**《数据结构》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2018级物联网工程专业01班** | | | **姓名** | **邓露** |
| **实验题目** | **有理数的堆排序** | | | | | |
| **实验时间** | **2019.11.18** | | **实验地点** | **DS1422** | | |
| **实验成绩** |  | | **实验性质** | **□验证性** √**设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  1. 掌握最小堆的基本概念，堆的基本运算以及堆排序的流程  2. 掌握有理数类的定义及逻辑运算规则 | | | | | | |
| 二、实验项目内容  1. 完成有理数的类定义以及有理数逻辑运算函数  class Rational{  friend bool operator<(const Rational& r1, const Rational& r2) {}  friend bool operator<=(const Rational& r1, const Rational& r2) {}  friend bool operator>(const Rational& r1, const Rational& r2) {}  friend bool operator>=(const Rational& r1, const Rational& r2) {}  friend bool operator==(const Rational& r1, const Rational& r2) {}  friend bool operator!=(const Rational& r1, const Rational& r2) {}  public:  int N; //分子  int D; //分母， 要求大于0    Rational() {} //default constructor  Rational(int n){} //constructor for integer value  Rational(int n, int d) {} //normal constructor  Rational(const Rational& r){} //copy constructor  Rational& operator=(const Rational& r) {} // assignment override  }；  2. 创建有理数的最小堆，实现siftdown, siftup, insert等功能  3． 实现基于最小堆的堆排序，按从小到大的顺序输出有理数  4． 为在线测评系统检测程序的运行，对程序文档及IO做如下规范：  （1）所有类、函数及主程序都写在一个**单cpp文档**里，不能有其他include用的.h或.cpp文档  （2）程序不能输出任何提示用的字符串  （3）输入： 第一行包含一个整数T (1≤T≤105)；接下来的T行，每一行有两个整数n, d (|n|<103, 0<d<103)，用空格隔开，表示输入的有理数的分子和分母。  （4）输出：第一行输出有理数的最小堆序列，第二行输出从小到大排序后的序列。  （5）输出的每个有理数必须规约，以n/d的形式输出，其中d>0且gcd(n,d)=0；如果d=1或n=0则直接输出n  （例子）  输入：  5 //有理数个数  3 2 //第1个有理数 3/2  1 3 //第2个有理数 1/3  4 2 //第3个有理数 4/2  12 10 //第4个有理数 12/10  4 6 //第5个有理数 4/6  输出：  1/3 2/3 2 6/5 3/2 //第一行 最小堆  1/3 2/3 6/5 3/2 2 //第二行 排序结果 | | | | | | |
| 三、实验过程或算法（源程序）  #include<iostream>  #include<stdio.h>  using namespace std;  template<typename E>  void swap(E A[],int i,int j)  {  E temp = A [i];  A [i] = A [j];  A [j] = temp;  }  //Heap class  template <typename E,typename Comp> class heap{  private:  E\* Heap;  int maxsize;  int n;  void siftdown(int pos) {  while(!isLeaf(pos)) {  int j=leftchild(pos);  int rc=rightchild(pos);  if((rc<n)&& Comp::prior(Heap[rc],Heap[j]))  j=rc;  if(Comp::prior(Heap[pos],Heap[j])) return;  swap(Heap,pos,j);  pos=j;  }  }  public:  heap(E\*h,int num,int max)  { Heap = h;n=num; maxsize=max; buildHeap(); }  int size() const  { return n; }  bool isLeaf(int pos) const  { return (pos>=n/2)&& (pos<n); }  int leftchild(int pos) const  { return 2\*pos+1; }  int rightchild(int pos) const  { return 2\*pos+2; }  int parent(int pos) const  { return (pos-1)/2; }  void buildHeap()  {  for (int i=n/2-1;i>=0;i--) siftdown(i);  }  void insert(const E& it){  if(n>=maxsize)  cout<<"Heap is full"<<endl;  else  {  int curr=n++;  Heap[curr]=it;  while((curr!=0)&&(Comp::prior(Heap[curr],Heap[parent(curr)])))  {  swap(Heap,curr,parent(curr));  curr=parent(curr);  }  }  }  E removefirst(){  if(n<=0)  cout<<"Heap is empty"<<endl;  else  {  swap(Heap,0,--n);  if(n!