

PROJECT REPORT

Name: Yingfei Xiang UFID: 08657889 College: CISE Email: yingfeixiang@ufl.edu

Development Tools: Eclipse Java EE IDE(Version: Mars.2 Release 4.5.2)

Development Environment: JDK 1.8.0_71, JRE 1.8.0_71, Windows 7, Ubuntu

DESCRIPTION

According to the problem description of the project, we need to implement a system to find out the N most popular hashtags appeared on social media such as Facebook or Twitter. In order to solve the problem, we can simplify and conclude it as “Top N problem”. There’re many different solutions to the problem, such as sorting all the hashtags by frequency in descending order and then get first N hashtags. We can also use priority queues, for instance, Linked List and Binary Heap in solving the problem. However, when concerning the efficiency and speed, the Fibonacci Heap definitely yields better performance. The comparison of the the time cost of different operations among the priority queues is shown in chart as following:

Operation	Linked List	Heaps			
		Binary	Binomial	Fibonacci †	Relaxed
make-heap	1	1	1	1	1
insert	1	log N	log N	1	1
find-min	N	1	log N	1	1
delete-min	N	log N	log N	log N	log N
union	1	N	log N	1	1
decrease-key	1	log N	log N	1	1
delete	N	log N	log N	log N	log N
is-empty	1	1	1	1	1

Fig.1 The comparison of time cost among the priority queues

From above, it is obvious that Fibonacci Heap is the better choice to be used to implement the project. Since we are supposed to find out the N most popular hashtags from input data files, the max Fibonacci Heap is the best data structure we need. The frequencies of hashtags are stored as nodes in max Fibonacci Heap, we also make use of Hash Table to store the hashtags as key and then combining with max Fibonacci

Heap through the value(as pointer to corresponding node in max Fibonacci Heap). If we need to find out Top N hashtags, then we just extract Top N nodes from max Fibonacci Heap. The details of each operation in max Fibonacci Heap and different functions in my program will be presented later.

STRUCTURE & FUNCTION

The Fibonacci Heap, which is developed by Michael L. Fredman and Robert E. Tarjan in 1984 and published it in a scientific journal in 1987. They named Fibonacci Heap after the Fibonacci numbers, which are used in their running time analysis. As for the structure of max Fibonacci Heap, it consists of set of max-heap ordered trees. In order to keep track the information of all the nodes as well as the corresponding hashtags, several variables are needed, like the key of the node, the degree of the node, mark the node whether its child is removed, the string to store the hashtags, etc.. The details of the construction of the structure of max Fibonacci Heap is shown as following graph:

```
//Node structure for maxFibHeap
public class maxFibNode {
    maxFibNode leftsib;    // left sibling
    maxFibNode rightsib;   // right sibling
    maxFibNode child;      // child of the node
    maxFibNode parent;     // parent of the node
    int key;               // key of the node
    int degree;            // degree of the node
    boolean mark_val;      // mark the node whether its child is removed
    String hashtags;       // store hashtags in string

    //Initialize the variables and parameters
    public maxFibNode(int key) {
        this.leftsib = this;
        this.rightsib = this;
        this.child = null;
        this.parent = null;
        this.key = key;
        this.degree = 0;
        this.mark_val = false;
        this.hashtags = null;
    }
}
```

Fig.2 The structure of max Fibonacci Heap(combined with Hash Table)

After the structure of max Fibonacci Heap is finished, it comes to implementing the main functions of max Fibonacci Heap which are used in finding out the N most popular hashtags. Since each node includes the information of its parent node, child node as well as siblings, all the nodes in max Fibonacci Heap are stored in linked list. As for the two basic functions: add() and insert(), it is very important to build up the max Fibonacci Heap with root and the other nodes. In the process of insert(), the node is auto-adjusted when new node has greater value than the present node. And the hashtags are combined with the node as Hash Table. The details of add() and insert() are shown as following:

```

//Add nodes before root(in linked list)
private void add(maxFibNode node, maxFibNode parent) {
    node.leftsib = parent.leftsib;
    parent.leftsib.rightsib = node;
    node.rightsib = parent;
    parent.leftsib = node;
}

//Insert nodes into maxFibHeap, combine the nodes with hashtags and the key(using hasht:
public void insert(int key, Hashtable table,String hashtags) {
    maxFibNode node = new maxFibNode(key);
    if (num_nodes == 0)
        max_node = node;
    else {
        add(node, max_node);
        //Adjust if new node is greater than max_node
        if (node.key > max_node.key)
            max_node = node;
    }
    num_nodes++;
    table.put(hashtags, node);
    node.hashtags = hashtags;
}

