Advances Data Structures (COP 5536)

Programming Project Report

Author Information:

Name: Surya Sudharhsan
UFID: 5019-3163
Email: surya.sudharshan@ufl.edu

Project Objective:

The goal of this project is to count the *n* most popular search words or keywords used in the search engine *"DuckDuckGo"* at any time.
The project is mandated to use the following data structures:

- **Max Fibonacci heap**: to keep track of the frequencies of keywords
- **Hash table**: keywords should be used as keys for the hash table and value is the pointer to the corresponding node in the Fibonacci heap.

Project Implementation:

I have implemented the project in JAVA. I have implemented the Max Fibonacci Heap to store the data of all the search words and their frequencies. I have also implemented the Hash Table data structure provided by the Java libraries to store the information of nodes in a key-value relationship.

Project Structure and Description:

The project is modularized into 3 files/classes.

1. **Node**- This class consists of the node structure and parameters of the nodes of the Fibonacci heap
2. **FibonacciHeap**- This class consists of all the standard operations of a Max Fibonacci Heap.
3. **PopularKeywords**- This class consists of the runner program that takes the input and write the output. It also implements the logical flow of the program.

The workflow of the program is that the main class from popularkeywords reads

the input file and inserts nodes into the Fibonacci heap according to the input. When it encounters a number in the input file, it performs a RemoveMax() operation as many times as the value denoted by the number and outputs that many search words in descending order of popularity. After this is done, it reinserts the nodes back into the heap as further lines of input may contain the same keywords and the previous frequencies also need to be considered for calculating the most popular keywords. The program terminates the moment it encounters a "stop" in the input file. The output is in a file created by the program called "Output_file.txt".

The prototypes of all the functions and classes used in the project are described below:

Node.Java

FunctionName(Parameters)	**Return Type**	**Description**
1. Node(String keyword, int key)	Void	Constructor that initializes the parameters of the node structure.
2. public String getKeyword()	String	Getter function to retrieve the keyword stored in a node.

FibonacciHeap.Java

FunctionName(Parameters)	**Return Type**	**Description**
1. public void insert(Node node)	Void	This function is used to insert a node into the Fibonacci heap. The function works in two ways. If the max is null i.e. if the maxNode is null, maxNode is assigned to be the root of the Fibonacci heap since it is the only node in the Fibonacci heap right now. However, if there is a maxNode , **node** is added to the top level of the Fibonacci heap, to the right of the max root in the doubly linked list of that level. It increases the node_count by 1
2. public void cut(Node child, Node parent)	Void	This function performs cut operation on Node **parent**. It cuts Node **child** from Node **parent** and also decreases its degree.

3. public void cascadingCut(Node target)	Void	This function checks the child_cut value and makes the necessary changes and performs remove if needed. If the child_cut value is false for the parent, make it true. If the child_cut is true, keep going up the tree and removing the nodes until it finds a node whose child_cut is false. It calls itself recursively till the node has a parent
4. public void increaseKey(Node target, int value)	Void	This function is used to increase the value of the **key** to value in Node **target**. The key value of parent is checked, and if it is less than the parent the node is cut and cascading cut is performed on the parent if the parent child_cut is true.
5. public Node removeMax()	Node	This function removes the maximum node, from the Fibonacci heap. And while there are children of max node, put them on root and then call degreewisemerge().
6. public void degreewiseMerge()	Void	This function performs degreewisemerge. It melds the nodes in the root list of the heap having same degrees by storing degrees of all roots in a degree table and combining them pair-wise.
7. public void makeChild(Node child, Node parent)	Void	This function is used to make Node **child** a child of Node **parent**.

Compiling and Running Instructions

The project has been compiled and tested on the thunder.cise.ufl.edu and java compiler on my local machine.

To execute the program,

Access the server using ssh and perform these steps.

For executing the program PopularKeywords.java

- Extract the contents of the zip file.
- Type ‘make’ without the quotes.
- Type ‘java PopularKeywords.java ‘file_path/inputfile_name.txt’’ without the quotes and add the file_path and inputfile_name.txt in the command above.

Results

Test Cases:

1. Sample1.txt – Input file given in the problem statement in the project description file.
2. Sample2.txt – Input file with around 500,000 keywords.
3. Sample3.txt – Input file with around a million+ keywords.
4. Sample4.txt – Input file with query greater than the number of keywords.
5. Sample5.txt – Input file with more than one “stop” statement. The program terminates when the first “stop” is encountered.

Outputs:

All output files for the above input test cases are available in the directory /Sudharshan_Surya/outputs.

Conclusion:

The program was run for input files ranging from 20 to a million in size. The time taken was found to be at the most logarithmic of the number of nodes and the performance was quite fast. Max Fibonacci Heap is a very efficient data structure to store and process million of data items as it has logarithmic performance.