CSC 310 Data Structures

Homework 6

CSC310\_HW6\_1

import java.util.LinkedList;

import java.util.Scanner;

class PriorityQueue{

private LinkedList<Item> l = new LinkedList<Item>();

class Item{

int key;

Object value;

Item(int k, Object v){

key=k;

value=v;

}

}

public void add(int k, Object v){

Item newest=new Item(k,v);

if(!(l.isEmpty())){

Item walk= l.getLast();

while(walk !=null && k<walk.key){

if(l.indexOf(walk)-1<0){

walk=null;

}else

walk =l.get(l.indexOf(walk)-1);

}

if(walk==null){

l.addFirst(newest);

}else

l.add(l.indexOf(walk)+1, newest);

}else{

l.add(newest);

}

}

public int size(){

if(l.isEmpty())

return 0;

return l.size();

}

public Object min(){

if (l.isEmpty())

return null;

return l.getFirst().value;

}

public Object removeMin(){

if(l.isEmpty())

return null;

Item out = l.removeFirst();

return out.value;

}

}

public class CSC310\_Hw6\_1 {

public static void main(String[] args) {

// simple menu for easy use

Scanner input = new Scanner(System.in);

String choice = "";

PriorityQueue pq = new PriorityQueue();

do{

System.out.println("1. Run Demo");

System.out.println("2. Add to queue");

System.out.println("3. Get Min");

System.out.println("4. Remove Min");

System.out.println("5. Get size");

System.out.println("e. Exit");

System.out.print("Pick an option from the menu above:");

choice = input.next();

switch(choice.charAt(0)){

case '1':

PriorityQueue pqDemo = new PriorityQueue();

System.out.println("Sequence: 7 => 4 => 8 => 2 => 5 => 3 => 9");

System.out.println("");

pqDemo.add(7, 7);

pqDemo.add(4, 4);

pqDemo.add(8, 8);

pqDemo.add(2, 2);

pqDemo.add(5, 5);

pqDemo.add(3, 3);

pqDemo.add(9, 9);

int size=pqDemo.size();

System.out.print("Priority: ");

for (int i = 0; i < size; i++) {

System.out.print(pqDemo.removeMin());

if(i < size-1){

System.out.print(" => ");

}

}

System.out.println("");

break;

case '2':

System.out.print("Enter key: ");

int key = input.nextInt();

System.out.print("Enter object: ");

String value=input.next();

try{

int intval=Integer.parseInt(value);

pq.add(key, intval);

}catch(Exception e){

pq.add(key, value);

}

break;

case '3':

Object min=pq.min();

if(min==null)

System.out.println("Queue is Empty");

else

System.out.println("Min Without Removing: "+ min);

break;

case '4':

Object minR=pq.min();

if(minR==null)

System.out.println("Queue is Empty");

else

System.out.println("Min Without Removing: "+ minR);

break;

case '5':

System.out.println("Size: "+pq.size());

break;

case 'e': case 'E':

System.out.println("Goodbye!");

break;

default:

System.out.println("You entered an invalid menu choice. Please try agin.");

break;

}

System.out.println("");

} while (choice.charAt(0)!='e' && choice.charAt(0) != 'E');

}

}

CSC310\_HW6\_2

import java.util.Scanner;

class MaxHeap{

private int[] heapa;

public int size(){

return heapa.length;

}

public int[] inPlaceHeapSort(int[] array){

heapa=array;

int size=heapa.length;

for (int i = size/2-1; i >= 0; i--) {

heap(size, i);

}

for (int i = size-1; i >=0; i--) {

int t =heapa[i];

heapa[i]=heapa[0];

heapa[0]=t;

heap(i,0);

}

return heapa;

}

private void heap(int size, int root){

int max = root;

int l= root\*2+1;

int r= root\*2+2;

//finding the largest of the 3 numbers within the heap

if(!(l>=size) && heapa[l]>heapa[max])

max=l;

if(!(r>=size) && heapa[r]>heapa[max])

max=r;

int t = heapa[max];

heapa[max]= heapa[root];

heapa[root]=t;

//to cheack its max

if (max != root)

heap(size, max);

}

public int removeMax(){

int[] t= new int[heapa.length-1];

for (int i = 0; i < t.length; i++) {

t[i]=heapa[i];

}

int out = heapa[heapa.length-1];

heapa=t;

return out;

}

}

public class CSC310\_Hw6\_2 {

public static void main(String[] args) {

// simple menu for easy use

Scanner input = new Scanner(System.in);

String choice = "";

MaxHeap mh = new MaxHeap();

do{

System.out.println("1. Run Demo");

System.out.println("2. Make heap");

System.out.println("3. remove max");

System.out.println("4. Empty heap");

System.out.println("e. Exit");

System.out.print("Pick an option from the menu above:");

choice = input.next();

switch(choice.charAt(0)){

case '1':

MaxHeap mhDemo = new MaxHeap();

System.out.println("Sequence: 9 => 7 => 5 => 2 => 6 => 4");

System.out.println("");

int[] a = {9,7,5,2,6,4};

mhDemo.inPlaceHeapSort(a);

int size=mhDemo.size();

System.out.print("Max Order: ");

//

for (int i = 0; i < size; i++) {

System.out.print(mhDemo.removeMax());

if(i < size-1){

System.out.print(" => ");

}

}

System.out.println("");

break;

case '2':

while(true){

try{

System.out.print("Enter heap Size: ");

int uinput = input.nextInt();

int[] uArray=new int[uinput];

for (int i=0; i < uArray.length; i++) {

System.out.print("Enter int: ");

uArray[i] = input.nextInt();

