**CSC 323 Project 2 (Java)**

Tyler Gaugler

Due Date: 9/22/2016

Algorithm Steps for this project:

Step 1:  
        - open the input file;  
        - read and count the number of date item in the  input file  
        - close the file  
        - dynamically allocate the PQAry of the size count+1  
          and initallize PQAry[0] to 0. Remember, PQAry[1] is the front.  
  
step 2: call buildPQAry (see algorithm below)  
  
step 3: call deletePQAry (see algorithm below)  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
buildHeap algorithm:  
  
    step 1: open the input file for the second time  
  
    step 2: data <-- get a data item from the input file  
  
    step 3: insertOneDataItem (data) // make sure heap is not full  
               bubbleUp // taught in class  
  
    step 4: printPQAry (print the content of PQAryAry up to the first 10 items of the PQAry  
                 (from PQAryAry[0]) to \*out1 file\*  
            // in the real life, this is only for your debugging purpose  
  
    step 5: repeat step 2 - 4 while input file is NOT empty  
  
    ste[ 6: close the input file  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
deleteRoot algorithm:  
  
    step 1: open the output file  
  
    step 2:  
                - deleteRoot  // make sure heap is not full  
                - print the root to \*out2 file\* // out2 is the sorted input data  
                - bubbleDown // taught in class  
  
    step 3: printPQAry (print the content of PQAryAry up to the first 10 items of the PQAry  
                 (from PQAryAry[0]) to \*out1 file\*  
            // in the real life, this is only for your debugging purpose  
  
    ste[ 4: repeat step 2 and 3 while PQAry is NOT empty  
  
    step 5: - close the two output files

**Source Code:**

//ProjectDriver.java

import java.io.\*;

import java.util.Scanner;

public class ProjectDriver {

public static void main(String args []){

int size=0;

try {

Scanner inputFile = new Scanner(new FileReader(args[0]));

while(inputFile.hasNext()){

inputFile.next();

size++;

}

inputFile.close();

PQSort myPriorityQueue= new PQSort(size);

myPriorityQueue.buildPQArray(args[0],args[1]);

myPriorityQueue.deletePQArray(args[1],args[2]);

} catch (Exception e) {

e.printStackTrace();

}

}

}

// PQSort.java

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class PQSort {

int PQArray[];

PQSort(int size){

PQArray= new int[size+1];

PQArray[0]=0;

}

public void buildPQArray(String filename1, String filename2){

try {

Scanner inputFile = new Scanner(new FileReader(filename1));

FileWriter outputFile = new FileWriter(filename2);

String data;

while(inputFile.hasNext()){

data= inputFile.next();

insertOneDataItem(Integer.parseInt(data));

outputFile.write(PQprint()+"\n");

}

inputFile.close();

outputFile.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private void insertOneDataItem(int value){

int size= PQArray[0];

PQArray[size+1]=value;

PQArray[0]++;

bubbleUp(size+1);

}

private void bubbleUp(int child\_index){

while(PQArray[child\_index] < PQArray[child\_index/2] ){

if(child\_index == 1)break;

swapValues(child\_index/2, child\_index);

child\_index= child\_index/2;

}

}

private void swapValues(int value1, int value2){

int temp= PQArray[value1];

PQArray[value1]= PQArray[value2];

PQArray[value2]= temp;

}

private String PQprint(){

int size= (PQArray[0] < 10)? PQArray[0] : 10;

int i=0;

String printed\_list="";

while(i <= size){

printed\_list= printed\_list.concat(PQArray[i]+" ");

i++;

}

return printed\_list;

}

public void deletePQArray(String filename1, String filename2){

try {

FileWriter outputFile1 = new FileWriter(filename1, true);

outputFile1.write("\n\n");

FileWriter outputFile2= new FileWriter(filename2);

while(PQArray[0] > 0){

outputFile1.write(PQprint() + "\n");

System.out.println(PQprint());

outputFile2.write(deleteRoot()+"\n");

bubbleDown();

}

outputFile1.close();

outputFile2.close();

