Advanced Data Structures Project (COP5536)

SPRING 2020

Hashtag Counter using Max-Fibonacci Heap

Kaustubh Katkar

UFID: 31470922

k.katkar@ufl.edu

Compilation Instructions:

Program can be compiled using either of the two ways:

1. Go to the project directory and type "make" in the command line
2. Go to the project directory and type "javac HashtagCounter.java" in the command line

Run program:

To display output on the console: **$** java HashtagCounter input.txt

To write output in a file: **$** java HashtagCounter input.txt output.txt

output and input filenames can be chose as per the validators liking.

Function Prototypes and Classes:

FiboNode structure:

Variables:

parent – Contains parent node pointer

child – contains child node pointer

left – contains left sibling node pointer

right- contains right sibling node pointer

degree – number of children of the node

childCut – denotes if a child has been cut since this node has become a parent.

hashtag – contains hashtag tree

key – contains frequency of hashtag

Functions	Working
FiboNode(String hashtag, int key)	• Is the constructor • Sets hashtag and key values of the FiboNode object

HashtagCounter Structure:

Variables:

key – stores key value(integer) from input file

hashtag – stores hashtag string from input file

line – stores each line as string while traversing the file

outputType – determines how the output is handled. Boolean variable with true or false values.

Data Structures:

Hashtable<hashtag, FiboNode> – stores hashtag to node mapping.

ArrayList<FiboNode> - stores removed nodes to reinsert them into the Fibonacci Heap.

IO:
Users BufferedReader, BufferedWriter, FileReader, FileWriter to traverse the file and develop corresponding output.

MaxFibHeap structure:

Variables:

maxNode – contains the node with the maximum key value.

numNodes – contains the total number of nodes in the Fionacci Heap.

Functions	Working
insert(node)	• Inserts nodes to the topmost level
delete(child, parent)	• Deletes a node pointer from parent and adds the node to the topmost level. Node.parent becomes null. • If node is max, new max is computer with setMax. Melding does not take effect.
removeMax()	• Deletes the max node using delete. • Calls meld function thereafter. • Computes new Max.
increaseKey(node, value)	• Increases the key of the node if node exists. No action otherwise
meld()	• Melds/merges two trees with equal degrees. • Determines the order of melding between nodes. • Calls meldAsChild(n1,n2) function.

Auxiliary Functions: These functions support the working of the above methods.

Functions	Working
checkCascadingCut()	• Checks if parent childCut = true after every delete operation. • If true cascadingCut takes place through recursion and the node that are cut get added to the topmost level.
meldAsChild(node1, node2)	• Makes node2 the root and node 1 a child of node 2

Program Structure:

The aim of the project is to take and input file with hashtags and the frequency describing the occurrence of each hashtag on each line. When the program encounters line starting with an integer the program outputs the number of hashtags with maximum key-value or frequency at this point. This process terminates only when “stop” is encountered as the beginning of a line.

The program has the following structure:

HashtagCounter.java

- Initialize variables
- Initialize Hashtable data structure
- Initialize MaxFibonacciHeap data structure -***MaxFibHeap.java***
- Initialize readers and writers
- try {
    - Declare readers and writers.
    - If line begins with "#" –
        - Stores string after "#" in hashtag
        - Stores in after " " in key
        - Checks Hashtable if hashtag exists
        - If exists – ***MaxFibHeap.increaseKey()***
        - If doesn't exist – create ***FiboNode***
        - ***MaxFibHeap.insert(FiboNode)***

    - If line begins with a digit –
        - Initialize ArrayList data structure to store remove nodes
        - For(number of iterations = digit) {
            - ***MaxFibHeap.removeMax***
            - Remove node from Hashtable
            - Add node to ArrayList
        - Display output as determined by outputType variable
        - Again, insert the removed nodes from ArrayList to the Max Fibonacci Heap
    - }

- catch {
    - Exception e
    - }
- finally {
    - Flush writer
    - Close reader and writer
    - }