```

Fig.3 The functions of add() and insert()

Besides above two basic functions, it is also essential to get the max key and max String which is combined with the max node from max Fibonacci Heap. Thus, maxkey() and maxString() are needed to be implemented. The details of maxkey() and maxString() are shown as following:

```

//Return the max key in maxFibHeap, return -1 if failed
public int maxkey() {
    if (max_node == null)
        return -1;
    return max_node.key;
}

//Return the String value combined with the max node
public String maxString()
{
    String max_str = null;
    if (max_node == null)
        return null;
    max_str = max_node.hashtags;
    return max_str;
}

```

Fig.4 The functions of maxkey() and maxString()

Next, there are two functions that takes as an important part in the process of max Fibonacci Heap as well as retrieving the Top N hashtags from input. The function removeMax() is used to remove the node with max key in max Fibonacci Heap and then add its child as well as its child's siblings to the root of the linked list. The details of removeMax() is shown as following:

```

//Remove the node with max key in maxFibHeap
public void removeMax() {
    if (max_node == null)
        return ;
    maxFibNode l = max_node;
    //Add the child and child's siblings to maxFibNode(the root of the linked list)
    while (l.child != null) {
        maxFibNode child = l.child;
        child.leftsib.rightsib = child.rightsib;
        child.rightsib.leftsib = child.leftsib;
        if (child.rightsib == child)
            l.child = null;
        else
            l.child = child.rightsib;
        child.parent = null;
        add(child, max_node);
    }
    l.leftsib.rightsib = l.rightsib;
    l.rightsib.leftsib = l.leftsib;
    //If l is the only node in the heap, set the max node to NULL
    //Else set the max node as l.leftsib, then consolidate()
    if (l.leftsib == l)
    {
        max_node = null;
        //System.out.println("only max node!");
    } else {
        max_node = l.leftsib;
        consolidate();
    }
    l = null;
    num_nodes--;
}

```

Fig.5 The functions of removeMax()

Furthermore, consolidate() is applied to merge the root nodes with same degrees after the max node is removed from max Fibonacci Heap. The details of consolidate() is shown as following:

```

//Consolidate the nodes in maxFibHeap after the max node is removed
private void consolidate() {
    int max_degree = num_nodes;
    maxFibNode[] cns = new maxFibNode[max_degree+2];
    for (int i = 0; i < max_degree+1; i++)
        cns[i] = null;
    //Merge the root node with same degrees
    while (max_node != null) {
        maxFibNode x = max_node;
        maxFibNode p = max_node;
        if (p==p.leftsib) max_node=null;
        else {
            p.leftsib.rightsib = p.rightsib;
            p.rightsib.leftsib = p.leftsib;
            max_node = p.leftsib;
        }
        p.leftsib = p;
        p.rightsib = p;
        int d = x.degree;
        while (cns[d] != null)
        {
            if (x.key < cns[d].key)
            {
                maxFibNode temp = x;
                x = cns[d];
                cns[d] = temp;
            }
            link(cns[d], x);
            cns[d] = null;
            d++;
        }
        cns[d] = x;
    }
}

```

Fig.5 The functions of removeMax()

Last but not the least, after increasing the key of the node from old value to new value, the new node may be greater than its parent. Hence, the node needs to be cut from its parent and then add it to the root of linked list. Cascading cut should also be performed to set the correct mark_val of each node. The details of cut() and cascadingCut() are shown as following:

```
//Cut the node from its parent and then add it to the root linked list of maxFibHeap
protected void cut(maxFibNode node, maxFibNode parent) {
    parent.degree--;
    node.rightsib.leftsib = node.leftsib;
    node.leftsib.rightsib = node.rightsib;
    //If node has no siblings
    if (parent.degree == 0) {
        parent.child = null;
    }
    if (parent.child == node) {
        parent.child = node.rightsib;
    }
    node.leftsib = max_node;
    node.rightsib = max_node.rightsib;
    max_node.rightsib = node;
    node.rightsib.leftsib = node;
    node.parent = null;
    node.mark_val = false;
}

//If it violates the rule of maxFibHeap after increasing the key,
//then cascading cut the node d
protected void cascadingCut(maxFibNode d) {
    maxFibNode parentd = d.parent;
    if (parentd != null) {
        if (!d.mark_val) {
            d.mark_val = true; //Set the mark_val to true
        } else {
            cut(d, parentd);
            cascadingCut(parentd);
        }
    }
}
```

