COP 5536 Advanced Data Structures – Spring 2020
Programming Project

Suprith Reddy Gurudu
UFID: 9961-2134 | sgurudu@ufl.edu

PROBLEM DESCRIPTION:

Many social media websites such as Facebook or Twitter contains hashtags created by the users. They are planning to show the most popular hashtags using a software system.
A hashtag has the following field to display:

- *hashTagName* – unique name associated with a hashtag.
- *hashTagEntry* – unique entry number associated with a hashtag.

The problem description is to implement a system to find the n most popular hashtags. For the scope of this project hashtags will be given from an input file. Basic idea for the implementation is to use a max priority queue and a hash table.
The needed operations are:

- printHashTags( ) – writes/prints 'n' most popular hashtags from given input file to the output file/stream.

Increase key operation must be performed many times on large number of hashtags appearing in the input stream. The Max Fibonacci Heap is used as a max priority queue structure because it has an amortized complexity *O (1)* for the increase key operation. For the hash table, a Hashtable class from Java Utilities(java.util) library is used to store hashtag data. The tie for the same frequency hashtags is broken by using *hashTagEntry* [TIE BREAKER] (unique number) i.e., *hashTagEntry* with lower values will be picked first.

IMPLEMENTATION:

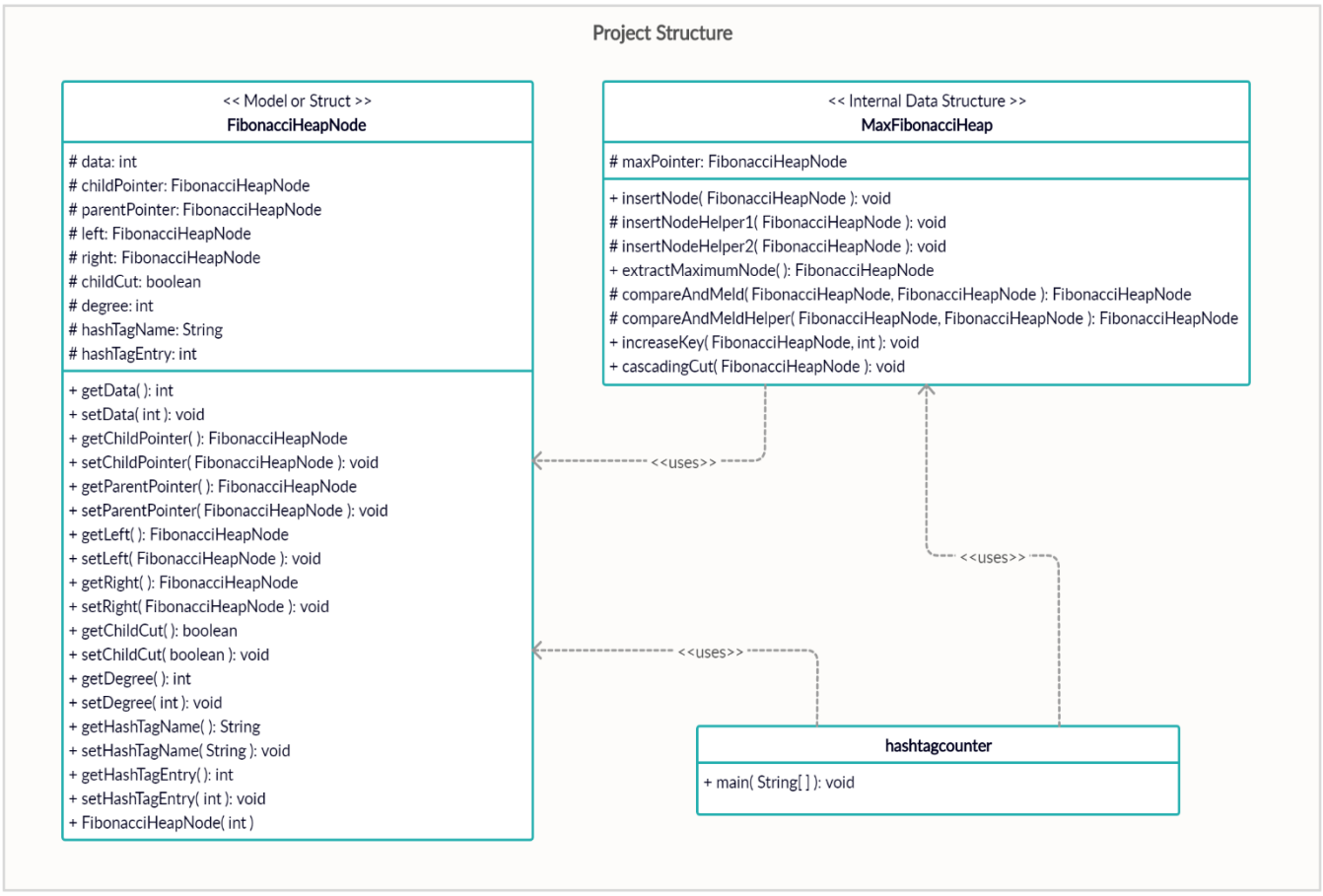
- **Max Fibonacci heap:**
  Fibonacci Heap is a collection of trees with min-heap or max-heap property. In Fibonacci Heap, trees can have any shape even all trees can be single nodes. It is used to keep track of the frequencies of hashtags.
  Amortized complexity is *O (1)* for *increaseKey*, *insertNode*, *compareAndMeld* and *cascadingCut* operations and *O (lg n)* for *extractMaximumNode* operation.
- **Hash table:**
  A hash table is a data structure that is used to store keys/value pairs. It uses a hash function to compute an index into an array in which an element will be inserted or searched. The key for the hash table is the hashtag name, and the value is the pointer to the corresponding node in the Max Fibonacci heap.
  Time complexity is *O (1)* for *insert* and *search* operations.

