

# **Advanced Data Structures (COP5536)**

**Fall 2018**

## **Programming Project Report**

Name : Niraja Ganpule

UFID : 17451951

Email : niraja1101@ufl.edu

## PROJECT DESCRIPTION

This project requires to find the  $n$  most popular keywords for a search engine Duck Duck Go, associated with frequencies appearing in the provided input file. This can be achieved by implementing a max priority structure i.e Max-Fibonacci Heap using primitive data structures such as pointers. For storing and retrieval of keywords and their associated nodes, we use a HashMap as it allows amortized complexity of  $O(1)$  for put and get operations.

# STRUCTURE OF THE PROGRAM AND METHOD PROTOTYPES

The project has been modularized into four classes namely *Node*, *KeywordMap*, *MaxFibonacciHeap* and *keywordcounter* which are detailed below.

## Node Class

The class Node defines the structure of the node in Max Fibonacci Heap.

### Variables

key

It is an integer variable which represents the frequency of the tag.

degree

It is an integer variable which represents the number of children of the node.

tag

It is a string variable which represents the label or tag stored in the node

parent

It is a Node which represents the parent of the node.

child

It is a Node which represents the child of the node.

previous

It is a Node which represents the node before the current node in the circular doubly linked list of roots of Max-Fibonacci heap.

nextptr

It is a Node which represents the node next to the current node in the circular doubly linked list of roots of Max-Fibonacci heap.

childcut

It is a boolean variable which is true if a child has been removed from the node and false otherwise.

### Methods

Method : Node(constructor)

Parameters : int key & String tag

Description : Node class constructor which initializes the class variables with the key and tag

Return value: None

## **KeywordMap class**

This class defines the Hashmap that contains the tags and the corresponding tag node.

### **Variables**

Map<String, Node> map

It is a HashMap object which stores (String tag, Tag Node) pairs.

### **Methods**

Method: KeywordMap(constructor)

Parameters: None

Description: KeywordMap class constructor which initializes the class variables

Return value: None

Method: putmap

Parameters: String tag, Node tagnode

Description: Inserts the passed (tag,tagnode) entry in Hashmap.

Return value: None

Method: getmap

Parameters: String tag

Description: Retrieves tagnode associated with the tag entry from Hashmap.

Return value: tagnode

## **MaxFibonacciHeap class**

The MaxFibonacciHeap class defines the operations performed on the Node and Max-Fibonacci heap.

### **Variables**

maxnode

It is a Node which represents the Max-Fibonacci heap node with maximum key value.

sizeofheap

It is an integer variable which represents the size of the Max-Fibonacci heap.

### **Methods**

Method : MaxFibonacciHeap(constructor)

Parameters : None

Description. : MaxFibonacciHeap class constructor which initializes the class variables

Return value : None

Method : returnmax

Parameters : Node first and Node second

Description : Returns the node with the bigger key among first and second

Return value: Node with larger key

Method: findmax

Parameters: None

Description: Returns the maxnode of MaxFibonacciHeap.

Return value: The maxnode of MaxFibonacciHeap.

Method: insert

Parameters: String tag & int key

Description: Inserts new node with passed tag and frequency key values and melds it with existing list of roots.

Return value: The newly created node.

Method: removeMax

Parameters: None

Description: Remove the maxnod, melds its children with existing list of roots.

Follow this operation by pairwise combine of trees with equal degree.

Return value: The removed maxnode

Method: increaseKey

Parameters: Node node & int newkey

Description: Increases the key value of the passed node to value of newkey . If this disturbs the maxheap property, cut the node from the parent and follow up with a cascadingcut operation on heap.

Return value: None

Throws: New key is smaller than old key if  $\text{newkey} < \text{node.key}$ .

Method: meld

Parameters: Node first and Node second

Description: Melds the first and second nodes into the MaxFibonacciHeap.

Return value: The maximum node of the two passed nodes.

Method: cascadingcut

Parameters: None

Description: cascadingcut operation performed on MaxFibonacciHeap beginning from parent of node cut away til the root where if childcut value of node is true it is cut away from its parent else its childcut value is changed to true.

Return value: None

Method: cutnode

Parameters: Node node & Node parent

Description: Cut the passed node from its parent and meld it with existing list of roots

Return value: None

Method: pairwisecombine

Parameters: None

Description: Combines trees of equal degree considering two at a time and updates the maxnode at the end by examining the trees from the degreetable.

Return value: None

Method: reinsertmaxnodes

Parameters: List<Node> list

Description: Reinserts the nodes in the passed list into the existing list of roots of MaxFibonacciHeap.

Return value: None

## **keywordcounter Class**

The keywordcounter class includes the main method and defines the methods related to the input and output text files processing.

### **Variables**

keymap

It represents a KeywordMap class object to store (tag,tag-node) pairs in HashMap.

maxfibheap

It represents a MaxFibonacciHeap class object to create and operate on the tag Nodes.

