Advanced Data Structures

Sparsh Srivastava : 352-871-5556
sparshsrivastava@ufl.edu
UFID – 1590 6619

Problem Statement:

The project aims to implement a counter to find the n most popular hashtags that appear on social media such as Facebook or Twitter. It makes use of a Max Fibonacci Heap to keep track of most frequent hashtags. Max Fibonacci Heap is recommended because it has an amortized complexity of O(1) for the increase key operation. The output of the developed code structure can be written into a separate file if name of output file is specified. Otherwise the output will be displayed on the console. I have used the following data structures for the implementation:

1. **Max Fibonacci Heap**: It is used to keep track of the frequencies of the hashtags.

2. **Hash table**: The key for the hash table is the hashtag and the value is the pointer to the corresponding node in Fibonacci Heap. A hash table is used to keep track of the hashtags that already appeared and to increase its value if it appears again later in the input.

Execution of program:

- The project is implemented in **Java** language. IDE Used: Eclipse.
- Extract **Srivastava_Sparsh.zip**.
- Open command prompt and enter command **make** to execute the makefile and build the binary.
- Execute the binary with the input file by running the command: **javac hashtagcounter name_of_input_file.txt name_of_output_file**.
- Output is written into **name_of_output_file**. If not mentioned then all the hashtags are printed onto the console.

Project Structure:

● hashtagcounter.java - The file contains three classes namely **hashtagcounter, FibonacciHeap** and **FibonacciNode.** The **hashtagcounter** class consists of the driver function **public static void main().**

● Makefile - This is an executable file that is used to build the binary. The binary is later used to run the program by passing the input file name as a command line argument.

Classes and Important Methods:

a) FibonacciNode Class:

This class is used to encapsulate all the data fields related to each node (structure of each node) created when a new hashtag is read from the input of doubly circular linked list of Fibonacci Heap. Apart from this, the class also contain the getter, setter and the constructor to it.

Data Fields:

- ✓ private String **key** - name of the hashtag.
- ✓ private int **degree** = 0 - stores the current degree of the FibonacciNode.
- ✓ private int **data** - Count associated with each hashtag.
- ✓ private FibonacciNode **left** - Points to the left sibling of the FibonacciNode.
- ✓ private FibonacciNode **right** - Points to the right sibling of the FibonacciNode.
- ✓ private FibonacciNode **child** - Points to child FibonacciNode.
- ✓ private FibonacciNode **parent** - Points to parent FibonacciNode.
- ✓ private boolean **childCutVal** = false - tells whether sub tree of this node has been cut before or not.

```
private String key; // name of the hashtag
private int degree = 0; // stores the current degree of the FibonacciNode
private int data; // Count associated with each hashtag

private FibonacciNode left; // Points to the left sibling of the FibonacciNode
private FibonacciNode right; // Points to the right sibling of the FibonacciNode
private FibonacciNode child; // Points to child FibonacciNode
private FibonacciNode parent; // Points to parent FibonacciNode

private boolean childCutVal = false; // tells whether sub tree of this node has been cut before or not
```

Important Methods:

1) **FibonacciNode(String keyValue, int value) :** It constructs a new Fibonacci heap node with the hashtag and the count value specified.

2) **Setter and Getter of each data field specified above**.

```
public String getKey() {
    return key;
}

public void setKey(String key) {
    this.key = key;
}

public int getDegree() {
    return degree;
}

public void setDegree(int degree) {
    this.degree = degree;
}

public int getData() {
    return data;
}

public void setData(int data) {
    this.data = data;
}

public FibonacciNode getleft() {
    return left;
}

public void setleft(FibonacciNode left) {
    this.left = left;
}

public FibonacciNode getright() {
    return right;
}

public void setright(FibonacciNode right) {
    this.right = right;
}

public FibonacciNode getChild() {
    return child;
}

public void setChild(FibonacciNode child) {
    this.child = child;
}
```

```
    public FibonacciNode getParent() {
        return parent;
    }

    public void setParent(FibonacciNode parent) {
        this.parent = parent;
    }
```

```
    public boolean isChildCutVal() {
        return childCutVal;
    }

    public void setChildCutVal(boolean childCutVal) {
        this.childCutVal = childCutVal;
    }

    FibonacciNode(String keyValue, int value) {
        this.right = this;
        this.left = this;
        this.data = value;
        this.key = keyValue;
    }
}
```

b) FibonacciHeap Class:

This class is used to represent Fibonacci Heap as a data structure. It consists of various functionalities related to Fibonacci Heap such as insert, remove max, increase key etc. Each functionality is defined in a separate method. Along with it, an instance of this class object consists of an output queue, a pointer to the max count node in the heap and its respective hash table and size.

