**COP 5536 Spring 2020**

**Project Report**

**Shantanu Ghosh**

UF Id: 4311 – 4360

shantanughosh@ufl.edu

**Problem description**

A system is implemented to find the n most popular hashtags that appear on social media such as Facebook or Twitter. 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 structure to find out the most popular hashtags.

**Assumption**: There will be a large number of hashtags appearing in the stream and I need to perform increase key operation many times. Max Fibonacci heap is recommended because it has an amortized complexity of O(1) for the increase key operation. I have implemented all Fibonacci heap functions discussed in class. For the hash table, existing implementation of Hashtable in the java.util package has been used.

**Programming Environment**

This project is implemented in Java version 13.

**Implementation**:

1. **Max Fibonacci heap**: Fibonacci heap is a data structure which is primarily used for priority queue operations. It has better amortized complexity than normal Binary heap and Binomial heap. The amortized time complexity for find\_minimum, insert, decrese\_key operations are constant. For a heap size of n, the amortized complexity of deleting an element is O(logn).

In this project, Fibonacci heap is used to keep track of the frequencies of hashtags.

2. **Hash table:** The key for the hash table is the hashtag, and the value is the pointer to the corresponding node in the Fibonacci heap.

**Execution**:

Following are the required steps to execute the project:

* Unzip the project
* Paste the input file into the project folder
* make
* java hashtagcounter <input\_file\_name> <output\_file\_name>
* If output file name is mentioned, view the output in the output file, else view the output in the output\_file.txt

**Project Structure:**

**A screenshot of a cell phone

Description automatically generated**

**FibHeapNode:** This is class contains the basic node structure of the Fibonacci Heap

**FibonacciHeapOperations:** This class consists of all methods which implements the different operations of Fibonacci Heap Data structure.

**HashtagCounterBL:** This class implements the business logic of the problem statement to find n most popular hashtags

**Util:** This is a utility class which consists of utility methods that has been used by HasTagConterBL Class.

**hashtagcounter:** This class contains the main method which is entry point of this project.

**Detailed description of the Methods of the classes will be mentioned below.**

**Project Documentation:**

**1. Class Name - FibHeapNode:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Return Type** | **Member Type** | **Member Name** | **Purpose** |
| FibHeapNode | Variable | child | Child node given a parent |
| FibHeapNode | Variable | left | Left node of a child given a parent |
| FibHeapNode | Variable | parent | Parent node |
| FibHeapNode | Variable | right | Right node of a child given a parent |
| String | Variable | tag | Hashtag |
| Integer | Variable | key | Key of stored at the node |
| Integer | Variable | degree\_of\_node | The no of children of a node |
| Boolean | Variable | child\_cut | * **True** if node has lost a child since it became a child of its current parent. * Set to **false** by remove min, which is the only operation that makes one node a child of another. * **Undefined** for a root node |
| String | Method | getTag | Getter method to retrieve the Hashtag |
| void | Method | setTag | Setter method to set the Hashtag |
| void | Method | setKey | Setter method to set the Key |
| String | Method | getKey | Getter method to retrieve the Key |

**2. Class Name - hashtagcounter:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Return Type** | **Member Type** | **Member Name** | **Purpose** |
| void | Method | main | Entry point of the project |

**3. Class Name – Util:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Return Type** | **Member Type** | **Member Name** | **Purpose** |
| FibHeapNode | Variable | child | Child node given a parent |
| FibHeapNode | Variable | left | Left node of a child given a parent |
| FibHeapNode | Variable | parent | Parent node |

**4. Class Name – HashTagCounterBL:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Return Type** | **Member Type** | **Member Name** | **Purpose** |
| File | Variable | file\_out | Output file Name |
| StringBuilder | Variable | sb | Temporary placeholder to hold the contents of the output file |
| ArrayList | Variable | node\_removed\_store | Hold the removed nodes ftom the heap |
| FibonacciHeapOperations | Variable | heap\_operations | Object of the class FibonacciHeapOperations which contains all the implementation of operations performed in Fibonacci Heap. |
| Hashtable | Variable | fib\_store | Object of the Hashtable |
| void | Method | initiateCounter | Entry point of the business logic which redirects to appropriate methods where operations on a Fibonacci heap is performed and to the method which query the Fibonacci heap |
| void | Method | writeToOutput | Writes the output to the output file |
| void | Method | createNewNode | * Creates a new node in the Fibonacci heap * Update the frequency and add the node with update frequency to the tree. |
| void | Method | increseKey | Update the frequency count for a given node |
| void | Method | insertNew | Insert a new node in the Fibonacci Heap |
| void | Method | queryHeap | Query the Fibonacci heap with the given frequency of a hashtag |
| void | Method | addBackRemovedNode | Remove the node to update the frequency and add with updated frequency |
| String | Method | removeMaxNode | Remove the node with max frequency |
| String | Method | getTagForMaxNode | Get the tag for the node to be deleted with max key |

**5. Class Name – FibonacciHeapOperations:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Return Type** | **Member Type** | **Member Name** | **Purpose** |
| FibHeapNode | Variable | node\_with\_max\_key | node with the maximum key |
| int | Variable | no\_of\_nodes | Total no of nodes of the Fibonacci heap |
| void | Method | insert | Insert a new node in the Fibonacci Heap |
| FibHeapNode | Method | removeMax | Remove the node with maximum key |
| FibHeapNode | Method | increaseKey | Increase the key of a node with updated frequency |
| void | Method | removeArbitaryNodeFromHeap | Remove any node from Fibonacci heap |
| FibHeapNode | Method | deleteRootWithSameDegree | Delete node with same degree |
| void | Method | deleteNodes | Perform delete nodes |
| void | Method | reconstruct\_root\_list | Reconstruct nodes during pairwise combine |
| void | Method | addNode | Add given node during insertion operation of Fibbonacci Heap |
| void | Method | removeNodeFromRoot | Perform removal of node from Fibonacci heap |
| void | Method | updateChildCount | Update the degree of the child during removeMax operation of Fibonacci Heap |
| void | Method | cut | Cut the child |
| void | Method | cascadeCut | Performs the cascade cut operation |
| void | Method | pairwiseCombine | pairwise combine min trees whose roots have equal degree. |
| void | Method | travarseTree | Traverse the Fibonacci Heap |
| FibHeapNode[] | Method | rootNumList | Uitlity method used by the method to remove a node with same degree |

**References:**

* <https://en.wikipedia.org/wiki/Fibonacci_heap>
* <https://www.geeksforgeeks.org/fibonacci-heap-set-1-introduction/>
* Introduction to Algorithms – CLRS
* <https://www.cise.ufl.edu/~sahni/cop5536/index.html>