}

uArray= mh.inPlaceHeapSort(uArray);

for (int i = 0; i < uArray.length; i++) {

System.out.print(uArray[i]);

if(i < uArray.length-1)

System.out.print(" ");

}

break;

}catch(Exception e){

break;

}

}

break;

case '3':

System.out.println("Rmoved: "+ mh.removeMax());

break;

case '4':

for (int i = 0; i < mh.size(); i++) {

System.out.println(mh.removeMax());

}

break;

case 'e': case 'E':

System.out.println("Goodbye!");

break;

default:

System.out.println("You entered an invalid menu choice. Please try agin.");

break;

}

System.out.println("");

} while (choice.charAt(0)!='e' && choice.charAt(0) != 'E');

}

}

CSC310\_HW6\_3

import java.util.LinkedList;

import java.util.Scanner;

class BinHeap{

private LinkedList<Object> heap = new LinkedList<Object>();

//constructors

BinHeap(){

}

BinHeap(LinkedList<Object> l){

for (int i = 0; i < l.size(); i++) {

insert((int)l.get(i));

}

}

//public functions

public int size(){

return heap.size();

}

public void insert(int in){

heap.add(in);

upheap(heap.size()-1);

}

public boolean isEmpty(){

if(heap.size()<1)

return true;

return false;

}

public Object findMin(){

if(heap.isEmpty())

return null;

return heap.get(0);

}

public Object delMin(){

if(heap.isEmpty())

return null;

swap(0,heap.size()-1);

Object min = heap.removeLast();

downheap(0);

return min;

}

public void buildHeap(LinkedList<Object> l){

for (int i = 0; i < heap.size(); i++) {

heap.remove(0);

}

for (int i = 0; i < l.size(); i++) {

insert((int)l.get(i));

}

}

//Private functions

private int parent(int i){

return (i-1)/2;

}

private int left(int i){

return (i\*2)+1;

}

private int right(int i){

return (i\*2)+2;

}

private boolean hasLeft(int i){

return left(i)<heap.size()-1;

}

private boolean hasRight(int i){

return left(i)<heap.size()-1;

}

private void swap(int j, int i){

int t= (int)heap.get(i);

heap.set(i, heap.get(j));

heap.set(j, t);

}

private void upheap(int i){

int parent = parent(i);

if((i>0) && ((int)heap.get(i)< (int)heap.get(parent))){

swap(i,parent);

upheap(parent);

}

}

private void downheap(int i){

if(hasLeft(i)){

int left=left(i);

int min=left;

if(hasRight(i)){

int right=right(i);

if(((int)heap.get(right))<((int)heap.get(left))){

min=right;

}

}

if((int)heap.get(min)<(int)heap.get(i)){

swap(i,min);

downheap(min);

}

}

if(heap.size()==2 && ((int)heap.get(1))<((int)heap.get(0)))

swap(0,1);

}

}

public class CSC310\_Hw6\_3 {

public static void main(String[] args) {

BinHeap bhDemo1 = new BinHeap();

bhDemo1.insert(5);

bhDemo1.insert(7);

bhDemo1.insert(3);

bhDemo1.insert(11);

System.out.println(bhDemo1.delMin()); //3

System.out.println(bhDemo1.delMin()); //5

System.out.println(bhDemo1.delMin()); //7

System.out.println(bhDemo1.delMin()); //11

System.out.println("");

System.out.println("IsEmpty: "+bhDemo1.isEmpty()); //true

bhDemo1.insert(5);

bhDemo1.insert(7);

bhDemo1.insert(3);

bhDemo1.insert(11);

System.out.println("IsEmpty: "+bhDemo1.isEmpty()); //false

System.out.println("");

System.out.println("Find min: "+bhDemo1.findMin()); //3

System.out.println("Delete min: "+bhDemo1.delMin()); //3

System.out.println("Find min: "+bhDemo1.findMin()); //5

System.out.println("");

System.out.println("Size: "+bhDemo1.size()); //3

System.out.println("Delete min: "+bhDemo1.delMin()); //5

System.out.println("Size: "+bhDemo1.size()); //2

System.out.println("");

LinkedList<Object> l1 = new LinkedList<Object>();

l1.add(10);

l1.add(4);

l1.add(12);

l1.add(6);

l1.add(2);

l1.add(7);

l1.add(1);

bhDemo1.buildHeap(l1);

System.out.println(bhDemo1.delMin()); //1

System.out.println(bhDemo1.delMin()); //2

System.out.println(bhDemo1.delMin()); //4

System.out.println(bhDemo1.delMin()); //6

System.out.println(bhDemo1.delMin()); //7

System.out.println(bhDemo1.delMin()); //10

System.out.println(bhDemo1.delMin()); //12

System.out.println("");

LinkedList<Object> l2 = new LinkedList<Object>();

l2.add(9);

l2.add(3);

l2.add(11);

l2.add(5);

l2.add(1);

l2.add(6);

l2.add(0);

BinHeap bhDemo2 = new BinHeap(l2);

System.out.println(bhDemo2.delMin()); //0

System.out.println(bhDemo2.delMin()); //1

System.out.println(bhDemo2.delMin()); //3

System.out.println(bhDemo2.delMin()); //5

System.out.println(bhDemo2.delMin()); //6

System.out.println(bhDemo2.delMin()); //9

System.out.println(bhDemo2.delMin()); //11

}

}