Fig.6 The functions of cut() and cascadingCut()

The details of increase() is shown as following:

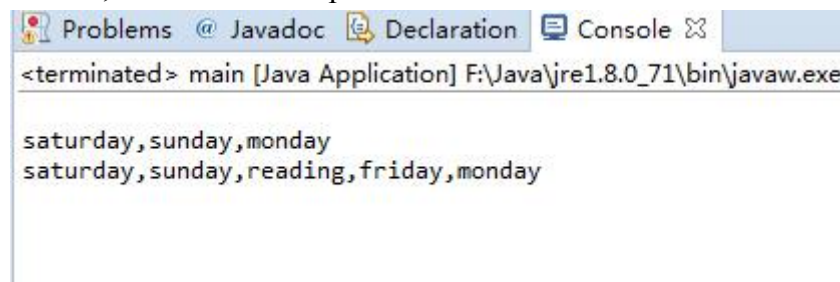
```
//Increase the old key of the node to new key
public void increase(Hashtable<String, maxFibNode> table, String hashtags, int new_key)
{
    maxFibNode nd;
    nd = table.get(hashtags);
    int old_val = nd.key;
    int new_val = 0;
    if (nd != null)
        //Update the key(frequency) to new_key
        if (new_key > 0)
        {
            new_val = new_key + old_val;
            if (max_node == null || nd == null)
                return ;
            if (new_val < nd.key)
                return ;
            maxFibNode parent = nd.parent;
            nd.key = new_val;
            //Cut the node from its parent and then add it to the root linked list
            if (parent != null && (nd.key > parent.key)) {
                cut(nd, parent);
                cascadingCut(parent);
            }
        }
    //Renew the max node
    if (nd.key > max_node.key)
        max_node = nd;
}
```

Fig.7 The functions of increase()

EXPERIMENT & RESULT

After the structure and the functions of max Fibonacci Heap(combined with Hash Table) are implemented, I build up a main.java file to test the sample input and it is also the entry point of whole project. To read from input data file, I have implemented three ways of text input. First, I set an static String variable input_data to store all the input data as a string. Although it is easiest to implement, it is hard to update and change its input data and it is also bad for maintainence. Second, I set two String variables, inputfileName and outputfileName, for typing the exact location of input data file and output file repectively. And then I use Scanner to read data from the input file and PrintWriter to ouput the result to destination file. This is better than first method, however, we cannot set any other location of the input data file as well as the output file. Therefore, the third method I used is the most preferred one. I set two arguments, args[0] args[1], which represents for inputfileName and outputfileName, are used as two input parameters when compiling and running the codes.

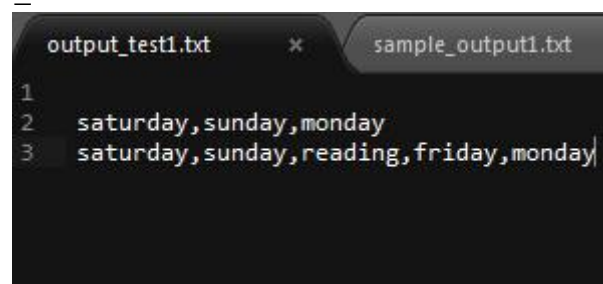
As for the sample input data from Project1.pdf, the size is small. After compiling and running the codes, the result in Eclipse console:



```
<terminated> main [Java Application] F:\Java\jre1.8.0_71\bin\javaw.exe
saturday,sunday,monday
saturday,sunday,reading,friday,monday
```

Fig.8 The result in Eclipse console for sample data in PDF

The result in output_test1.txt:



```
1
2 saturday,sunday,monday
3 saturday,sunday,reading,friday,monday
```

Fig.9 The result in output_test1.txt for sample data in PDF

And the correct answer is:

