COP5536: Advanced Data Structures: Fall 2016

Project Report

Maithili Gokhale

maithilig@ufl

Project Description:

The main aim of the project is to give the top hashtags from a given dataset. This has been executed by implementing a Fibonacci Max Heap. A hashtable is maintained to keep the records of new and recurring hashtags, and then the Fibonacci Heap is implemented.

Compiling Instructions:

The project has been compiled and tested on thunder.cise.ufl.edu and Eclipse Mars.

To execute the program, one can remotely access the server using FileZilla.

On linux, get into the folder by command prompt, and then enter as follows

make –f Makefile

java HashtagCounter <input file name in double quotes> <output file name in double quotes>

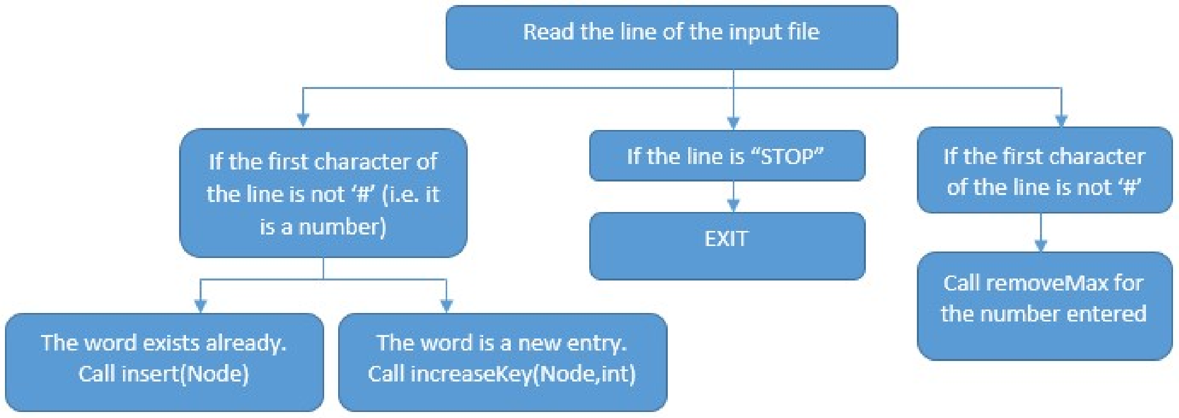
Make sure that the input file is in the same folder.

Program Structure:

The project has three classes:- “HashtagCounter.java”, “FibonacciHeap.java” and “Node.java”. Node.java” is an inner class of “FibonacciHeap.java”.

`HashtagCounter.java`

This class contains the main. It is where all the functions are basically called. We read the input file from here, and write into the output file. The hashtags are removed, and the word is stored as a key in a hashtable. The values are in the form of nodes. The following flow chart explains how the HashtagCounter.java class works. The functions are explained in detail later.


Read the line of the input file
If the first character of the line is not ‘#’ (i.e. it is a number)
If the line is “STOP”
If the first character of the line is not ‘#’
EXIT
Call removeMax for the number entered
The word exists already. Call insert(Node)
The word is a new entry. Call increaseKey(Node,int)

Flow of HashtagCounter.java

Node.java

This is an inner class of HashtagCounter.java.

Variables

The variables are as follows:

int frequency	It is the key value of the node. In our program, it is the frequency of the particular hashtag.
int degree	This variable is of type Integer and it signifies the number of nodes that a node can have in its next level.
boolean childCut	This variable is of type Boolean and it signifies whether or not a child has already been removed from that node. A childCut of false means that no child has ever been removed from that node.

The following elements are required for the doubly linked list structure of the Fibonacci Heap.

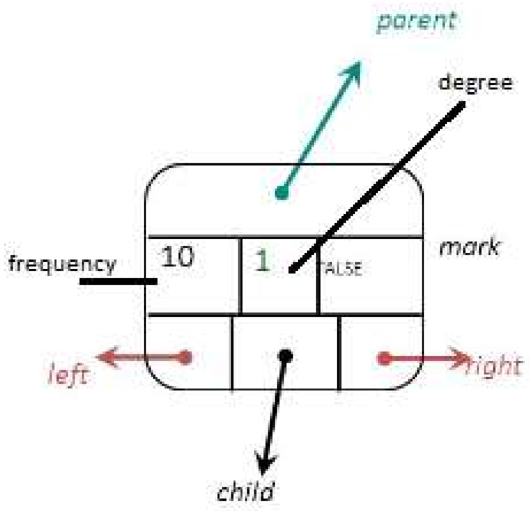
Node parent	Every node points to its parent node.
Node leftSibling	Every node has a left sibling.
Node rightSibling	Every node has a right sibling.
Node child	Every node points to one of its children. The one which has no child has it set to null.

The following image gives a basic idea of the structure of the node

Function Prototypes

`Node(int frequency)`

This is the constructor class, in order to create a sentinel node. It sets the degree to 0, the siblings to itself, and the child and parent as null.



Node structure

FibonacciHeap.java

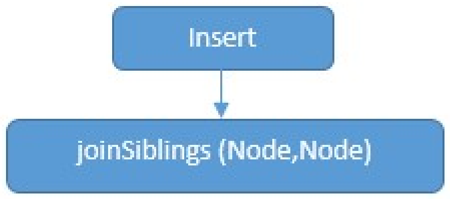
This class has all the main functions of a Max Fibonacci heap.

The variables are as follows:

Node maxNode	This stores the node with the maximum key (here, frequency).
int totalNodes	This stores the number of total nodes in the heap.

