

Advanced Data Structures (COP5536)

Fall 2018

Project Report

Details:

Shivaditya Jatar

UFID: 6203-9241

shivadityajatar@ufl.edu

Introduction

We have implemented a system to count n most searched/popular keywords used in a search engine named “DuckDuckGo” at any given time. For this, we built Max Fibonacci Heap and Hash Table with basic data structures to perform the function.

To implement the above system, I have written three classes:

- First one is keywordcounter which performs reading operation from input file consisting of keywords along with their frequencies and Query number.
- Second is the MaxFibonacciHeap which has got all the needed functionalities for the maximum Fibonacci heap to perform its operations.
- Third is the DDGNode (DDG - DuckDuckGo) which stores all the attributes of the node in the Max Fibonacci heap.

Compilation and Running

- First of all Unzip the file.
- For compilation use ‘make’ command.
- After compilation, keywordcounter.class, MaxFibonacciHeap.class and DDGNode.class will be generated.
- To execute program for given input file, type: java keywordcounter inputFilename.txt
- File named ‘output_file.txt’ with the output will be generated in the present directory.

(Before running, Goto Jatar_Shivaditya directory, where all the Java files are present)

\$ cd Jatar_Shivaditya

\$ make

\$ java keywordcounter inputFilename.txt

Structure of the Program

1. Class DDGNode (DDGNode.java)

Contains the definitions and constructors to create a node for the Fibonacci Tree.

We have eight **instance variables** defined in this class:

DDGNode parent: Stores the parent of the node.

DDGNode child: Stores the child of the node.

DDGNode prev: Stores previous (left) sibling of the node.

DDGNode next: Stores next (right) sibling of the node.

int key: Stores the node value.

int degree: Stores the number of children that a node has.

boolean childCut: Stores Child Cut value for a node. (whether a child of the node is deleted)

String keyword: Stores the Keyword.

The class contains the constructor:

- **public DDGNode(int key, String keyword)**
Used to initialize a node.

2. Class MaxFibonacciHeap (MaxFibonacciHeap.java)

Contains some basic Max-Fibonacci heap methods needed to perform the desired functionalities.

We have two **instance variables** defined in this class:

int numOfNodes: Keeps track of the total number of nodes.

DDGNode maxNode: Stores the maximum node of the Fibonacci tree.

Methods

public void insert(DDGNode ddgnode)

Description: Used to insert a new node in max Fibonacci heap. If the key of new node is larger than previous max node, then, update the max node to the new node and increment the total number of nodes.

Parameters: DDGNode ddgnode

Return value: void

public void removeMax()

Description: Used to remove the maximum node in heap and add the children of the maximum node to the root (tree) list. After this operation, we need to perform degree wise Merge. At the end, we need to decrease the total number of nodes.

Parameters: none

Return value: void

void degreewiseMerge()

Description: Used to perform degree wise combine to form the heap structure again after removeMax().

Brief Working: We initialized a new ArrayList that contains the nodes.

Firstly, if the degree of current node equals to the degree of node in the ArrayList, we need to link them and add the max node of new tree in the ArrayList.

Secondly, we check if the nodes in the ArrayList have the same degree. If same degree, we combine them together. Finally, we need to find the next (right) sibling of current node, and perform same steps as listed above.

Parameters: none

Return value: void

void cut(DDGNode p, DDGNode q)

Description: It cuts the node p from node q. After cut, it joins p to the root.

Parameters: DDGNode p, DDGNode q

Return value: void

void cascadeCut(DDGNode ddgnode)

Description: It cuts the nodes of the parents as well, if their childCut value is true from the heap.

Brief Working: This method is used to perform cascading cut operation in Fibonacci heap. If its parent did not loose child before (childCut = false), we just need to cut the

node, and set childCut value to true. If its parent lost child before (childCut = true), we perform keep on cutting till we find childCut value is false or till we reach till root.

Parameters: DDGNode ddgnode

Return value: void

public void increaseKey(DDGNode p, int q)

Description: Increases the value of the node in Fibonacci heap.

Brief Working: Cascading cut function is called if the increased node exceeds the value of the parent.

Parameters: DDGNode p, int q

Return value: void

public int fetchMaxKey()

Description: Used to fetch the key of maximum node in the Fibonacci heap.

Parameters: none

Return value: maxNode.key (int)

public DDGNode fetchMaxDDGNode()

Description: Used to fetch the maximum node in the Fibonacci heap.

Parameters: none

Return value: maxNode (DDGNode)

3. Class keywordcounter (keywordcounter.java)

Reads the input file, call the methods in MaxFibonacciHeap to calculate the frequency of keywords and prints the result ('n' most searched keywords) in 'output_file.txt'.

Methods

public static void main (String args[])

Description: Program flow starts from here. The argument is the input file. It reads in the file and makes calls to build Fibonacci heap.

Brief Working:

- (i) Firstly, Store the file name in a string and create a Hash Table and a Max Fibonacci heap.
- (ii) Read the input file line by line.
 - If string equals "STOP", just stop the program. If the first character equals "\$" and the "\$keyword" string is not present in the Hash Table, then insert "\$keyword" and frequency as a node in Max Fibonacci heap.
 - If the first character equals "\$" and "\$keyword" exists in the Hash Table, we perform increaseKey operation (increase the frequency).