Following is the output file for the above input file.

```
saturday,sunday,monday
saturday,sunday,reading,friday,monday
```

Fig.10 The correct answer for sample data in PDF

As for the sample input data from sampleInput.txt, after compiling and running the codes, the result in Eclipse console:

```
#cholangiography 10 #chiseled 2 #chivarras 1 #chivarras 5 #chloroform 8 #chlorination 5 #chlamydia 4 #chivarras 5 #chloroprene 9 #cholelithotomy, chlorococcum, chloramine, chon, chivarras
chloramine, chloroprene, chivarras, chloral, chlorococcum, cholelithotomy, chlorothiazide
chloramine, chirurgy, chivarras, chloroprene, chisel, chocolate, chloral, chloroquine, chlorococcum
choke, chokidar
choke, chishona, chloroquine, chloramphenicol, chloroprene, chokidar, chirurgy, chlorothiazide, choleraic, chokra, chloramine, chivarras, chlorococcum, chishona, choke, chirurgery, cholelithotomy, chitterings, chokra, chloroprene, chisel, choleraic, cholecystectomy, chloramphenicol
chishona, cholelithotomy, chlorococcum, choke, choleraic, chloramphenicol, chivarras
choke, chlorura, chisel, cholelithotomy, choleraic, chishona, chlorophyll, chivarras
choke, chisel, cholelithotomy
chlorura, chlorophyll, cholecystectomy, choleraic, cholelithotomy, chisel, choke
chlorophyll, chlorura, choleraic, cholelithotomy, cholecystectomy, choke, chlorella, chlorococcum, chisel
```

Fig.11 The result in Eclipse console for sample data in sampleInput.txt

The result in output_test2.txt:

```
cholelithotomy, chlorococcum, chloramine, chon, chivarras
chloramine, chloroprene, chivarras, chloral, chlorococcum, cholelithotomy, chlorothiazide
chloramine, chirurgy, chivarras, chloroprene, chisel, chocolate, chloral, chloroquine, chlorococcum
choke, chokidar
choke, chishona, chloroquine, chloramphenicol, chloroprene, chokidar, chirurgy, chlorothiazide, choleraic, chokra, chloramine, chivarras, chlorophyll, chon, chirurgery, choir, cholelithotomy, chlorura
chlorococcum, chishona, choke, chirurgery, cholelithotomy, chitterings, chokra, chloroprene, chisel, choleraic, cholecystectomy, chloramphenicol, chirurgy, chivarras, chloroquine
chishona, cholelithotomy, chlorococcum, choke, choleraic, chloramphenicol, chivarras
choke, chlorura, chisel, cholelithotomy, choleraic, chishona, chlorophyll, chivarras
choke, chisel, cholelithotomy
chlorura, chlorophyll, cholecystectomy, choleraic, cholelithotomy, chisel, choke
chlorophyll, chlorura, choleraic, cholelithotomy, cholecystectomy, choke, chlorella, chlorococcum, chisel
```

Fig.12 The result in output_test2.txt for sample data in sampleInput.txt

And the correct answer is:

```
cholelithotomy, chlorococcum, chloramine, chivarras, chon
chivarras, chloroprene, chloramine, chloral, chlorococcum, cholelithotomy, chlorothiazide
chloramine, chivarras, chirurgy, chloroprene, chisel, chocolate, chloral, chloroquine, chokidar
choke, chokidar
choke, chishona, chloroquine, chloramphenicol, chokidar, chloroprene, chirurgy, chlorothiazide, choleraic, chokra, chloramine, chivarras, chlorophyll, chon, choir, chirurgery, cholelithotomy, cholecystectomy
chlorococcum, chishona, choke, chirurgery, cholelithotomy, chokra, chloroprene, chitterings, chisel, choleraic, cholecystectomy, chloramphenicol, chirurgy, chloroquine, chivarras
chishona, cholelithotomy, chlorococcum, choke, choleraic, chloramphenicol, chivarras
choke, chlorura, chisel, cholelithotomy, choleraic, chishona, chlorophyll, chivarras
choke, chisel, cholelithotomy
chlorura, cholecystectomy, chlorophyll, choleraic, cholelithotomy, choke, chisel
chlorophyll, chlorura, choleraic, cholelithotomy, cholecystectomy, choke, chlorella, chlorococcum, chisel
```

Fig.13 The correct answer for sample data in sampleInput.txt

Besides running the project in Eclipse or through console window using javac/java manually, I also build up a makefile which has 6 different functions when dealing with the project.(make new; make build; make clear; make rebuild; make run; make jar) The details are described in makefile file.