EXECUTION:

Steps to execute the project:

- Unzip the project folder titled *'Gurudu_SuprithReddy.zip'*.
- Paste project folder to the current directory.
- Type $*make* on thunder server terminal to compile the source code.
- On successful compilation, type $*java hashtagcounter <input_file_name > [output_file_name]*
- On successful execution, view the output in output_file_name if output file is given in the arguments else on the terminal/stdout.

PROJECT STRUCTURE:



Suprith Reddy Gurudu
UFID: 9961-2134

- **FibonacciHeapNode:**
  This class creates a structure for the node which holds the data. Other two classes use this class to create objects/instances for the node by calling the respective getter/setter methods.

- **MaxFibonacciHeap:**
  This class implements a max-fibonacci-heap. It includes the operations such as insert, delete-max and increase-key. A global variable *maxPointer* is maintained to point to the maximum key node and a hashmap is used to store the degrees of the nodes.

- **hashtagcounter:**
  This is where the execution starts. This class has a *main* method takes the input file from the command line arguments store input strings as hashtags in a hashtable, processes it by calling specific methods in the **MaxFibonacciHeap** class and outputs the file or stream which basically gets executed until all the strings are scanned or string "stop" is encountered.

PROJECT DOCUMENTATION:

1. **Class Name – FibonacciHeapNode**

➢ MEMBER VARIABLES:

Member Name	Data Type	Purpose
data	int	Stores the frequency value
childPointer	FibonacciHeapNode	Pointer to the child of a node
parentPointer	FibonacciHeapNode	Pointer to the parent of a node
left	FibonacciHeapNode	Pointer to the left node
right	FibonacciHeapNode	Pointer to the right node
childCut	boolean	Previous flag for any child cut
degree	int	Degree of a node
hashTagName	String	Name of a hashtag
hashTagEntry	int	Entry number of a hashtag

Suprith Reddy Gurudu
UFID: 9961-2134

- METHODS:

Method Name	Return Type	Purpose
getData()	int	Gets the frequency data value
setData(int)	void	Sets the frequency data value
getChildPointer()	FibonacciHeapNode	Gets the child pointer node
setChildPointer(FibonacciHeapNode)	void	Sets the child pointer
getParentPointer()	FibonacciHeapNode	Gets the parent pointer node
setParentPointer(FibonacciHeapNode)	void	Sets the parent pointer
getLeft()	FibonacciHeapNode	Gets the left pointer node
setLeft(FibonacciHeapNode)	void	Sets the left pointer
getRight()	FibonacciHeapNode	Gets the right pointer node
setRight(FibonacciHeapNode)	void	Sets the right pointer
getChildCut()	boolean	Gets the child cut value
setChildCut(boolean)	void	Sets the child cut value
getDegree()	int	Gets the degree of the node
setDegree(int)	void	Sets the degree of the node
getHashTagName()	String	Gets the hashtag name
setHashTagName(String)	void	Sets the hashtag name
getHashTagEntry()	int	Gets the hashtag entry value
setHashTagEntry(int)	void	Sets the hashtag entry value
FibonacciHeapNode(int)	-	Parameterized Constructor

2. Class Name – MaxFibonacciHeap

- MEMBER VARIABLES:

Member Name	Data Type	Purpose
maxPointer	FibonacciHeapNode	Pointer to max-frequency node

Suprith Reddy Gurudu
UFID: 9961-2134

- METHODS:

Method Name	Return Type	Purpose
insertNode(FibonacciHeapNode)	void	Inserts the node into the heap
insertNodeHelper1(FibonacciHeapNode)	void	Helps the insertNode method
insertNodeHelper2(FibonacciHeapNode)	void	Helps the insertNode method
extractMaximumNode()	FibonacciHeapNode	Deletes the maximum node
**compareAndMeld(f1, f2)*	FibonacciHeapNode	Compares and merges nodes
**compareAndMeld(f1, f2)*	FibonacciHeapNode	Helps the compareAndMeld
increaseKey(FibonacciHeapNode, int)	void	Increases node's key value
cascadingCut(FibonacciHeapNode)	void	Cascades the cut in hierarchy