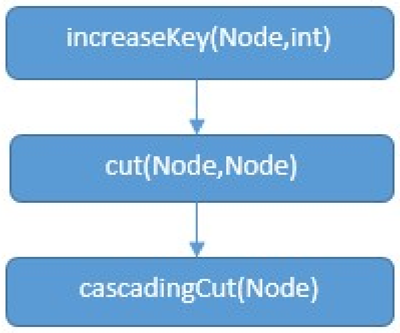
The flowchart of all the functions is given below:

1) Case 1: If the word is not in the hash table

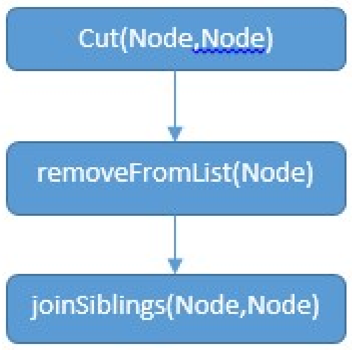


2) Case 2: If the word exists in the hash table

a) increaseKey(Node,int)

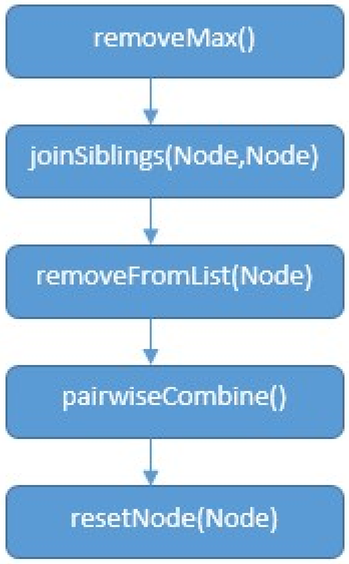


b) cut(Node,Node)

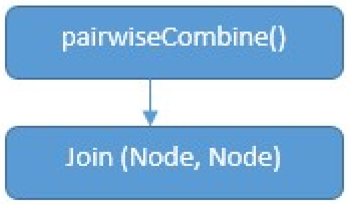


3) Case 3: Extract the max nodes (from given count)

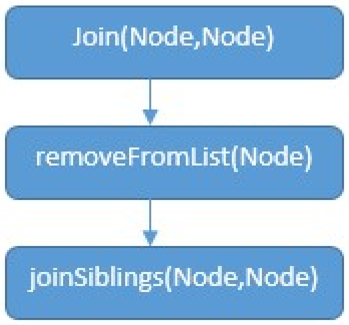
a) removeMax()



b) pairwiseCombine()



c) join(Node,Node)



Function prototypes

The function prototypes for HashtagCounter.java are as follows.

```
public FibonacciHeap()
```

This is the constructor class, in order to create a heap. It sets the maxNode to null, and the total number of nodes to 0.

```
public Node insert(Node insertNode)
```

Parameters: Node insertNode

Return type: Node

Description: The insertNode is the new node that has to be inserted in the heap.

```
public void increaseKey(Node node, int frequency)
```

Parameters: Node node, int frequency

Return type: void

Description: This function is to increase the frequency (key) of a particular node. The node, and the increased frequency are given as parameters. This function calls cut() and cascadingCut() later.

```
private void cut(Node x, Node y)
```

Parameters: Node x, Node y

Return type: void

Description: It is a function to remove the node x from below node y. If x is a child of the node y, then the node to the right of x is set as child of y. It is then added to the top most level to the right of maxNode.

```
private void cascadingCut(Node y)
```

Parameters: Node y

Return type: void

Description: The function of cascadingCut is to cut out of its sibling list in a remove or increase key operation. It follows path from parent of the Node to the root, and cuts the nodes. If the childcut is true, keep going up the tree and removing the nodes until it finds a node whose childcut is false.

`private void removeFromList(Node remNode)`

Parameters: Node remNode

Return type: void

Description: This function removes the node remNode from the sibling's doubly circular list, and points it's left and right sibling to each other.

`private void joinSiblings(Node max, Node n)`

Parameters: Node max, Node n

Return type: void

Description: This is used to join two circular doubly linked list of two siblings.

`public Node removeMax()`

Parameters: none

Return type: Node

Description: This function extracts the node with the maximum frequency (key). Since the maxNode is the root of the Fibonacci heap, on removing the root, we have to adjust the nodes on the next level of the root. We call the joinSiblings function to join the children with the nodes at maxNode’s level. The maxNode is then returned. This class calls the pairwiseCombine function to meld the subtrees with the same number of nodes, at the top most level. It also resets the removed max node by calling reset() function.

`void resetNode(Node n)`

Parameters: Node n

Return type: void

Description: It resets the values of the removed Max node. It sets the degree to 0, the siblings to itself, and the child and parent as null.

`private void pairwiseCombine()`

Parameters: none

Return type: Node

Description: It is called after the maxNode is removed. If there are two nodes with the same degree, it melds the two subtrees by attaching the subtree with larger root value as the child of the subtree of the smaller tree value and adjust maxNode. It calls the join() function to join two roots of the same degree.

```
private void join(Node y, Node x)
```

Parameters: Node y, Node x

Return type: void

Description: This function is called by the pairwiseCombine() function when it wants to join two nodes. It links y under x. 'y' is the smaller of the two, and it is set as the child of x.

Conclusion

HashtagCounter successfully takes note of all the hashtag words (new and repeating). Upon hitting a number, those number of top most hashtags are extracted successfully by implementing Max Fibonacci Heap.