As for the sample input data from sample_input1.txt, after compiling and running the codes via makefile, the result in command console and output_test3.txt are shown as following:

```
F:\workplace\ADSprojecttest\hashtagcounter>make run inputfile=sample_input1.txt
outputfile=output.txt
```

```
choirmaster,chokefull,chlamys,chlorination,chirr,chirurgy,chloramphenicol,chloramine
chokefull,chlorophthalmidae,chlorococcales,chitterings,chlorination,chokra,chloramphenicol,cholinergic,chiseling
chokefull,chirr,chlamys,chlorophyta,chiseling,chirurgy,chlorophyllose,chloramphenicol,chlorococcales,chlorophthalmidae
chokefull,chiseling,chloramphenicol,chirr,chirurgy,chitterings,chisel
chiseling,chloramphenicol,chokefull,chloroform,chlorococcales,chlamys,choline,cholinergic,chisel,chloranthaceae,chirr,chirurgy,chitterings,chlorophthalmidae
chloramphenicol,chiseling,chisel,chlorination,chirr
chiseling,chloramphenicol,chlorination,chloranthaceae,chlorococcales,chisel,chloroform,choeronycteris,choleric,chirr,chlorosis,chirurgical,chlortetracycline,chlorophyta
chiseling,chloramphenicol,choeronycteris,chokra,chlorination,chloranthaceae,chlorococcales,choleric
choeronycteris
chloramphenicol,choice,chitinous,choeronycteris
chondrosarcoma,choice,chitinous,choeronycteris,chloramphenicol,chiseling,choirmaster,chokra,chlorination,chlorosis
```

```
choirmaster,chokefull,chlamys,chlorination,chirr,chirurgy,chloramphenicol,chloramine
chokefull,chlorophthalmidae,chlorococcales,chitterings,chlorination,chokra,chloramphenicol,cholinergic,chiseling
chokefull,chirr,chlamys,chlorophyta,chiseling,chirurgy,chlorophyllose,chloramphenicol,chlorococcales,chlorophthalmidae
chokefull,chiseling,chloramphenicol,chirr,chirurgy,chitterings,chisel
chiseling,chloramphenicol,chokefull,chloroform,chlorococcales,chlamys,choline,cholinergic,chisel,chloranthaceae,chirr,chirurgy,chitterings,chlorophthalmidae
chloramphenicol,chiseling,chisel,chlorination,chirr
chiseling,chloramphenicol,chlorination,chloranthaceae,chlorococcales,chisel,chloroform,choeronycteris,choleric,chirr,chlorosis,chirurgical,chlortetracycline,chlorophyta
chiseling,chloramphenicol,choeronycteris,chokra,chlorination,chloranthaceae,chlorococcales,choleric
choeronycteris
chloramphenicol,choice,chitinous,choeronycteris
chondrosarcoma,choice,chitinous,choeronycteris,chloramphenicol,chiseling,choirmaster,chokra,chlorination,chlorosis
```

Fig.14 The result in console and output_test3.txt for sample data in sample_input1.txt

And the correct answer is:

```
choirmaster23,chokefull121,chlamys18,chlorination18,chirr17,chirurgy16,chloramphenicol15,chloramine14
chokefull137,chlorococcales29,chlorophthalmidae29,chitterings28,chlorination27,chokra25,chloramphenicol24,chiseling24,cholinergic24
chokefull137,chlamys37,chirr37,chiseling34,chlorophyta34,chlorophyllose32,chirurgy32,chloramphenicol31,chlorococcales29,chlorophthalmidae29
chokefull147,chiseling46,chloramphenicol46,chirr42,chitterings41,chirurgy41,chisel38
chiseling55,chloramphenicol54,chloroform47,chlorococcales46,chlamys46,cholinergic44,choline44,chloranthaceae43,chisel43,chirr42,chitterings41,chirurgy43
,chlorophthalmidae38
chloramphenicol65,chiseling63,chisel54,chlorination52,chirr51
chiseling76,chloramphenicol73,chlorination58,chloranthaceae58,chlorococcales56,chisel54,choleric52,choeronycteris52,chloroform52,chirr51,chirurgical50,chlorosis50,chlortetracycline50,chlorophyta49
chiseling76,chloramphenicol73,choeronycteris65,chokra63,chloranthaceae58,chlorococcales58,chlorination58,choleric57
choeronycteris79
chloramphenicol191,choice84,chitinous82,choeronycteris81
chondrosarcoma106,choice97,chitinous97,choeronycteris91,chloramphenicol191,chiseling89,chlorination77,chokra77,choirmaster77,chlorosis74
```