- If first character is not equal to “\$”, and is a number, then we resolve the query string by performing appropriate operations of Hash Table and Max Fibonacci heap.
- (iii) Before going any further, We reinsert the node which we had removed.
- (iv) Now, continue with the next line of the input and do step (ii) again.
- (v) Finally, the output can be seen in the file “output_file.txt”

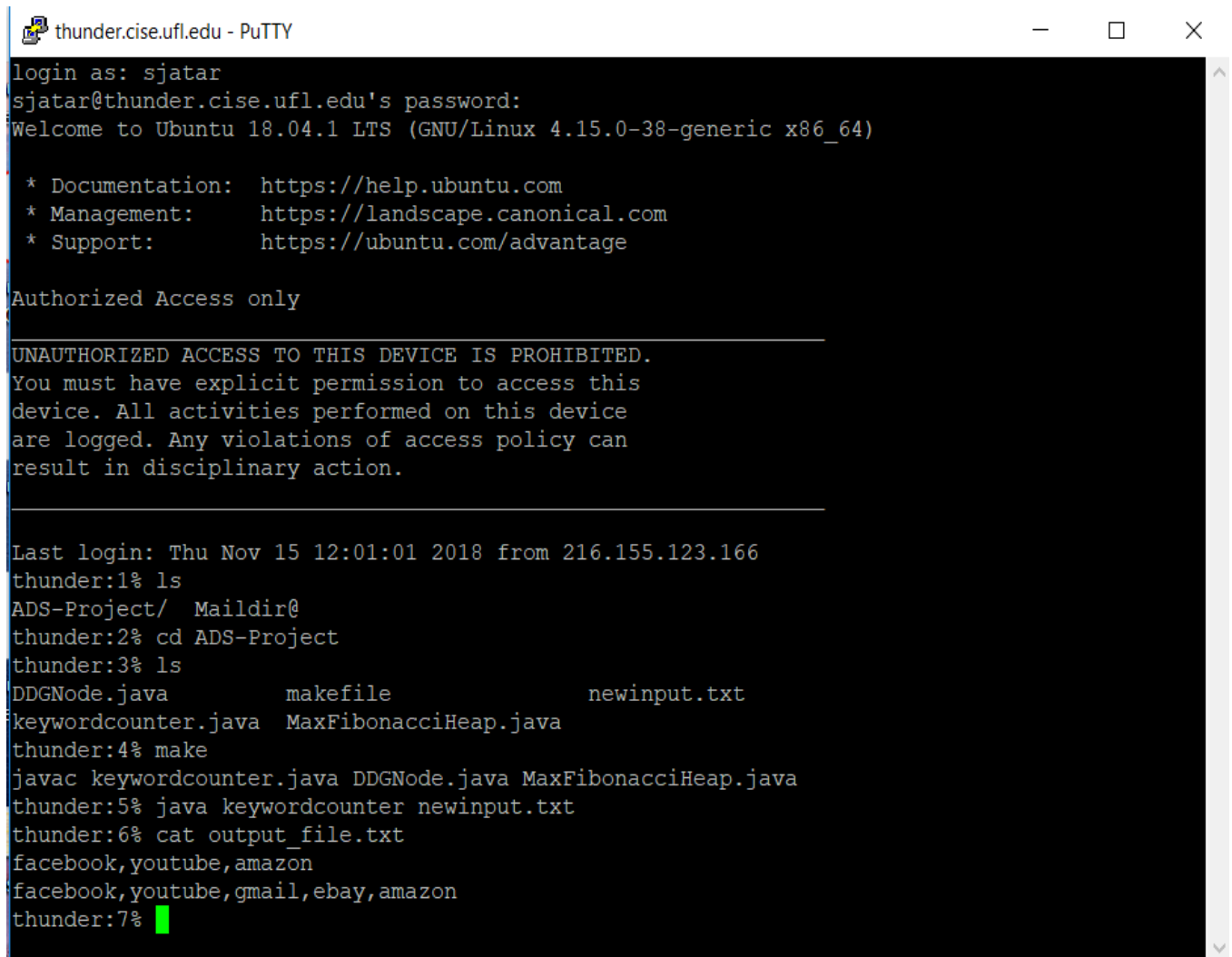
Parameters: String args[]

Return value: void

Sample Results

\$ make

\$ java keywordcounter newinput.txt



```
thunder.cise.ufl.edu - PuTTY
login as: sjatar
sjatar@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 12:01:01 2018 from 216.155.123.166
thunder:1% ls
ADS-Project/  Maildir@
thunder:2% cd ADS-Project
thunder:3% ls
DDGNode.java      makefile          newinput.txt
keywordcounter.java  MaxFibonacciHeap.java
thunder:4% make
javac keywordcounter.java DDGNode.java MaxFibonacciHeap.java
thunder:5% java keywordcounter newinput.txt
thunder:6% cat output_file.txt
facebook,youtube,amazon
facebook,youtube,gmail,ebay,amazon
thunder:7% █
```