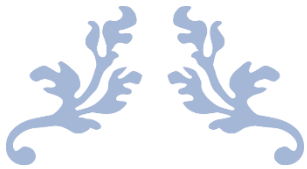

ADVANCED DATA STRUCTURE

Project: hashtagcounter

Yue Jia

UFID: 8511-2366

yuejia@ufl.edu

Project description:

You are required to implement a system to find the n most popular hashtags that appear on social media such as Facebook or Twitter. For the scope of this project hashtags will be given from an input file.
Basic idea for the implementation is to use a max priority structure to find out the most popular hashtags.

Structures for the implementation:

1. Max Fibonacci heap: use to keep track of the frequencies of hashtags.
2. Hash table: The key for the hash table is the hashtag, and the value is the pointer to the corresponding node in the Fibonacci heap.

Programming Environment:

I implement this project in Java. This project has been tested on thunder.cise.ufl.edu and Intellij on local machine. To run the program, use the following command:

```
make
java hashtagcounter <input_file_name> [output_file_name]
```

to create an output file with the name out_file_name, which includes all output.

Structure of The Program

This project has 3 classes:

1. Node.class: This class is mean to create node type in Fibonacci heap

2. MaxFibonacciHeap.class: This class is to carry out all Max Fibonacci Heap operations, includes insert, remove max, increase key, pairwise, insert list at root level, delete node, get list of node.

3. hashtagcounter.class: The main method is included, it takes the input and output an output file.

Function Prototypes

Class: Node

Variables	Type	Description
degree	int	Degree of node
parent	Node	Parent of node
child	Node	The first child of node
left	Node	Left sibling of node
Right	Node	Right sibling of node
data	int	The key value of node (Frequency of the hashtag)
hashtag	String	Hashtag of node
childcut	boolean	Used for cascading cut

Nodes in same level are linked to form a circle. If there is only one node in Fibonacci Heap, its siblings are itself.

Function name	*public Node(int degree, Node parent, Node child, Node left, Node right, int data, String hashtag, boolean childCut)*
Description	Create a node type for Fibonacci Heap

Function name	*public int getDegree()*
Return type	int
Description	Return the degree of node

Function name	*public void setDegree(int degree)*
Return type	void
Description	Set the degree of node

Function name	*public Node getParent()*
Return type	Node
Description	Return the parent of node

Function name	*public void setParent(Node parent)*
Return type	void
Description	Set the parent of node

Function name	*public Node getChild()*
Return type	Node
Description	Return the first child of node

Function name	*public void setChild(Node child)*
Return type	void
Description	Set the first child of node

Function name	*public Node getLeft()/public Node getRight()*
Return type	Node
Description	Return the left sibling/ right sibling of node

Function name	*public void setLeft(Node left) /* *public void setRight(Node right)*
Return type	void
Description	Set the left and right sibling of node

Function name	*public int getData()*
Return type	int
Description	Return the data of node (frequency of hashtag)

Function name	*public void setData(int data)*
Return type	void
Description	Set the data of node

Function name	*public String getHashtag()*
Return type	String
Description	Return the hashtag of node

Function name	*public boolean getChildCut()*
Return type	boolean
Description	Return the childcut value

Function name	*public void setChildCut(boolean childCut)*
Return type	void
Description	Set the node childcut value

Class: MaxFibonacciHeap

Variable: private Node max
Description: the pointer to the node in Fibonacci Heap which has the largest data value(the most popular hashtag)

Function name	*public Node insert(Node node)*
Return type	Node
Parameter	Node: the node to insert
Description	Insert node into Fibonacci Heap and update the node information and max pointer. If the node to be inserted is bigger than max, do *insertBig* function and update max pointer, else, do *insertSmall* function.

Function name	*public Node removeMax(int queriesNumber)*
Return type	Node
Parameter	Integer: the number of queries in the input file which are detected currently.
Description	Remove current max node from Fibonacci Heap, Use function *pairwise* to update the Fibonacci Heap and return the max node before remove operation.

Function name	*public void increaseKey (Node node, int addend)*
Return type	Void
Parameter	Node: the node to be increased Addend: the addend to the node data
Description	Increase the node data value with addend number, update the Fibonacci Heap and max pointer

Function name	*private void pairWise(Node startNode, int queriesNumber)*
Return type	Void
Parameter	Node: the node where pairwise happens Integer: the number of queries in the input file which are detected currently
Description	Using Hash table to carry out pairwise operation from the start node. (In remove max operation, start node is first child of max) Using function *insertListOfNodesAtRootLevel* to update max pointer. Queries number is updated according to the number of queries in the input file.

Function name	*private Node insertListOfNodesAtRootLevel (List<Node> nodesList, Node next)*
Return type	Node
Parameter	List<Node>: the root level list to be inserted Node: the node to insert
Description	Insert Node next into root level list and return the max Node after insert operation.

Function name	*private Node deleteNode(Node node){*
Return type	Node
Parameter	Node: the node to be deleted
Description	Delete the node and update the sibling pointer of its siblings, and remove the deleted node parent pointer.

Function name	*private Node parentChild(Node parent, Node child)*
Return type	Node
Parameter	Node parent: the parent of node Node child: the child of node
Description	Update the parent and child pointer, and update the degree of parent.

Class: hashtagcounter

Variables	Type	Description
hashtagTable	Hashtable	The key for the hash table is the hashtag, and the value is the pointer to the corresponding node in the Fibonacci heap
maxFibonacciHeap	MaxFibonacciHeap	Max Fibonacci Heap structure

Function name	*public static void printResults(List<String> resultList)*
Return type	void
Parameter	List<String>: the output list of strings
Description	Print the output result

Function name	*public static List<String> readHashtagsAndQueries(String fileName)*
Return type	List<String>
Parameter	String: the name of the input file
Description	Take the input file and return a list of string with hashtag and queries.

Function name	*public void outQueries(List<String> input, PrintWriter output)*
Return type	void
Parameter	List<String>: a list of string with hashtag and queries PrintWriter: an output file
Description	Read the input file, and insert hashtags and frequency into hash table, update the corresponding max Fibonacci Heap. If query case happens, output the result in output format.

Function name	*public static void main(String[] args)*
Return type	void
Parameter	String args[0]: the name of the input file String args[1]: the name of the output file
Description	This is the main method. It takes the input file and output a file. In this function, args[0] represents the input file name, args[1] is output file name.