### **Methods**

Method: keywordcounter(constructor)

Parameters: None

Description: keywordcounter class constructor which initializes the class variables keymap and maxfibheap

Return value: None

Method: main

Parameters : Commandline argument input file

Description: Creates object of keywordcounter class , objects of BufferedReader, BufferedWriter are used for text file processing. It invokes its readf method to process the inputFile passed in the command line. The output file is written to using the writef method

Return value: None

Method: writef

Parameters: BufferedWriter writer, Node node and String newLine

Description: Writes the tag associated with the passed tagnode node in the Output file associated to BufferedWriter writer.

Return value: None

Method: readf

Parameters: BufferedReader reader and BufferedWriter writer

Description: Reads and processes the Input file passed associated with BufferedReader reader.

Return value: None

## PROGRAM EXECUTION AND SAMPLE OUTPUT

To compile:

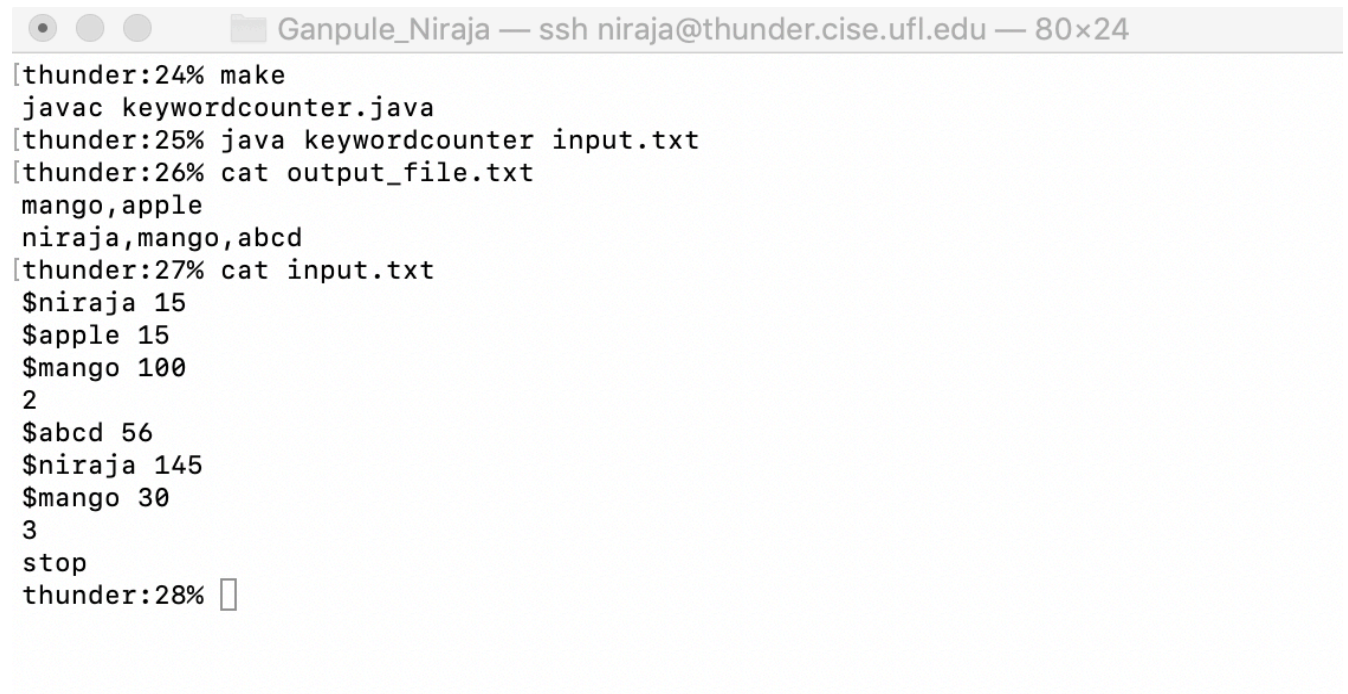
In the folder containing code, on the terminal run the command:  
make

To run:

In the same folder on the terminal, run the command:  
java keywordcounter input.txt

Output:

The output will be generated as a text file output\_file.txt



A terminal window titled "Ganpule\_Niraja — ssh niraja@thunder.cise.ufl.edu — 80x24" displays the following commands and output:

```
[thunder:24% make
javac keywordcounter.java
[thunder:25% java keywordcounter input.txt
[thunder:26% cat output_file.txt
mango,apple
niraja,mango,abcd
[thunder:27% cat input.txt
$niraja 15
$apples 15
$mango 100
2
$abcd 56
$niraja 145
$mango 30
3
stop
thunder:28% ]
```



## **CONCLUSION**

The project to implement a system to count the most popular keywords used in their search engine new search engine “DuckDuckGo” has been successfully completed using Max Fibonacci heap and HashMap with very good running times even for Million input sample file.