Fig.15 The correct answer for sample data in sample_input1.txt

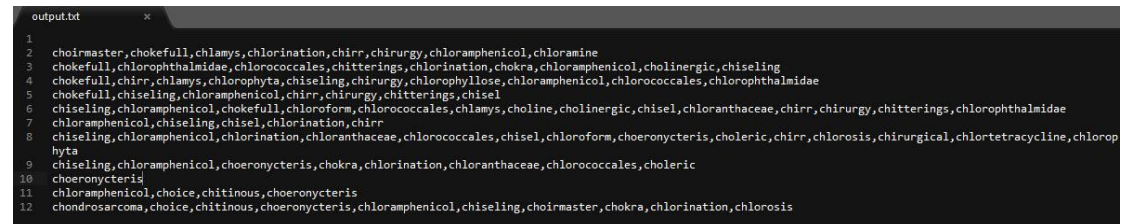
Besides compiling and running in Eclipse and via makefile, I also pack the project into an executable jar package which is compatible for different platforms and environment.

.settings	2016/11/17 11:59
bin	2016/11/18 14:37
src	2016/11/18 14:37
.classpath	2016/11/17 11:59
.project	2016/11/17 11:59
hashtagcounter	2016/11/18 18:07
makefile	2016/11/18 16:20
output	2016/11/18 18:09
output_test1	2016/11/18 17:49
output_test2	2016/11/18 17:55
output_test3	2016/11/18 18:01

Fig.16 The location of hashtagcounter.jar and makefile

As for the sample input data from sample_input1.txt, after compiling and running the codes by using hashtagcounter.jar, the result in output.txt is shown as following:

```
F:\workplace\ADSprojecttest\hashtagcounter>java -jar hashtagcounter.jar "F://workplace\ADSprojecttest\hashtagcounter/src\hashtagcounter/sample_input1.txt" "F://workplace\ADSprojecttest\hashtagcounter/output.txt"
```



```
output.txt
1
2 choirmaster,chokefull,clamys,chlorination,chirr,chirurg,chloramphenicol,chloramine
3 chokefull,chlorophthalmidae,chlorococcales,chitterings,chlorination,chokra,chloramphenicol,cholinergic,chiseling
4 chokefull,chirr,clamys,chlorophyta,chiseling,chirurg,chlorophyllose,chloramphenicol,chlorococcales,chlorophthalmidae
5 chokefull,chiseling,chloramphenicol,chirr,chirurg,chitterings,chisel
6 chiseling,chloramphenicol,chokefull,chloroform,chlorococcales,clamys,choline,cholinergic,chisel,chloranthaceae,chirr,chirurg,chitterings,chlorophthalmidae
7 chloramphenicol,chiseling,chisel,chlorination,chirr
8 chiseling,chloramphenicol,chlorination,chloranthaceae,chlorococcales,chisel,chloroform,choeronycteris,choleric,chirr,chlorosis,chirurgical,chlortetracycline,chlorophyta
9 chiseling,chloramphenicol,choeronycteris,chokra,chlorination,chloranthaceae,chlorococcales,choleric
10 choeronycteris
11 chloramphenicol,choice,chitinous,choeronycteris
12 chondrosarcoma,choice,chitinous,choeronycteris,chloramphenicol,chiseling,choirmaster,chokra,chlorination,chlorosis
```

Fig.17 The result in output.txt for sample data in sample_input1.txt

From above experiments and results, it is clear that all the output results are the same as the correct answers given by TA. And all the codes are running successfully not only on my personal laptop but also on thunder.cise.ufl.edu.

For detailed information of my project, please refer to the codes and comments, makefile and Readme.txt.