Important Methods:

1) **public boolean checkKey(String checkKey1):** It checks whether the hash table contains the String being added by the user or not. The argument passed for this method is the name of the hashtag.

2) **public void addValue(String checkKey1, FibonacciNode node):** It adds the inserted hashtag into the hash table. The arguments passed for this method is the name of the hashtag along with it the node structure of the hashtag.

3) **public FibonacciNode getright1(String checkKey1):** It points to the position of the node in the heap and based on it count of this node will be increased.

4) **public void removeElements(int num, String outputPath**): This method is used to perform remove maximum functionality number of times asked by the user. Along with it, it is used to call the printTags method where the output of remove maximum functionality will be written into a separate output file and insertAgain method to again insert the removed hashtags into the list. The arguments passed for this method is the number of most popular hashtags which should be printed into the ouput file if needed. Along with it output path which will tell us where the contents should be written. If not specified that n most popular hashtags will be printed onto the console.

5) **public void printTags(FibonacciNode node, PrintWriter printWriter, String outputPath, int numPrinted, int tobePrinted):** This method prints the n most popular hastags in the Fibonacci Heap into the output file if present. The arguments passed for this method is the node where the name of the hashtag resides, printWriter writes into the output file, outputPath is given but not used, numPrinted tells how many hashtags have been written till now and tobePrinted tells how many hashtags needs to be printed in total.

6) **private void insertionAgaintoFiboHeap(Queue<FibonacciNode> que):** This method inserts the n most popular hashtags removed in the previous function again

into the heap. The argument passed is the queue which stores name of all the hashtags removed from heap.

7) **private void cutChildNode(FibonacciNode fibNode):** Based on the child cut flag value checked in increase key method this one will be executed. If true, we cut the node from the heap, move it to the root degree, and then recombine with the heap. We then check the same thing for its parent. For that another method is called whose name is parentNode() . The argument for this method is the node on which this child cut functionality needs to be performed.

8) **private void parentNode(FibonacciNode node):** Similar functionality like child cut will be performed here. The only difference is that it is for the parent of the child cut from the heap. This process will be repeated until the parent has child cut flag set to false occurs. The argument for this method is the node on which this child cut functionality needs to be performed.

9) **public void incKey(int incVal, FibonacciNode fibNode):** This method increases the frequency of an existing hashtag in the heap with the new value provided by the user. Its increased value is then checked against the value of its parent. If its value comes out to be more than its parent, then setting child cut flag to false and removing it from its parent list and adding into the root list. The arguments passed for this method is the value by which hashtag needs to be incremented and node on which this operation needs to be performed.

10) **public FibonacciNode insertNode(FibonacciNode fibonacciNode):** This method inserts a new node which has name of new hashtag along with its frequency into the heap. It is then compared with max element of the heap. If it is found to be greater than max element then this node will be made new max element of the heap. The argument for this method is the node which will be inserted into the heap.

11) **public void initialInsertNode(FibonacciHeap heap, String hashtag, int count):** This method is called in the main method. It is the actual insert which is performed when data is from input file. The arguments passed to this method are name of the hashtag, count or frequency of the hashtag and heap in which node will be inserted.

12) **public FibonacciNode removeMaximum():** This method removes the maximum frequency hashtag from the heap. No arguments are required for it as it makes use of heapMax which tells the maximum element of the heap.

c) hashtagcounter Class:

This class is the driver class that contains the main method. In addition to it, couple of member variables are defined which cannot be changed and has to be used as constants in this class.

Data Fields:

- ✓ public static final char **HASHTAG** = '#';
- ✓ public static final String **BLANK** = " ";
- ✓ public static final ArrayList<String> **STOPPERS** = new ArrayList<>() – This gives us a

list of values, occurrence of which program terminates.

Important Methods:

public static void main (String args[]): This is the first method which is executed on running our program. Based on it remaining functions or classes will be called. It contains the name of input and output file (if present) that will be used to retrieve data from and write to. It starts reading every line from the input file, determines whether we need to add this hashtag into the heap or read hashtags. It terminates when “stop” or “STOP” or “s” comes in input file. If the input line has “#” in it, then it checks whether this hashtag already exists or not. If it exists then it calls increase key function otherwise it is inserted into the heap. If it is an integer, it reads the number of popular hashtags given by the input.