Advances Data Structures (COP 5536)
Fall 2016
Programming Project Report

Leela Krishna Chaitanya Prava
UFID- 95304385
leelakprava@ufl.edu

PROJECT DESCRIPTION:

The aim of the project is to use a max Fibonacci Heap to find the most used hashtags. The input would be a list of hashtags along with their frequency. The data would be queried for finding the trending hashtags frequently. The query requires to find a certain number of highest used hash tags from the data accumulated until then.

The Fibonacci Heap created, stores every hashtag with a unique node in the heap along with its frequency of occurrence. To evaluate a query, the node with the highest frequency ‘max Node’ is removed from the heap. After the removal of the max node, its children are added to the heap and the node with the same degree are combined to form a bigger tree with higher degree and the heap is assigned a new ‘max Node’. After the evaluation of the query, the nodes removed are added back into the heap.

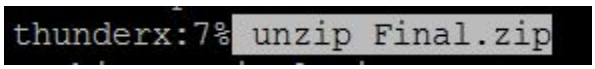
WORKING ENVIRONMENT:

HARDWARE REQUIREMENT

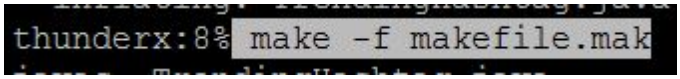
1. Hard Disk space: 4 GB
2. Memory:512 MB
3. CPU: x86
4. OPERATING SYSTEM: Windows 10
5. Compiler: javac

COMPILING INSTRUCTIONS :

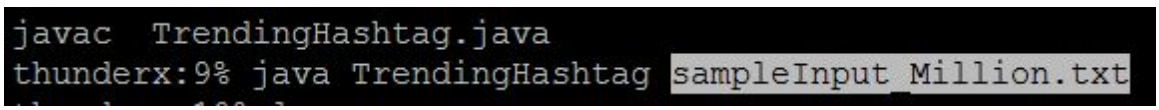
1. Unzip the file ‘Prava_Chaitanya.zip’



2. Run the makefile file - ‘makefile.mak’
   Use the following command - make -f makefile.mak



3.Run the java class ‘TrendingHashtag’ using command ‘java’



4.Give the input text file as command line argument as shown in the above figure, ie replace the ‘sampleInput_Million.txt’ with the name of the input text file along with its extension.

5. The result would be saved in the same folder with the name ‘output_file.txt’.

STRUCTURE OF THE PROGRAM AND FUNCTION DESCRIPTIONS

There are three classes, used in the assignment. They are Node, Fibo_Heap and TrendingHashtag.

Node: gives the structure of a heap element and has the constructor defining the element.

Fibo_Heap: contains the Head node(element with maximum frequency) and the methods required for insertion and removal of fibonacci heap elements.

TrendingHashtag: contains the main function which is used for instantiation of the Fibo_Heap object and reading and writing I/O to file.

class Node:

This class gives the structure of a heap element and has the constructor defining the element

Variables:

elem:

It contains the Hash tag. It is a variable of type String.

frequency:

It contains the number of times the tag has appeared. It is a variable of type int.

degree:

It contains the number of child nodes which are pointed towards the node. It is a variable of type int.

marked:

It signifies whether or not a child has already been removed from that node. It is of type boolean.

head:

It serves as a pointer to the parent of the node. It is of type Node.

left:

It serves as a pointer to the left sibling of the node. It is of type Node.

right:

It serves as a pointer to the right sibling of the node. It is of type Node.

child:

It serves as a pointer to a child of the node. It is of type Node.

class Fibo_Heap:

This class holds the Head node, which holds the element with the maximum frequency. The class has many methods to access and manipulate the heap to achieve desired results.

Variables:

Head-

This is a variable of type Node. It stores the element with the maximum frequency in the heap.

tree_count-

This variable stores the total number of trees in the heap. It is a variable of type int.

Method:

Node Insert(String element, int frequency):

Input parameters: element, frequency

Return type: Node

This method takes in the hashtag and its frequency as input, creates a node and inserts it onto the heap. I the newly inserted node has a higher frequency than the Head node of the Fibo_Heap object then the Head nod is updated by the function. It returns the newly inserted node.

void Casc_cut(Node t):

Input parameters: Node t which needs to be cut

Return type: void

This function checks if the node needs to cut from its parent and siblings and be added to the heap. It recursively calls itself for the parent of the cut node if the parent is marked ‘true’ and the head of the parent is ‘null’.

void Insert_back(Node t):
Input parameters: Node which has to added back to the heap.
Return type:void
This function adds elements to the heap. It is different from Insert() as it does not create new nodes it just moves trees from on level to another during Casc_cut(),remmax() and while adding back the max element removed from the heap.

Node remmax():
Input parameters:void
Return type: Node this is the maximum node which is removed from the heap during query processing.
Variables:
HashMap<Integer,Node>consol_queue,stores the degree of the trees to assist pairwisemerge().This function is invoked when the heap is queried. The method removes the Head node consolidate the heap by void pairwisemerge() and returns the removed max node.

pairwisemerge(HashMap<Integer,Node> consol_queue ,Node t1,Node t2):
Input parameters: consol_queue is used for the merging of the trees t1 and t2.
Return type : void
The function combines given trees depending on their frequencies and their head and siblings. It checks the consol_queue to find if there is an entry of similar degree to merge with the current merged tree. If the there exists a tree in consol_queue whose root's degree is same as that of the merged node, then pairwisemerge() would call itself for both the trees.

class TrendingHashtag:

This class has the main() method. It also reads the data from the file and writes output back to the file. It creates the an object of Fibo_Heap, which used for creating the heap and performing various operations to achieve the required result.

CONCLUSION:

Hence I implemented a Fibonacci heap to find out the trending hashtags for a particular set of data. It was a max Fibonacci Heap.