COP 5536 -Advanced Data Structures

Spring 2020 - Programming Project Report

Implementation of **Max Fibonacci Heap** & **Hash Table** to find the n most popular Hashtags

Author: Shashank Bantakal **UFID#** 9592-6559 **UF Email:** sbantakal@ufl.edu

Problem Description

To implement a program to determine the most popular hashtags that appear in social media such as Facebook and Twitter. The hashtag and their corresponding number of occurrences will be fed to the program through an input source file.

Idea of implementation is to use a priority queue structure to find the n most popular hashtags.

How to run the program?

```
$java hashtagcounter.HashTagCounter <input_file> <output_file>
```

Program Structure

The submission contains 4 source files, each having functions/methods. Below are the files/classes and details of the functions and methods defined in them.

1. **Node.java**, This class defines the structure of the nodes in the Max Fibonacci Heap.

   Class Variables:

private int degree = 0;	→ No of child nodes associated to the node, 0 by default.
private int data;	→ To track the count associated with the hashtag.
private String key;	→ To track individual hashtags.
private boolean childCut = false;	→ Tracks whether a node has lost any child node, false by default.
Node right, left;	→ Pointer to right & left siblings of the node.
Node parent = null;	→ Pointer to the parent node, null by default.
Node child = null;	→ Pointer to the parent node, null by default.

   Methods:

   - public Node(int d, String k) : is a constructor method that creates a new node with data as d and its corresponding key equal to k.
   - public int getDegree() : getter function to retrieve the degree of a node.
   - public void setDegree(int degree): setter function to reset the degree of the node.

- public int getData(): getter function to get the data associated with a node.
- public void setData(int data): setter function to reset the data associated with a node.
- public String getKey(): getter function to get the key associated with a node.
- public void setKey(String key): setter function to reset the key associated with a node.
- public boolean isChildCut(): function to check if the childCut field is set.
- public void setChildCut(boolean childCut): setter to reset the childCut field of a node.

2. **FiboHeap.java**, Contains implementation of a Max Fibonacci Heap.

Class Variables:

private Node max = null; → Pointer to the node with maximum data in the heap. Null by default.
private int size = 0; → To track the current size (no of nodes) in the heap.
Queue removedMaxQueue = new LinkedList(); → To keep track of nodes removed from the heap.
HashMap<String, Node> hashMap = null; → To track all the nodes in the heap.

Methods:

- public int getSize(): getter function to retrieve the current size of the Fibonacci heap.
- public void setSize(int s): setter function to reset the size of the Fibonacci heap.
- public void setMax(Node max): setter function to reset max node pointer to a specified node.
- public Node getMax(): getter function to retrieve the node containing the maximum data in the Fibonacci heap.
- public boolean isEmpty():
- private static Node combine(Node p, Node q): this function combines two independent lists into one and returns a pointer to the maximum node in the list.
- public Node insert(int value, String key): this function creates a new node and adds the same to the Fibonacci heap.
- public FiboHeap meld(FiboHeap f, FiboHeap s): this function takes two fibonacci heaps as input and merges them into one and returns the result. It also deletes the node heaps in the process.
- public Node removeMax(): this function deletes the node with the maximum data value from the Fibonacci heap. It also performs pairwise combine to nodes that have same degree, thereby creating nodes/trees of higher degree.
- private void cascadingCut(Node pNode): during increase key operation if the data of a node becomes greater than that of its parent node, it is removed from its parents subtree and added to list at the root level. At this instance, if the parent node's childCut field is already then it too is removed from its subtree and added to root level linked list. This process continues recursively until first node with childCut field not set is found or root level node is reached.
- public void increaseKey(Node target, int incBy): increases the data associated with the node by the amount specified. If the data of a node becomes greater than that of its parent node, it is removed from its parents subtree and added to list at the root level.
- public void removeNMaxes(int n, String filename): performs n remove operations, tracks the removed nodes in the Fibonacci heaps removedMaxQueue for reinsert. The

function also redirects the hashtags associated with each of the removed nodes to an output data file.

- public void removeNMaxes(int n): performs n remove operations, tracks the removed nodes in the Fibonacci heaps removedMaxQueue for reinsert. The function also redirects the hashtags associated with each of the removed nodes to be printed on the screen.

