Nicholas Kroeger

**Program structure**

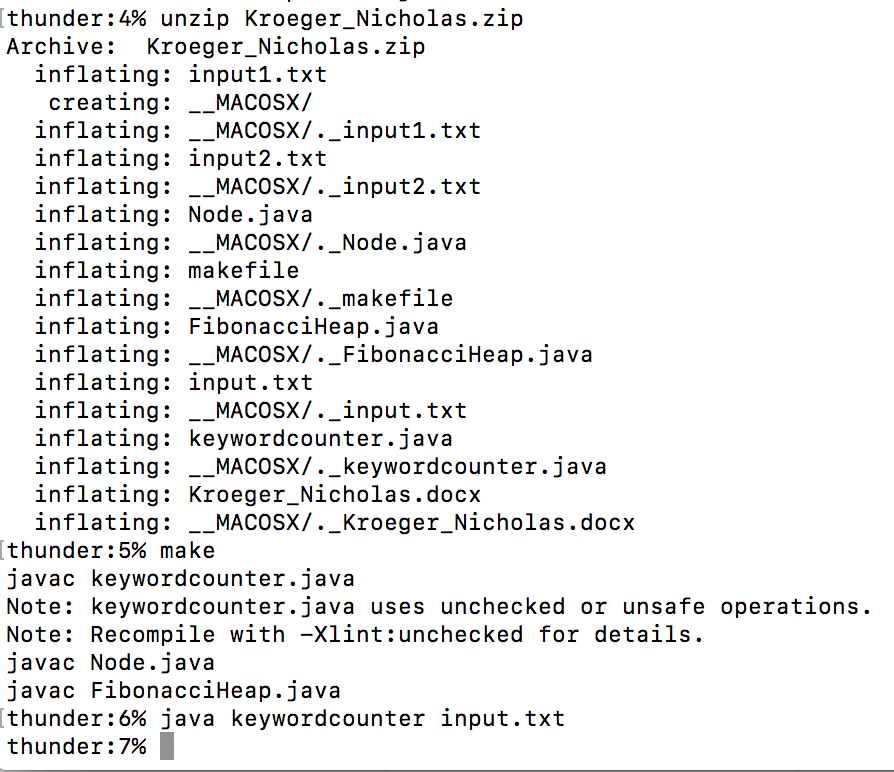
Classes: keywordcounter.java, FibonacciHeap.java, Node.java

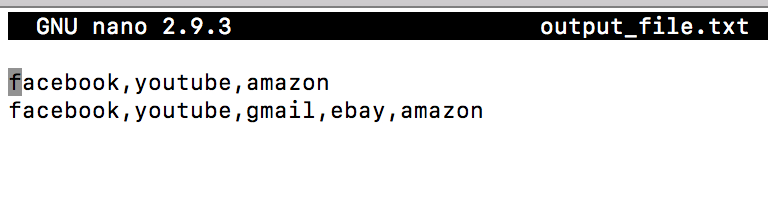
The keyword counter function has the main which acts as the entry point with the command prompt by reading in the input text file name. The main initializes the hashmap and the Fibonacci heap. As more lines are read from the textfile, the frequencies are stored in the heap and the keyword are keys in the hashmap and the node in the heap associated with that keyword is stored in the value part of the hashmap. It also handles the writing out to a text file when a query is performed.

The Fibonacci heap maintains all functionality of the data structure such as insert, removeMax, increaseKey, pairwiseCombine, and cascading cut.

The Node file is a class to hold properties of each node in the Fibonacci heap such as degree, children, next and previous nodes in their respective level list.

**Proof of running on thunder.cise.ufl.edu**





**keywordcounter.java**

void main(String[] args);

|  |  |  |
| --- | --- | --- |
| **Description** | Controls the entire program. Read in text file input, builds heap structure, outputs answer. | |
| **Parameters** | String[] args | String of the name pointing to the input textfile |
| **Return Value** | void | |

void writeTopNkeywords(HashMap<String,Node> hashMap, FibonacciHeap fibHeap, String firstString, BufferedWriter writer);

|  |  |  |
| --- | --- | --- |
| **Description** | This function is triggered on a query to print to an output file, the top N words | |
| **Parameters** | hashMap  fibHeap  firstString  writer | Stores keyword and frequency in hash map  Used to store all frequencies of keywords  A String from one individual line in the input file  Object used to write to the output file |
| **Return Value** | void | |

The remaining functions in keywordcounter.java serve as helper functions and functions created for the purpose of maintaining code readability by ensuring that functions have primarily one purpose. These remaining functions are small in purpose and quantity of code, thus should be readable from a quick inspection.

**Node.java**

Node(String keyword, int frequence);

|  |  |  |
| --- | --- | --- |
| **Description** | A node structure functions as an object that will live within a Fibonacci heap | |
| **Parameters** | Keyword  frequency | A keyword that was searched using duckduckgo  The amount of time the keyword appears total in searches. |
| **Return Value** | NA, it’s a constructor | |

**FibonacciHeap.java**

FibonacciHeap();

|  |  |  |
| --- | --- | --- |
| **Description** | Constructor of the Fibonacci heap. Initializes class values. | |
| **Parameters** | maxNode  numberOfNodes | A pointer to the node with maximum frequency  The total number of nodes within the heap |
| **Return Value** | NA, it’s a constructor | |

void increaseKey(Node n, int k);

|  |  |  |
| --- | --- | --- |
| **Description** | Increases frequency of Node n by frequency k. Checks if node is a root or not, and if the parent’s frequency becomes smaller than Node n, then cut it and trigger a cascading cut if the parent node’s childCut value is true. | |
| **Parameters** | n  k | A node object associated with a node whose frequency will increase  The frequency that will be increased in Node n. |
| **Return Value** | void | |

void cut(Node child, Node parent)

|  |  |  |
| --- | --- | --- |
| **Description** | Removes child node from the parent node and moves child to root list. | |
| **Parameters** | child  parent | Node to be removed from parent  Node which is parent of the child |
| **Return Value** | void | |

void insert(Node node);

|  |  |  |
| --- | --- | --- |
| **Description** | Insert a node into the Max Fibonacci Heap | |
| **Parameters** | node | Adds node to root level list, if root level list is empty, make node the max node. |
| **Return Value** | void | |

void removeMax();

|  |  |  |
| --- | --- | --- |
| **Description** | Remove max node in the Fibonacci heap and run pairwiseCombine() remaining trees in root level list with the children of max node, if any. | |
| **Parameters** | NA |  |
| **Return Value** | void | |

void cascadingCut(Node n);

|  |  |  |
| --- | --- | --- |
| **Description** | Triggered from a node losing two children. If the childcut is true, remove nodes as you recursively move up the tree, otherwise set the childcut for the parent to be true. | |
| **Parameters** | n | A node of a child for which the cascading cut starts |
| **Return Value** | void | |

void pairwiseCombine();

|  |  |  |
| --- | --- | --- |
| **Description** | With a tree table, pairwise merge trees in the heap that have the same degree. | |
| **Parameters** | NA |  |
| **Return Value** | void | |

The remaining functions in FibonacciHeap.java serve as helper functions and functions created for the purpose of maintaining code readability by ensuring that functions have primarily one purpose. As with the keywordcounter.java, these remaining functions are small in purpose and quantity of code, thus should be readable from a quick inspection.