Advanced Data Structures.
COP5536.
Fall 2016.
Programming Project Report.

Name: Sridhar Reddy Maddireddy.
UF-ID: 02151989.
srimadd@ufl.edu

Project Description

Our idea is to implement fibonacci heap to store the twitter/facebook hashtags and query the top n tops at any time. We are using hash table to store the hash tag and a pointer to the node in heap.

We read a file of hash tags to store in fibonacci heap and output the query results to output_file.txt file which contains top n tags separated by comma.

Running the program

Implemented the project in C++.

Compiler: g++.

To compile “g++ -o hashtagcounter FibonacciHeapMain.cpp”.

To run “./hashtagcounter file_name”.

Makefile is included in the project. So, use make to generate hayhtagcounter executable file.

Program Structure

Nodes in the structure have the below structure.

```
//Fibonacci node structure.

struct FibonacciNode{

    //parent node...
```

```
        FibonacciNode *nodeParent;

        //right sibling of the node...
        FibonacciNode *rightSibling;

        //hash tag count value...
        int count;

        //degree of the node...
        int degree;

        //is child cut value marked?...
        bool isMarked;

        //storing hashtag value...
        string hashTag;

        //left sibling of the node...
        FibonacciNode *leftSibling;

        //child node...
        FibonacciNode *nodeChild;

        //constructor for a new node...
        //which takes hastag and the corresponding count values...
        FibonacciNode(int hashCount, string hshTag){
                nodeParent = NULL;

                //circular linked list right sibling points to same node..
                rightSibling = this;

                count = hashCount;

                //every newnode degree will be 0...
                degree =0;

                //default ismark is set to false...
                isMarked = false;

                hashTag = hshTag;

                // circular linked list left sibling points to same node...
                leftSibling = this;
                nodeChild = NULL;

        }
};
```

Contains a single class “FibonacciClass” which contains the below functions and private variables.

Private variables:

FibonacciNode * maxPtr;

//Always points to a node which contains maximum hash tag count.

map<string, FibonacciNode*> hashTable;

//hashtable to store the hashtag and corresponding pointer to node in fibonacci heap.

int noOfNodes;

//number of nodes in the fibonacci heap.

Functions used:

void **createNode**(int hshCount, string hshTag){

Parameters: int hashtagcount value, string hashtag

Return type: void

//takes hashtag and its count and creates a new node or if hash tag is already there then increase key will be called..

}

string **getMax**(){

Parameters: none

Return type: string

//return hash tag of the max count if at least single node is present.

}

int **getMaxCount**(){

Parameters: none

Return type: int

//returns hash tag's count of the max count if at least single node is present.
}

FibonacciNode * **deleteNode**(FibonacciNode * node){

Parameters: Fibonacci node pointer

Return type: FibonacciNode

//deleting the node "FibonacciNode *node"
//and merge with siblings if any are present.
//deletes in O(1) time.
}

int **removeMax**(){

Parameters: none

Return type: int

//removes the max pointer node from the heap.
//do the pairwisecombine if maxptr is present.
//change the required the maxPtr after removing the existing one.
}

void **pairWiseCombine**(){

Parameters: none

Return type: void

//pair wise combine after removeMax().
//check the degrees of node in the root circular list and merge root node with same number of degree.
}

void **increaseKey**(int new_count, string hsTag){

Parameters: int new count, string hashtag

Return type: void
//get the hashtag and pointer and store in a temp node.
// increase the hashtag count of the temp node
// remove the old entry from the hash table and reinsert the temp node with the updated hashtag count value.
// no check whether it satisfied max heap property
//if max heap property fails then call child cut to reinstate the max heap property.

```
        //modify the maxPtr if required.
        //increases the hash tag count by new_count
}
```

void **childCut**(FibonacciNode *node){

Parameters: Fibonacci *node

Return type: void

```
        //cascading child cut for nodes which are marked true after increase key.

        //will call recursively until it reaches node with isMarked value as false.

        //for every call stack it removes from the heap and reinserts to top level circular list.
}
```

int **main**(int argc, char* argv[]) {

```
        //driver program to start and running

        //takes the input from the files given and stores/adds to the fibonacci heap

        //for every query in the input file, output is written to output_file.txt and each hastag is
          separated by comma
}
```

——————————————

Conclusion

Successfully implemented fibonacci heap to store twitter/facebook hashtags and retrieve the top n nodes from the heap from the files and write output to output file.