} catch (IOException e) {

e.printStackTrace();

}

}

private void bubbleDown(){

int parent\_index= 1;

while( parent\_index\*2 <= PQArray[0] ){

int smaller\_child\_index= parent\_index\*2;

if( (parent\_index\*2)+1 <= PQArray[0] && PQArray[smaller\_child\_index] > PQArray[(parent\_index\*2)+1]){

smaller\_child\_index= (parent\_index\*2)+1;

}

if(PQArray[parent\_index] > PQArray[smaller\_child\_index]){

swapValues(parent\_index, smaller\_child\_index);

}else {break;}

parent\_index= smaller\_child\_index;

}

}

private String deleteRoot(){

if(PQArray[0] > 0){

int root= PQArray[1];

int size= PQArray[0];

PQArray[1]=PQArray[size];

PQArray[0]--;

return Integer.toString(root);

}

return "";

}

}

**Output**

//output1.txt

1 23

2 23 82

3 12 82 23

4 12 53 23 82

5 9 12 23 82 53

6 9 12 23 82 53 39

7 2 12 9 82 53 39 23

8 2 12 9 13 53 39 23 82

9 2 12 9 13 53 39 23 82 30

10 2 12 9 13 53 39 23 82 30 61

11 2 12 9 13 21 39 23 82 30 61

12 2 12 9 13 21 39 23 82 30 61

13 2 12 9 13 21 18 23 82 30 61

14 2 12 9 13 21 18 19 82 30 61

15 2 12 9 13 21 18 19 82 30 61

16 2 12 9 13 21 18 19 53 30 61

17 2 4 9 12 21 18 19 13 30 61

18 2 4 9 12 21 18 19 13 30 61

19 2 4 9 12 21 18 19 13 30 61

20 2 4 9 12 8 18 19 13 30 21

21 2 4 9 12 8 18 19 13 30 21

22 2 4 9 12 8 18 19 13 30 21

23 2 4 9 12 8 18 19 13 30 21

24 2 4 9 12 8 11 19 13 30 21

25 2 4 7 12 8 9 19 13 30 21

26 2 4 7 12 8 8 19 13 30 21

27 2 4 5 12 8 7 19 13 30 21

28 2 4 5 12 8 7 10 13 30 21

29 2 4 5 12 8 7 10 13 30 21

29 2 4 5 12 8 7 10 13 30 21

28 4 8 5 12 10 7 10 13 30 21

27 5 8 7 12 10 8 10 13 30 21

26 7 8 8 12 10 9 10 13 30 21

25 8 10 8 12 20 9 10 13 30 21

24 8 10 9 12 20 11 10 13 30 21

23 9 10 10 12 20 11 19 13 30 21

22 10 12 10 13 20 11 19 39 30 21

21 10 12 11 13 20 18 19 39 30 21

20 11 12 18 13 20 23 19 39 30 21

19 12 13 18 30 20 23 19 39 34 21

18 13 20 18 30 21 23 19 39 34 61

17 18 20 19 30 21 23 42 39 34 61

16 19 20 23 30 21 36 42 39 34 61

15 20 21 23 30 37 36 42 39 34 61

14 21 30 23 34 37 36 42 39 93 61

13 23 30 36 34 37 45 42 39 93 61

12 30 34 36 39 37 45 42 53 93 61

11 34 37 36 39 53 45 42 53 93 61

10 36 37 42 39 53 45 82 53 93 61

9 37 39 42 53 53 45 82 61 93

8 39 53 42 61 53 45 82 93

7 42 53 45 61 53 93 82

6 45 53 82 61 53 93

5 53 53 82 61 93

4 53 61 82 93

3 61 93 82

2 82 93

1 93

//output2.txt

2

4

5

7

8

8

9

10

10

11

12

13

18

19

20

21

23

30

34

36

37

39

42

45

53

53

61

82

93

**Input**

//input.txt

23  
82 12 53  
9 39 2 13  
30  
61  
21  
45  
18 19 42  
53 4 93 34 8 36 37  
10 11  
7 8  
5 10 20