Advances Data Structures (COP 5536)
Fall 2018
Project Report

Name: **Eesh Kant Joshi**
UFID: **1010-1069**
Email ID: **eesh.joshi@ufl.edu**

1. Problem Statement

There is a new search engine “DuckDuckGo” which implements a system to count the most popular keywords used in this search engine. We want to know the n most popular keywords are at any given time. The keywords together with their frequencies are given from an input file. We have to use a max priority structure to find the most popular keywords.

We have used the following data structures:

1. **Max Fibonacci heap:** Keeps track of the frequencies of keywords
2. **Hash table:** The keywords are used as keys for the hash table and value field points to the corresponding node in the Fibonacci heap.

2. Implementation Details

The project consists of 3 .java files:

1. FibonacciHeapNode.java
2. DuckDuckGo.java
3. FibonacciHeapDataStructure.java

2.1 FibonacciHeapNode.java

This file contains the FibonacciHeapNode class which is used for instantiating a node. The node has properties such as **parent, left, right, child, degree, childCut, frequency** and **keyword**. The node properties get initialized with the default values through a constructor. Two getter methods are used for getting the frequency and keyword values of the node. The Node class is public.

- **parent:** Stores the reference of the parent of a particular node. It is of type Node.
- **left:** Stores the reference of the left sibling of a particular node. It is of type Node.
- **right:** Stores the reference of the right sibling of a particular node. It is of type Node.
- **child:** Stores the reference of the child of a particular node. It is of type Node.
- **degree:** Stores the number of children a particular node has. It is of type Integer.
- **childCut:** It is of Boolean type and it stores whether a child has already been removed from a particular node or not. True means that a child had already been removed previously. False means that no child was removed previously from the particular node.
- **frequency:** It signifies the number of times a particular string has appeared in the search engine. It is of type Integer.
- **keyword:** It is the string which is entered in the search engine and corresponds to a particular node in Fibonacci heap. It of type String.

2.2 DuckDuckGo.java

This class contains the Main method from where the execution starts. In the command line arguments, we have provided the input file name from which we take the input. We read the inputs and write the desired output in the output file.

2.3 FibonacciHeapDataStructure.java

This file contains FibonacciHeapDataStructure class which is used for instantiating a Fibonacci heap. The FibonacciHeapDataStructure class contains 7 functions:

1. insertKeyword()
2. increaseKeyValue()
3. detachNode()
4. triggerCascadeCut()
5. removeMaxNode()
6. makeChild()
7. treePairWiseMerging()

public void insertKeyword(String keyword, int frequency)		
Description	Inserts the keyword associated with the node and its frequency in the Fibonacci Heap	
Parameters	Keyword	The keyword associated with the node
	Frequency	The frequency associated with the keyword
Return value	Void	

public void increaseKeyValue(FibonacciHeapNode node, int frequency)		
Description	Increases the frequency of the keyword in the Fibonacci Heap	
Parameters	Node	The node associated with the keyword
	Frequency	The frequency value by which the frequency of the keyword has to be increased
Return value	Void	

public void detachNode(FibonacciHeapNode childNode, FibonacciHeapNode parentNode)		
Description	Detaches the childNode from the parentNode and places the childNode in the root level circular DLL	
Parameters	childNode	The node which has to be detached
	parentNode	The node from which the detachment has to be done
Return value	Void	

public void triggerCascadeCut(FibonacciHeapNode node)		
Description	Recursively updates the childCut values of the parent node from which a child has been detached	
Parameters	Node	The node from which a child has been detached
Return value	Void	

public FibonacciHeapNode removeMaxNode()		
Description	Removes and returns the maximum node from the Fibonacci Heap	
Parameters	None	
Return value	Returns the reference of the node with maximum frequency which has been removed	

public void makeChild(FibonacciHeapNode childNode, FibonacciHeapNode parentNode)		
Description	Makes one node child of the other node and updates all the pointers	
Parameters	childNode	The node which has to be made the child
	parentNode	The node which has to be made the parent
Return value	Void	

public void treePairWiseMerging()		
Description	Performs a pairwise merge on the max trees in the Fibonacci Heap	
Parameters	childNode	The node which has to be made the child
	parentNode	The node which has to be made the parent
Return value	Void	

3. Input format

Keywords appear one per each line in the input file and starts with $ sign. In each line, an integer will appear after the keyword and that is the count (frequency) of the keyword (There is a space between keyword and the integer). You need to increment the keyword by that count. Queries will also appear in the input file and once a query (for most popular keywords) appears, you should append the output to the output file. Query will be an integer number (n) without $ sign in the beginning. You should write the top most n keyword to the output file. When "stop" (without $ sign) appears in the input stream, program should end. Following is an example of an input file.

$facebook 5
$youtube 3
$facebook 10
$amazon 2
$gmail 4
$weather 2
$facebook 6
$youtube 8
$ebay 2
$news 2
$facebook 12
$youtube 11
$amazon 6 3
$facebook 12
$amazon 2
$stop 3
$playing 4
$gmail 15
$drawing 3
$ebay 12
$netflix 6
$cnn 5
5
stop

4. Output format

Once a query appears, you need to write down the most popular keywords to the output file in descending order. Output for a query should be comma separated list without any new lines. Once the output for a query is finished you should put a new line to write the output for the next query. You should produce all the outputs in the output file named “output_file.txt”. Following is the output file for the above input file.

facebook,youtube,amazon facebook,youtube,gmail,ebay,amazon

5. Execution details

- The project has been compiled and tested on local machine and thunder.cise.ufl.edujava.
- Extract the zip file contents.
- Type ‘make’ (without quotes).
- Type ‘java DuckDuckGo file_name’ (without quotes).