UFID: 01794985
Name: Jayetri Bardhan
Email id: jayetri.bardhan@ufl.edu

Project Report:

My project creates a hashtag counter. It makes use of a hashtable where key of the hashtable contains the hashtag and the value points to the corresponding node in the Fibonacci max heap. I have used Java to implement my code and Intellij IDE.

I have implemented the following classes in my project:

1) hashtagcounter: This is the main class which has the main function. I have explained its logic in the later section.
2) fibMaxNode: This class is again used in the class fibMaxHeap. It is used to represent each of the nodes in the Fibonacci max heap. It contains a constructor where all the node attributes have been initialized.
3) fibMaxHeap : This class represents the Fibonacci Max Heap. It also has a constructor to initialize all its attributes along with various other methods the Max Fibonacci heap like- insert, remove max, increase key, cascading cut etc.

Procedure to compile and run the file:

- First unzip the folder. The zip folder contains the following files: ADS_Project_report, a folder called HashtagCounter containing the folder named src containing Makefile, hashtagcounter.java, fibMaxNode.java. fibMaxHeap.java
- So use command cd HashtagCounter/src
- Use the command ‘make’. The command will generate the following classes: hashtagcounter.class, fibMaxNode.class and ibMaxHeap.class.
- Then use either of the following commands:

1)$java hashtagcounter <input_filename> <outputfilename>

This will generate the results in the output file. Make sure that the input file in the current directory.
2)$java hashtagcounter <input_filename>
This will generate in the output console. Make sure that the input file in the current directory.

Project Structure:

1) hashtagcounter.class

It has the main function.
Working/Explanation:

- It creates a hashtable object called: all_hashtags_diction. This contains keys having all the hashtags and values pointing to the nodes of the Fibonacci heap.
- It initially checks the command line input. If the number of command line arguments =2 , ie. in the form of : $java hashtagcounter <input_filename>
- Then it generates the result in the console.
- If 3 arguments are present, i.e. in the form of : $java hashtagcounter <input_filename> <outputfilename>:

Then the output gets written the output file specified.

- The input file is read line by line.
- The program stops whenever ‘stop’ is present.
- It splits the line using space. So the first argument will contain the hashtag and the second argument will contain the frequency.
- Whenever the line contains #, it check if the first argument (hashtag_character) is present as a key in the hashtable. If present, then it calls the FibMax_Key_Increase function present in the fibMaxHeap class and the frequency value is increased for the corresponding node. The node is accessed using the hashtable.
- If the key is not present in the hashtable- ‘all_hashtags_diction’, then it creates a new key with the hashtag and creates a value pointing to

the Fibonacci heap node. In the fibonacci heap, the FibMax_Insert function is called where a new node is inserted containing the key= hashtag and value = frequency.

- If there is no # in the line and if an integer is present, it outputs that many no. of hashtags with the highest frequency in the output file. For this case, it runs a loop, calling the function Fib_RemoveMax() from the fibMaxHeap class. This returns all the hashtags having the highest frequency. Each of the results are separated by ‘,’. During this process, remove the corresponding elements from the hashtable too.
- Once the loop ends, it reinserts all those nodes back to the fibonacci heap by calling the FibMax_Heap function and also adds those keys back to the hashtable.
- This process is repeated till the line containing the word- ‘stop’ is reached.
- Note: The code is written to stop writing the program when ‘stop’ is reached and not ‘stop’, i.e. It is assumed that capital letters are not present.

2)fibMaxNode.class

The following member variables are used:

1. Heap_Key- this refers to the key of the Fibonacci Heap. This is of type double.
2. NodeDegree- this represents the no. of children the node contains. This is of type integer.
3. flag_for_cascading- This shows if the childcut value is true or false. It is of type boolean. It is false initially. For every node, when one of its children is removed, it is marked true.
4. fibMaxNode<T> Parent_Node - It points to the parent field.
5. fibMaxNode<T> Child_Node - It points to the child field.
6. fibMaxNode<T> sibling_right_node - It points to the left sibling of this node since fibonacci heap has a doubly linked list.
7. fibMaxNode<T> sibling_left_node - It points to the right sibling of this node since fibonacci heap has a doubly linked list.

8. Value: It is used to represent the frequency of the hashtag and the value of node in the Fibonacci Heap.

   Constructor:

- **public fibMaxNode()**

  Name: **fibMaxNode**

  Argument: value, Heap_Key

  Description: It is a constructor for initializing the member variables of this class like- sibling_right_node, sibling_left_node, value and key.

3) fibMaxHeap.class

Description: THis represents the Fibonacci heap. It executes functions like insert, remove max etc.

It contains the following member variables:

1) SizeOfHeap: This refers to the number of nodes in the fibonacci heap. IT is of type integer.
2) fibMaxNode<T> Biggest_node : This refers to the node containing the maximum value.

It contains the following constructor:

1) **public** fibMaxHeap() :

   Argument: It contains no argument.

It has the following member functions:

- **public void** FibMax_Clear()

Name: FibMax_Clear

Argument: None

Function Description: It initializes the member function of the class like Biggest_node and SizeOfHeap.

- **public** fibMaxNode<T> FibMax_maximum()

  Name: FibMax_maximum

  Argument : null

  Function Description: This is the maximum or highest element in the heap. THis has the highest key element.

- **public void** removeNodeFromList(fibMaxNode<T> node)
  Name: removeNodeFromList(
  Function Description: This function removes the node from the doubly linked list in the Fibonacci Heap. I have created this function because it is called by more than one function in this class.

- **public void** add_to_root_list (fibMaxNode<T> x, fibMaxNode<T> Biggest_node)
  Name: add_to_root_list
  Function description: This function inserts this node to the top-most doubly linked list containing the root. This function has been created since it has been called in more than one function in this class.

- **public void** FibMax_Insert(fibMaxNode<T> node, **double** Heap_Key)
  Name: FibMax_Insert
  Arguments: node, Heap_Key
  Function description: If no heap is present, this would append the node to be inserted to the left of the biggest node and it is present in the same doubly

linked list where the biggest element is present. Otherwise if no max element is present, this newly inserted node is made as the biggest node.

- **public** fibMaxNode<T> Fib_RemoveMax()
  Name: fibMaxNode
  Arguments: null
  Function description: This function removes the biggest element or the node with the highest key and appends the children to the top level list. It uses pairwise-combine method to merge nodes with the same degree.

- **protected void** Fib_Pairwise_Merging()
  Name: Fib_Pairwise_Merging
  Function_Description: This function merges nodes having same degree by making the node with smaller element as child of the other node. THis function is made protected so that it can be accessed only by functions of the same class. It is called by the function Fib_RemoveMax().

- **protected void** FibMax_CascadeCut(fibMaxNode<T> y)
  Name: FibMax_CascadeCut
  Arguments: fibMaxNode<T> y
  Function description: This function is called when child cut is true and the node has lost more than one child.
  THis is called by the fibmax_key_increase function.

- **public void** FibMax_Key_Increase(fibMaxNode<T> x, **double** k)
  Name: FibMax_Key_Increase
  Function description: Thiis function increases the key of the node. If key is bigger than parent node, it is cut off from its parenent.
  Argument: fibMaxNode<T> x, **double** k