

Advance Data Structures (COP 5536)

FALL 2018

Programming Project Report

ROHIT

DEVULAPALLI

UFID: 4787-4434

rohitdevulapalli@ufl.edu

PROJECT DESCRIPTION

The goal of this project is to implement a system to find the n most popular keywords that appear on the search engine DuckDuckGo. For the scope of this project, keywords are taken as an input file. Basic idea for the implementation is to use a max priority structure to find out the most popular keywords.

The project uses the following data structure.

1. Max Fibonacci heap: Use to keep track of the frequencies of Keywords.
2. Hash table(Hash Map in java) : Key for the hash table is keyword and value is pointer to the corresponding node in the Fibonacci heap.

The project is written in JAVA. I have implemented Max Fibonacci Heap in java and stored the address of all the nodes in a Hash Map (Built in Data Structure). I have written the project using JAVA without any external in-built data structure. Max Fibonacci heap is required because it has better theoretical bounds for increase key operation.

WORKING ENVIRONMENT

HARDWARE REQUIREMENT

Hard Disk space: 4 GB minimum

Memory: 512 MB

CPU: x86

OPERATING SYSTEM

LINUX/UNIX/MAC OS (If using other OS make command might not work)

COMPILER

Javac

COMPILING & RUNNING INSTRUCTIONS

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

To execute the program,

You can remotely access the server using ssh
username@thunder.cise.ufl.edu

For running the Key Word Counter

1) Extract the contents of the zip file

2) Type ‘make’ without the quotes.

3) Type ‘java keywordcounter ‘file

path/input_file_name.txt’ ’ without the quotes

and add the file name and path. I have included

the file input1.txt .

STRUCTURE OF THE PROGRAM AND FUNCTION DESCRIPTIONS

The program consists of a keywordcounter java file inside which there exists 3 classes namely.

- 1) keywordcounter - The main class that reads and writes the output.
- 2) Nodes – This class is used to instantiate and object of node in memory.
- 3) MaxFibonacciHeap – This class is used to instantiate the methods and functions of the Fibonacci Heap class.

The basic workflow of the program is the keywordcounter.class takes input file and reads it creates a hash map and performs insert node operations on an instance of the MaxFibonacciheap.class by creating new nodes using Nodes.class. It performs remove max as soon as it finds a single digit in the input file. And then prints it in the output_file.txt, after which it reinserts the node back into the hashmap.

The detailed overview of all the class functions is given below.

keywordcounter.java

It contains three classes within it namely keywordcounter, Nodes and MaxFibonacciHeap

keywordcounter class

The Class Variables of keywordcounter are

path: Input file name taken from the user. Type is String.

hm: The HashMap where Node and keyWords are stored. Type is HashMap<String,Nodes>.

fh: Max Fibonacci Heap instance. Type is MaxFibonacciHeap.

file: Name of the output file. Type is String.

pa1: It contains the format of the input that needs to be given. Format is "([\$])([a-z_]+)([\\s](#))([\\d+](#))". Type is Pattern.

pa2: Contains the pattern of the digits to be removed from the keyword. Type is Pattern.

ma1: Matcher object for pattern pa1. Type is Matcher.

ma2: Matcher object for pattern pa2. Type is Matcher.

keyWord1: Keyword to be stored. Type is String.

key: Key value to be stored in node. Type is Integer.

increasingKey: Contains sum of old key value and new key value. Type is Integer.

removeNum: Contains number of nodes to be removed. Type is Integer.

remNodes: Stores all the removed nodes. Type is ArrayList<Nodes>.

Nodes Class

The Class Variables of Nodes are

degree: Contains the number of nodes that a node can have in its next level. Type is Integer.

keyWord: Contains the Key Word. Type is String.

key: Signifies the value of an integer. Type is Integer.

left: Points to the left node in the circular list. Type is Nodes.

right: Points to the right node in the circular list. Type is Nodes.

parent: Points to the parent node. Type is Nodes.

child. Points to the child node. Type is Nodes.

mark: If the mark value is false, it means that no child has ever been removed from that node.

Functions present in this class are

Function Name	Return Type	Parameters
1. Node(String keyWord, int key)	-	String keyWord, int key
2. public String getKeyWord()	String	-

Function Name	Description
1. Node(String keyWord, int key)	Constructor that initializes the keyword and key.
2. public String getKeyWord()	Returns the keyWord of that node.

MaxFibonacciHeap Class

The Class variables are

numOfNodes: Contains the number of nodes that are stored in the heap. Type is Integer.

maximumNode: Points to the maximum node in the max Fibonacci heap. Type is Nodes.

