# Advance Data Structures (COP 5536)

FALL 2018

Project Report

**Name:** BHARAT BANSAL
**UFID:** 2591-1118
**UF Email Account:**
bharatbansal@ufl.edu

# PROBLEM STATEMENT

A new search engine “DuckDuckGo” is implementing a system to count the most popular keywords used in their search engine. They want to know what the $n$ most popular keywords are at any given time. You are required to undertake that implementation. Keywords will be given from an input file together with their frequencies. You need to use a max priority structure to find the most popular keywords.

# PROJECT DESCRIPTION

The aim of this project is to implement a system to find the ‘n’ most popular keywords that are used in the search engine called ‘DuckDuckGo’.

The project is written in java and the project basically uses the two data structures which are **Max Fibonacci heap** and the **Hash table(i.e. Hash Map in java)**. Max Fibonacci heap is used to keep track of the frequencies of the keywords and the HashMap is used to map the keywords with the pointer to its corresponding node in the heap. No other external inbuild data structures are used.

# 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 Hash Tag Counter

**1) Extract the contents of the zip file**

**2) Type ‘make’ without the quotes.**

**3) Type ‘java popular_keyword  ‘file name**

# PROGRAM STRUCTURE

## Classes:

There are three classes in the program. They are:

1) **popular_keyword.java** – This is the main class and handles all the input and output.

2) **fibo_heap_fh.java** – This class is used to define all the functions pertaining to the Fibonacci heap.

3) **node_fibo_heap.java** – This class is used to instantiate the object of node in memory.

## Variables in classes and their description:

## popular_keyword.java

| Variables | Type | Description |
|---|---|---|
| input_filepath | String | The input filename taken from user. |
| hash | HashMap<String,node_fibo_heap> | The HashMap where node and keyword are stored. |
| fibo | fibo_heap_fh | The instance of the Fibonacci heap. |
| output_file | String | Name of the outputfile. |
| input_pattern | Pattern | Stores the pattern of the input given, the format of the input.<br><br>([$])([a-z_]+)(\\s)(\\d+) |
| digit_pattern | Pattern | Stores the pattern of the digits to be removed from the keyword. |

| | | |
|---|---|---|
| input_match | Matcher | Matcher object for input_pattern |
| digit_match | Matcher | Matcher object for digit_pattern |
| Keyword | String | keyword to be stored. |
| Key | Integer | Key value to be stored in node. |
| n | node_fibo_heap | Represents a new node |
| inc_key_val | Integer | Old key value+ new key value. |
| num_rem_nodes | Integer | Number of nodes to be removed |
| rem_nodefiboheaps | ArrayList<node_fibo_heap> | Stores all the removed nodes |
| prog_sTime | long | Start time of program. |
| prog_eTime | long | End time of the program. |
| prog_tTime | long | End time of the program. |

## fibo_heap_fh.java

| Class Variable | Type | Description |
|---|---|---|
| 1. maximum_nodefiboheap | node_fibo_heap | Points to the maximum node in the heap. |
| 2. nodes_in_heap | Integer | Signifies the number of nodes that are stored in the heap. |

## node_fibo_heap.java

| Class Variable | Type | Description |
|---|---|---|
| d_node | Integer | Signifies the number of nodes that a node can have in its next level. |
| childCut | Boolean | A Child Cut of false means that no child has ever been removed from that node. |
| keyword | String | Contains the keyword. |
| key | Integer | Signifies the value of integer. |
| l_nodefiboheap | node_fibo_heap | Points to the left node in the circular list. |
| r_ nodefiboheap | node_fibo_heap | Points to the right node in the circular list. |
| p_nodefiboheap | node_fibo_heap | Points to the parent node |
| c_nodefiboheap | node_fibo_heap | Points to the child node. |
| | | |

## Function Prototype in classes and their description:

## node_fibo_heap.java

| Function Name | Return Type | Parameters | Description |
|---|---|---|---|
| **node_fibo_heap(String , int )** | - | String, int | Constructor that initializes the keyword and key. |
| **public String pop_keyword()** | String | - | Returns the keyword of the node. |

## fibo_heap_fh.java

| Function Name | Return Type | Parameters | Description |
|---|---|---|---|
| **public void node_insertion( node_fibo_heap)** | Void | node_fibo_heap | This function inserts a node into the max Fibonacci heap. |
| | | | |
| **public void node_childCut_ check(node_fibo _heap)** | Void | node_fibo_heap | This function checks the childCut value and makes the necessary changes and performs remove if needed. |
| **public void node_inc_key_v al(node_fibo_he ap, int )** | Void | node_fibo_heap , int | This function is used to increase the value of the key to given value in node. |
| **public node_fibo_heap node_rem_maxi mum()** | node_fibo_heap | Null | This function removes the maximum node, from the Fibonacci heap. |
| **public void node_same_degr ee_merge()** | Void | Null | This function performs degree wise merge. It melds the nodes in the root list of the heap. |
| **public void node_child(node _fibo_heap, node_fibo_heap )** | Void | node_fibo_heap, node_fibo_heap | This function is used to make one node a child of other node . |