COP 5536 Advanced Data Structures Fall 2016
Programming Project Report

SANKET ACHARI
UFID – 71096329
sanketachari@ufl.edu

Working Environment:

1. Hardware Requirement: -
    - Hard Disk space: Minimum 5GB
    - Memory: 512 MB Minimum

2. Operating System:
    - WINDOWS 7 and above
    - Linux
    - MacOS

3. Compiler: - Standard Java Compiler (JDK)

Compiling Instructions:

- Place java files, input file and makefile in the same directory
- Open the terminal/cmd and change directory to where the files are located.
- For the compilation, type: **make compile**
- After compilation, you should see **hashtagcounter.class** and **FiboHeapHashTag.class and Node.class**
- To execute program for given input file, type: **make arg="*Input file name*" run**
- You should see *output_file.txt* in the working directory

Function Prototypes & Structure of the Programs

- Java Files:
    1. hashtagcounter.java
    2. FiboHeapHashTag.java

- hashtagcounter.java

    1. public class hashtagcounter
       Description: This class can be used for counting the hashtags present in the input file provided by the user. It calls utility functions for storing hashtags and its counts. Following operations can be performed:
       1. Remove Max
       2. Insert hashtag & its key
       3. Increase Key

    2. public static void main(String[] args)
       Description: This is the main method which reads the input text file and calls method countHashTags method to parse the information.
       Input Arguments: String argument which should be input file name along with its path if path is not the working directory
       Return Value: void

    3. private void countHashTags(String str)
       Description: Method which counts the number of hashtags present in the file. Based on input it can insert hashtags, remove max occurred hashtag and stop processing based on "STOP" input. This method also writes top hashtags in output_file.txt.
       Input Arguments: String of the input data
       Return Value: void

    4. private static void parseHashTag(String[] lines, FiboHeapHashTag fh)
       Description: Method which extracts hashtag and its count. It calls utility method for insertion of extracted hashtag and its count into the fibonacci heap
       Input Arguments: input of string array, object of Utility class
       Return Value: void

    5. private static String parseQuery(int n, FiboHeapHashTag fh)
       Description: Method which extracts the n number of top hashtags. This method calls remove Max method of FiboHeapHashTag class for the extraction of hashtag.
       Input Arguments: number, object of Utility class
       Return Value: String of top hashtags

6. private static boolean isNumeric(String str)
   Description: Method which checks whether given string has numeric characters
   Input Arguments: String
   Return Value: true if given string has numeric characters else false

- FiboHeapHashTag.java

1. `public class FiboHeapHashTag`
   Description: his class maintains advanced data structure Max fibonacci heap to store hashtags & its counts. Two methods insert & removeMax can be accessed, to store an element and delete max element from the fibonacci heap.
   Following operations can be performed:
   1. Remove Max
   2. Insert
   3. Cut
   4. Cascading Cut
   5. Increase Key
   6. Pairwise Combine

2. `private void addToRootList(Node x)`
   Description: Maintains list of roots of fibonacci heap
   Input Arguments: Node which has to be added to the root list of Fibonacci heap
   Return Value: void

3. `private void increaseKey(Node x, int k)`
   Description: Method which performs increase key operation of fibonacci heap. If child's count element is greater than parent's count element then cut that child and insert into root list.
   Input Arguments: Node whose count has to be increased, increase by number
   Return Value: void

4. `private void cut(Node current, Node parent)`
   Description: Method which performs cut operation of fibonacci heap. Child is cut from its parent and added to the root list of fibonacci heap.
   Input Arguments: Node which has to be cut from its parent, parent of that node
   Return Value: void

5. `private void cascadingCut(Node current)`
   Description: Method which performs cascading cut operation of fibonacci heap
   Input Arguments: Node parent whose child has been cut
   Return Value: void

6. `private Node combine(Node n1, Node n2)`
   Description: Method which combines the two node
   Input Arguments: two nodes
   Return Value: resulting node after the combination

7. `public String removeMax()`
   Description: Method which performs remove Max operation of fibonacci heap
   Input Arguments: Nothing
   Return Value: Name of the hashtag and its count in string format

8. `public void insert(String tag, int num)`
   Description: Method which performs insert operation of fibonacci heap
   Input Arguments: Hashtag, count of hashtag
   Return Value: void

9. `class Node`
   Description: This class maintains data structure of node which will be used in fibonacci heap. It has following fields
   1. Neighbors
   2. Parent
   3. Child
   4. Degree
   5. ChildCut
   6. Hashtag
   7. Count of hashtag

Running the program:

```
thunderx:14% ls
FiboHeapHashTag.java  hashtagcounter.java  Makefile  sampleInput.txt
thunderx:15% make compile
javac -g hashtagcounter.java
thunderx:16% ls
FiboHeapHashTag.class  hashtagcounter.class  Makefile    sampleInput.txt
FiboHeapHashTag.java   hashtagcounter.java   Node.class
thunderx:17% make arg=sampleInput.txt run
java hashtagcounter sampleInput.txt
thunderx:18% ls
FiboHeapHashTag.class  hashtagcounter.class  Makefile    output_file.txt
FiboHeapHashTag.java   hashtagcounter.java   Node.class  sampleInput.txt
thunderx:19%
```