Advanced Data Structures (COP5536)

Spring 2020

Programming Project Report

Pulkit Sanadhya

UFID : 2101-2451

pulkitsanadhya@ufl.edu

COMPILING INSTRUCTIONS:

Compile the file by using the following instructions:

```
javac hashtagcounter.java
```

Now , run by using the following command to generate output in the output file specified by the user as a second argument.

```
java hashhtagcounter input.txt output.txt
```

Run the following command to generate the output on the console if no second argument is provided.

```
Java hashtagcounter input.txt
```

FUNCTION PROTOTYPES AND CLASSES

We have used three classes to implement the functionalities and to fulfil the requirements of the project.

1.hashtagcounter

2.FiboHeap

3.FibNode

hashtagcounter:

This class essentially contains the main function for given program where we have used the FileReader class to read the input file from command line arguments. Here we implemented that functionality that if no command line input are provided or if more than two command line inputs are provided then the program will be terminated and ask the user to provide correct arguments. If <u>no output file is provided</u> as a command line argument, then the output will be generated on the console window. And, <u>if the output file name is provided</u> then the result will be written to the provided output file name in the same directory and we have used the FileWriter class for implementing this . The program throws an IO exception if the file is inaccessible. The class uses instances of Hashtable , FiboHeap and Node to implement various functionalities.

FiboHeap :

We implemented the functionalities of a max Fibonacci heap in this class by using various methods as explained below.

void nodeInsert(FibNode n1) : This method was used for insertion of a new node into the max Fibonacci heap structure.

void incKey(FibNode a, int val): This method takes the increased value of frequency of the hashtag which was calculated in the hashtagcounter class and makes a corresponding increase in the node value of max Fibonacci heap .

void removeNode(FibNode a , FibNode b) : This method removes the node a from child of node b .

void cascadeCut(FibNode b) : This method performs the cascading cut operation if the child cut value of the parent of a removed node is false.

void merge() : This method is used to merge the nodes of same degree into a single heap. It is called in the removeMax method.

Node removeMax(): This method is used to extract the maximum value of Fibonacci heap and it returns the removed node which can be either displayed on the console or can be written to an output file as per the requirements.

void mChild(FibNode b,FibNode a) : This method is used to make a node b as a child of a.

FibNode:

This class essentially contains the structure of the node that will be inserted into the Fibonacci heap:

The variables that are declared in this class are :

deg: It defines the degree of the node which is set as 0 by default.

childCut : It defines the value of child cut which is set as false by default.

nHash : It defines the hash value of the node which is being passed by using its constructor.

Key : It defines the key value of the node which is passed by using its constructor.

The method implemented in it are :

String getTag() : It is used to return the hash value of a given node with a string return type.

PROGRAM STRUCTURE

```
public class hashtagcounter
{
 --Hashtable and heap initialization
 --Condition to check the command line arguments.
    try {
        --Input and output stream initialization
        --parsing the input
         If(input is a hashtag)
           {
             If ( hashtag exists in hashtable)
              {
                 incKey(FibNode a,int val);
              }
```

```
                else
                {
                nodeInsert(FibNode a);
                }
             }
          else if ( parsed string is a digit)
                {
                  removeMax();
                  --Display the removed nodes on console or in output file.
                  --Again insert the nodes into the heap.
                  nodeInsert(FibNode n);
                }
            else if ( parsed string is stop)
               {
               --stop the execution of code.
               }
    }
```