* f1, f2 => FibonacciHeapNode, FibonacciHeapNode

3. **Class Name – hashtagcounter**

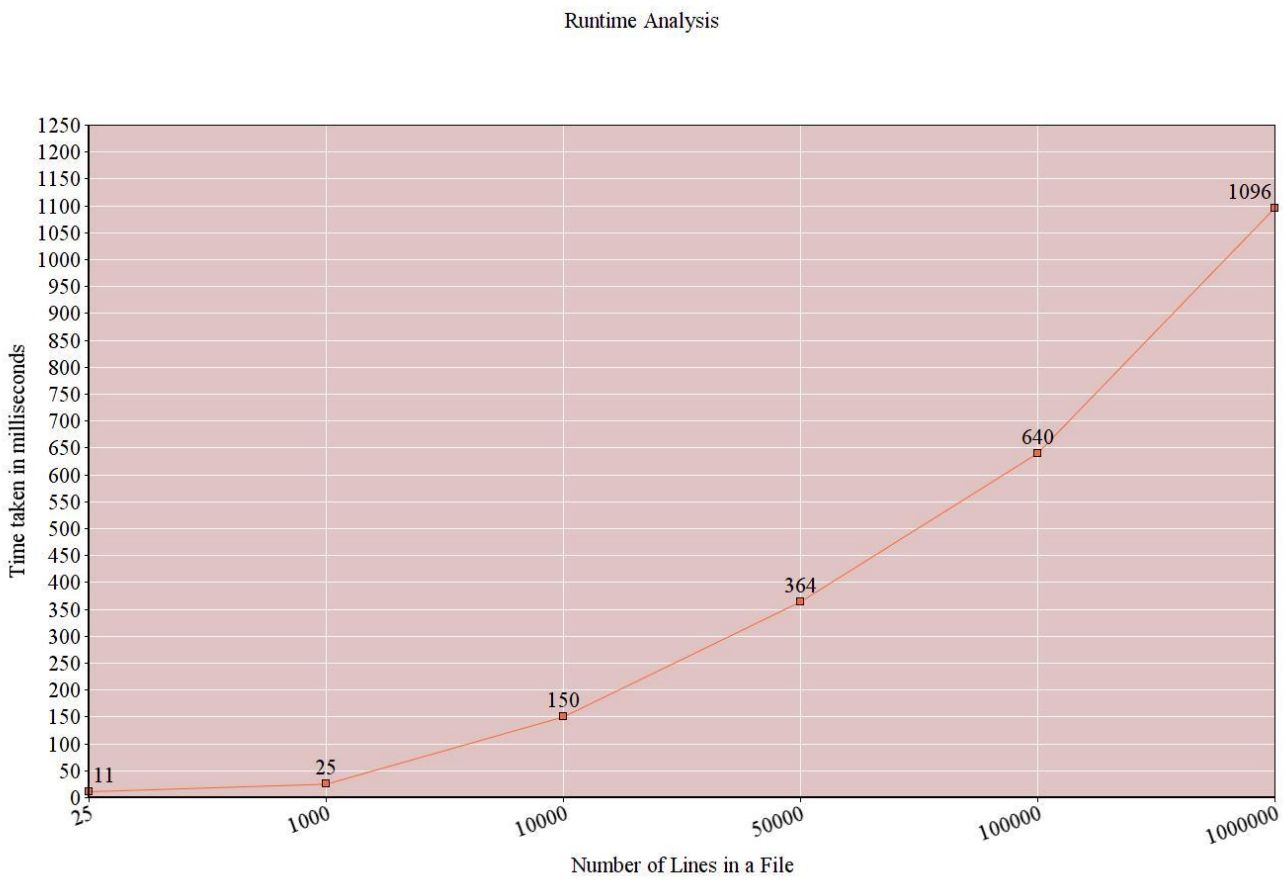
- METHODS:

Method Name	Return Type	Purpose
main(String[])	void	Start point of the program

RUNTIME ANALYSIS:

The amortized times for the operations of Fibonacci Heap is *O (lg n)* [delete-max] and *O (1)* [other operations]. Therefore, after running multiple files with different number of lines of input, we get a curve like 'm' times logarithmic graph i.e., *O (m lg n)* where 'm' is number of times the delete-max operation is executed.

Suprith Reddy Gurudu
UFID: 9961-2134



REFERENCES:

[1] Introduction to Algorithms, by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein

[2] Core Java: an Integrated Approach, by Dr. R. Nageshwara Rao

[2] GeeksforGeeks - https://www.geeksforgeeks.org/fibonacci-heap-set-1-introduction/ , https://www.geeksforgeeks.org/fibonacci-heap-insertion-and-union/?ref=rp and https://www.geeksforgeeks.org/fibonacci-heap-deletion-extract-min-and-decrease-key/?ref=lbp

Suprith Reddy Gurudu
UFID: 9961-2134