumyasen$ java hashtagcounter inputfile.txt
inputfile.txt (No such file or directory)
Soumyas-MacBook-Pro:src soumyasen$ java hashtagcounter /Users/soumyasen/Documents/workspace/FINALADSPROJECT/inputfile.txt
Soumyas-MacBook-Pro:src soumyasen$ cd ..
Soumyas-MacBook-Pro:FINALADSPROJECT soumyasen$ vi output_file.txt



Soumyas-MacBook-Pro:FINALADSPROJECT soumyasen$ 
```

The second input file which was provided for reference was the input file. This was run on my terminal.



```
choirmaster,chokefull,chlamys,chlorination,chirr,chirurgy,chloramphenicol,chloramine
chokefull,chlorophthalmidae,chlorococcales,chitterings,chlorination,chokra,chloramphenicol,cholinergic,chiseling
chirr,chlamys,chokefull,chiseling,chlorophyta,chirurgy,chlorophyllose,chloramphenicol,chlorophthalmidae,chlorococcales
chokefull,chloramphenicol,chiseling,chirr,chirurgy,chitterings,chisel
chiseling,chloramphenicol,chokefull,chloroform,chlamys,chlorococcales,cholinergic,choline,chloranthaceae,chisel,chirr,chirurgy,chitterings,chlorophthalmidae
chloramphenicol,chiseling,chisel,chlorination,chirr
chiseling,chloramphenicol,chlorination,chloranthaceae,chlorococcales,chisel,chloroform,choleric,choeronycteris,chirr,chlorosis,chlortetracycline,chirurgical,chlorophyta
chiseling,chloramphenicol,choeronycteris,chokra,chlorination,chloranthaceae,chlorococcales,choleric
choeronycteris
chloramphenicol,choice,chitinous,choeronycteris
chondrosarcoma,chitinous,choice,choeronycteris,chloramphenicol,chiseling,choirmaster,chlorination,chokra,chlorosis
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

This is the output file created for the above input file.

Structure of the Program

1) **Class FibonacciNode** : Contains the definitions and constructors to create a node for the Fibonacci Tree.

Instance variables :

i) Node leftNode: stores the left sibling of the node.
ii) Node rightNode: stores right sibling of the node.
iii) Node parent : stores the parent of the node.
iv) int degree : stores the degree of the node.
v) boolean childCut : It can be either True or False for every node.
vi) int data: Contains the data of the node

This class has all the getters and setter methods of all the attributes of the Fibonacci Node.

It has a function of **public Boolean equals(Object o)** which takes an object as a parameter and it checks if the name and count of the node is present in the hashtag or not. If it's the same then it overrides the equal() method.

This class also has a function of **public int hashCode()** with no argument which maps the memory address to an integer value. This function can overrides the original allocation of memory for the nodes.

2) **Class FibonacciHeap**: Contains the definitions, functions and constructors to create a maximum Fibonacci Heap.

Instance Variables :

i) FibonacciNode max: stores the maximum node of the Fibonacci tree

Functions:

1) It has a getter and a setter method for maximum node of the Fibonacci Heap.

Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

2) public FibonacciHeap insert(FibonacciHeap H, FibonacciNode x): This is used to insert a new node to the heap.

3) public FibonacciHeap increaseKey(FibonacciHeap H, FibonacciNode child, int increaseValue): This function basically increases the value of the node. It even calls the cascading cut function if it exceeds the value of the parent.

4) public FibonacciHeap cut(FibonacciHeap H, FibonacciNode child, FibonacciNode parent): It cuts the node from the parent. After it's cut, the pointers are reset.

5) public FibonacciHeap cascadingCut(FibonacciHeap H,FibonacciNode parent): It cuts the nodes of the parents as well if their childCut value is true from the heap.

6) public FibonacciHeap removeMax(FibonacciHeap H): Removes the maximum node of the tree and temporarily assigns the value of the max node.

7) public FibonacciHeap meld(FibonacciNode x, FibonacciNode y): Meld function is done after removeMax() function. This functions melds the two nodes.

8) public FibonacciHeap removeFibonacciNode(T FibonacciNode): It just removes the Fibonacci node whenever it's required.

9) public FibonacciHeap pairwiseCombine(FibonacciHeap H) After removemax() it does a pairwise combine of the nodes to form a tree structure again.

3) **Class HashTag**: This class contains all the attributes of the hashtags.

Instance Variables :

String title: The name of the hashtag to be stored in the Fibonacci heap structure.

Functions:

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.

4) **Class hashtagcounter**: This class takes a file as an input and results in an output file. It contains the main function which calls all the functions of the Fibonacci heap.

Soumya Sen
UFID – 19217737
soumyasen@ufl.edu

Functions:

1) public static void main(String[] args) : Main program flow starts from here. It reads in the file and makes the calles to initailize the heap. Exceptions are dealt while during handling of the file.

2) public static FibonacciHeap optags(FibonacciHeap H, int topHashtags, BufferedWriter bw): This function removes the top counts from the Fibonacci heapand writes them to the output file.

Conclusion

I have successfully implemented and tested the Fibonacci heap which calculates the most popular hashtags from the input file.