Advanced Data Structures(COP 5536)

Fall 2016

Programming Project Report

VAIBHAV SOMANI

UFID: 2945-9679

vaibhavsomani@ufl.edu

PROJECT DESCRIPTION

The project consists of finding the n most popular hashtags in the social media. The hashtags will be given from a input file. The implementation needs to be done using a max priority structure.

We use a maximum Fibonacci Heap to keep the frequencies of hashtags. Also, a hash table is used. The key for the hash table is the hashtag and value is the corresponding node in the Fibonacci heap. The idea is to remove the max element from the top, store it in a list and re insert it again.

WORKING ENVIRONMENT

HARDWARE REQUIREMENT

Hard Disk space: 1 GB minimum

Memory: 512 MB

CPU: x64

OPERATING SYSTEM

Windows 8.1 and higher

COMPILER

Javac

COMPILING INSTRUCTIONS

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

To execute the program,

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

For running Max Fibonacci Heap algorithm on an input file, type

make

After the program is compiled, type
java hashtagcounter

Now, the user will be asked to input the file name containing input.
Example:
Enter a file name: **input.txt**

STRUCTURE OF THE PROGRAM AND FUNCTION DESCRIPTIONS

There are 3 classes that I have used to implement the programming assignment. The 3 classes are ‘Fibonaccinode.java’, ‘hashtagcounter.java’ and ‘Fheap.java’.

Here’s how the functions are defined in each class and a short description of how they work.

Fibonaccinode.java

The class FibonacciHeapNode mainly describes the structure of a node that is used in a Fibonacci heap.

The class variables are:

- degree

This variable is of type Integer and it signifies the number of nodes that a node can have in its next level.

- mark

This variable is of type Boolean and it signifies whether or not a child has already been removed from that node. A mark of false means that no child has ever been removed from that node.

- The Fibonaccinode class further has objects parent, child, right, left and key. Right and left are used for the doubly linked list that is created at each level. Key stores the value of the Fibonacci node.

- A constructor of this class is created and left, right and parent is initialized to null, degree to zero. Hash and Key values are passed as parameters.

hashtagcounter.java

The class hashtagcounter is where the hashing is defined. The main function is defined inside this class.
Initially, we take an input using scanner class from the user in the form of a file. Then, the hashmap is defined and a file to collect the output is made. File reader is used to read the stream of characters in input. Then, the pattern class is used to make character of different types. Then, matcher function matches the character sequences against the regular expressions. Store the alphabets in the data field and store the digit value as key. If similar objects are found, increase the key by the digit value. To write the output to the file, remove the max elements and store them in a list and reinsert them again.

Fheap.java

This class contains all the Fibonacci heap operations. Two instance variables max node and nNodes are defined. The following functions are defined:

- Public void insert(Fibonaccinode node)
  In this function, the new node is to the right of the max node and the circular doubly linked list is modified. If the inserted node is maximum, it becomes the maxnode. If the maxnode was null, the inserted node becomes the maxnode.

- Public void increasekey(Fibonaccinode m, int k)
  Initialize the key of m to be k. Iniltialize b to be parent of m. If b is not null and key of m is greater than key of b, perform cut(m,b) and perform cascadingcut(b). If key of m is greater than key of maxnode, initialize max node to be m.

- Public Fibonaccinode removemax()
  In this function, for each child a of q, remove a from the childlist and a to the rootlist of heap and perform consolidate(). Set the parent of removed file to be null and decrease the number of kids.

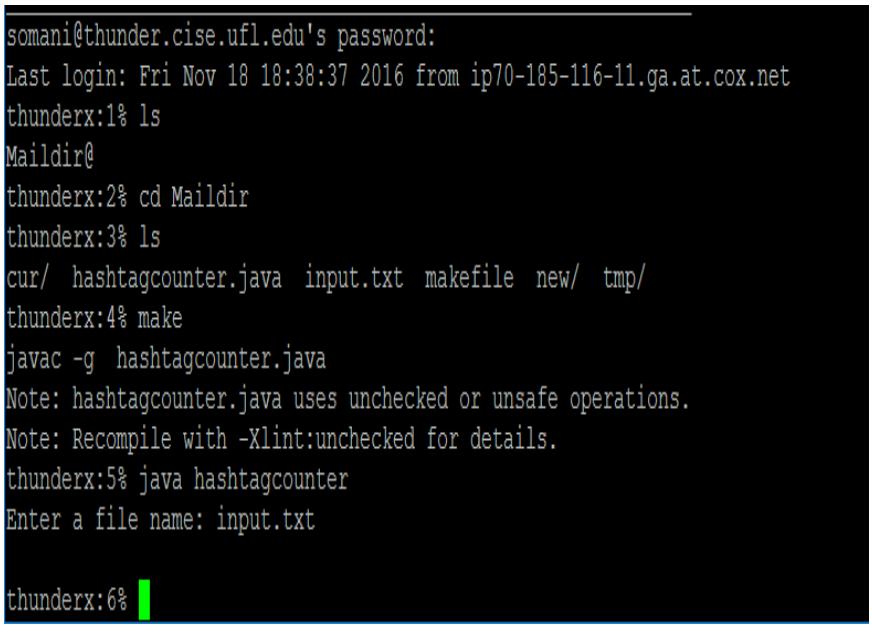
- Public void cascadingcut(Fibonaccinode s)
  Set w to be the parent of s. If w is not null, check the mark of s. If it is false, set it to true. If it is true, it has to be cut and repeat the cascadingcut(w).

- Public void cut(Fibonaccinode p, Fibonaccinode q)
  Remove p from the child list and decrement the degree of q. Reset the value of q.child. Now, add p to the root list. Set the parent of p to null and make the mark of p to be false.

- Public void consolidate()
  Create an array to keep track of roots according to their degrees. No two trees should be of same degrees. Find two roots q and p of same degree. Without loss of generality, q.key > p.key. Perform link(p,q). Repeat until there are no two trees of same degree and decrease the number of roots.

- Public void link(Fibonaccinode a, Fibonaccinode b)
  Remove b from the root list and make b a child of a. Increase the degree of a set the mark value of b as false.

RUNNING THE PROGRAMS



CONCLUSION

The program is successfully implemented using maximum Fibonacci heap.