3. HashTagCounter.java

Methods:

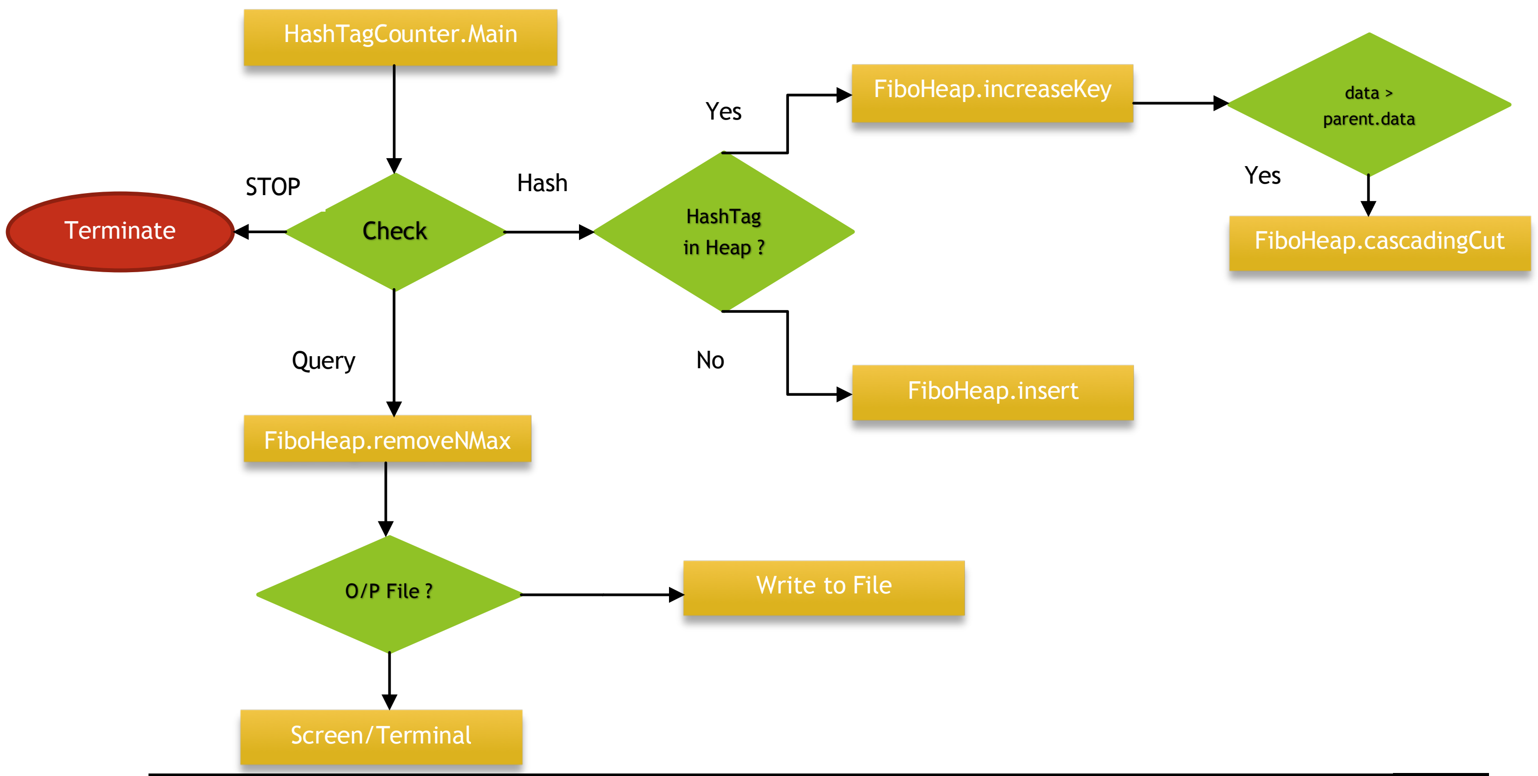
- public static void main(String[] args): The control function of the program. If no input file is specified it raises an IncorrectArgumentsException. If an input file is specified, the program reads the data from the file and performs creation of a Fibonacci heap, insertion and removeMax operations as per the data in the feed file. In case if an output files is specified, the program redirects the outputs from the removeMax operations into the output file, if not, it redirects the outputs to be printed on the terminal.

4. IncorrectArgumentsException.java

Methods:

- public IncorrectArgumentsException(String eMessage): overrides the constructor of the parent class Exception by the error message eMessage passed.

Program Structure & Flow


HashTagCounter.Main
STOP
Terminate
Check
Hash
HashTag in Heap ?
Yes
FiboHeap.increaseKey
data > parent.data
Yes
FiboHeap.cascadingCut
No
FiboHeap.insert
Query
FiboHeap.removeNMax
O/P File ?
Write to File
Screen/Terminal