Functions present in this class are:

Function Name	Return Type	Parameters
public void nodeInsert(Nodes node)	Void	Nodes node
public void increasingKey(Nodes a, int k)	Void	Nodes a , int k
public Nodes removeMax()	Nodes	Null
public void cut(Nodes a, Nodes b)	Void	Nodes a , Nodes b
public void cascadingCut(Nodes b)	Void	Nodes b
public void degreewiseMerging()	Void	Null
public void makeChild(Nodes b, Nodes a)	Void	Nodes b, Nodes a

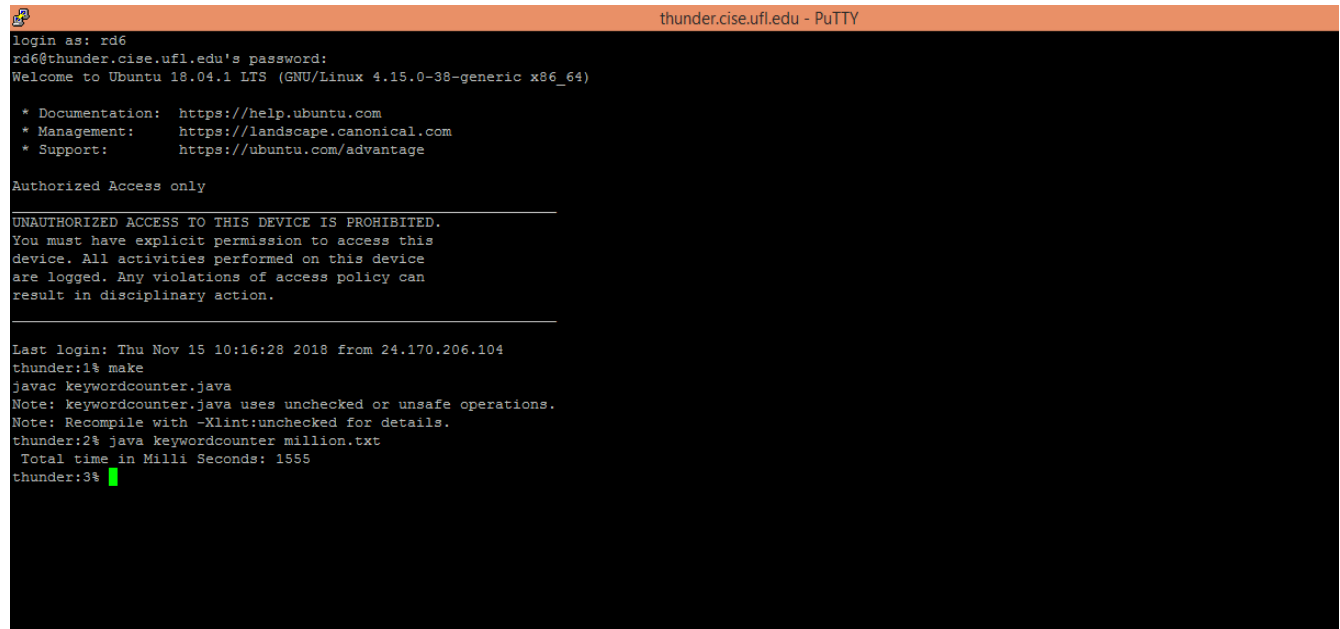
Function Name	Description
1. public void nodeInsert(Nodes node)	This function inserts a node into the max Fibonacci heap. The function works in two ways. If the max is null i.e if the maximumNode is null, maximumNode is assigned to be the root of the Max Fibonacci heap since it is the only node in the Fibonacci heap right now. However, if there is a maximumNode , myNode is added to the top level of the Max Fibonacci heap, to the right of the max root in the doubly linked list of that level. It increases the numOfNodes by 1.
2. public void increasingKey(Nodes a, int k)	This function is used to increase the value of the key to k in Node a. 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 childCut is true.
3. public Node removeMax()	This function removes the maximum node from the Max Fibonacci heap. And while there are children of max node, put them on root and then call degreewiseMerging().
4. public void cut(Nodes a, Nodes b)	This function performs cut operation on Node b. It cuts Node a from Node b, then it decreases the degree of that Node.
5. public void cascadingCut(Nodes b)	This function checks the make value and makes the required changes and performs remove if needed. If the make (childcut) value is false for the parent, make it true. If the make (childcut) is true, keep going up the tree thereby removing the nodes until it finds a node whose make (childcut) is false. It calls itself recursively till the node has a parent.

6. <code>public void degreewiseMerging()</code>	This function performs degreewisemerging. It melds the nodes in the root list of the Max Fibonacci heap.
7. <code>public void makeChild(Nodes b, Nodes a)</code>	This function is used to make Node 'b' a child of Node 'a'.

RESULTS

The code was run for 1 million inputs files and the output time was found to be 1555ms. The output file containing most popular keywords is successfully generated when a sample input file is being given as input. The order of the output can be different as there are many nodes with same values when remove max was called.

MAKE FILE EXECUTION AND PROGRAM RUNNING



```
thunder.cise.ufl.edu - PuTTY
login as: rd6
rd6@thunder.cise.ufl.edu's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-38-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Authorized Access only

UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED.
You must have explicit permission to access this
device. All activities performed on this device
are logged. Any violations of access policy can
result in disciplinary action.

Last login: Thu Nov 15 10:16:28 2018 from 24.170.206.104
thunder:1% make
javac keywordcounter.java
Note: keywordcounter.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
thunder:2% java keywordcounter million.txt
Total time in Milli Seconds: 1555
thunder:3% █
```

CONCLUSION

The objective of this assignment has been met. The program successfully implemented Max Fibonacci Heap as well as its removeMax and Increase Key operations which helped in finding most popular keywords.