=0) siftdown(0);  return Heap[n];  }  }  E remove(int pos)  {  if((pos<0)||(pos>n))  cout<<"Bad position"<<endl;  else{  if(pos==(n-1)) n--;  else{  swap(Heap,pos,--n);  while((pos!=0)&&(Comp::prior(Heap[pos],Heap[parent(pos)])))  {  swap(Heap,pos,parent(pos));  pos=parent(pos);  }  if(n!=0) siftdown(pos);  }  }  return Heap[n];  }  };  class Rational  {  public:  Rational(int N,int D);  Rational(int N);  Rational();  int N;  int D;  bool operator =(const Rational&r);  friend void change(Rational&r);  friend bool operator ==(Rational&r1,Rational&r2);  friend bool operator >=(Rational&r1,Rational&r2);  friend bool operator <=(Rational&r1,Rational&r2);  friend bool operator >(Rational&r1,Rational&r2);  friend bool operator <(Rational&r1,Rational&r2);  friend bool operator!=(Rational&r1,Rational&r2);  friend int gcd(int d1,int d2);  private:  bool check(int d);  };  bool Rational::check(int d)  {  if(d==0)  return false;  else  return true;  }  int gcd(int d1,int d2)  {  if(d2==0)  return d1;  else  return gcd(d2,d1%d2);  }  void change(Rational& r)  {  if(r.D<0)  {  r.D\*=-1;  r.N\*=-1;  }  }  Rational::Rational() //default constructor  {  N=0;  D=1;  }  Rational::Rational(int n) //constructor for integer value  {  N=n;  D=1;  }  Rational::Rational(int n,int d)  {  int temp\_gcd;  bool flag;  flag=check(d);  if(flag==false)  {  exit(1);  }  temp\_gcd=gcd(n,d);  N=n/temp\_gcd;  D=d/temp\_gcd;  }  bool operator ==(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.D==r2.D && r1.N==r2.N)  return true;  else  return false;  }  bool operator !=(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.D!=r2.D|| r1.N!=r2.N)  return true;  else  return false;  }  bool operator >=(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.N\*r2.D>=r2.N\*r1.D)  return true;  else  return false;  }  bool operator <=(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.N\*r2.D<=r2.N\*r1.D)  return true;  else  return false;  }  bool operator >(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.N\*r2.D>r1.D\*r2.N)  return true;  else  return false;  }  bool operator <(Rational& r1,Rational& r2)  {  change(r1);  change(r2);  if(r1.N\*r2.D<r1.D\*r2.N)  return true;  else  return false;  }  bool Rational::operator =(const Rational& temp)  {  N=temp.N;  D=temp.D;  return true;  }  class minIntCompare {  public:  static bool prior(Rational x, Rational y) { return x<y; }  };  template<typename E>  void output1(E\*h,int num)  {  for(int i=0;i<num;i++)  {  if(h[i].D==1)  cout<<h[i].N<<" ";  else  cout<<h[i].N<<"/"<<h[i].D<<" ";  }  cout<<endl;  }  //template<typename E>  void output2(heap<Rational,minIntCompare> h,int num)  {  Rational ra2[num+10]={0};  Rational pairtwo;  for (int j=0;j<num;j++)  {  pairtwo=h.removefirst();  ra2[j]=pairtwo;  }  for (int j=0;j<num;j++)  {  if(ra2[j].D==1)  cout<<ra2[j].N<<" ";  else  cout<<ra2[j].N<<"/"<<ra2[j].D<<" ";  }  cout<<endl;  }  int main()  {  int num;  cin>>num;  Rational ra[num+10]={0};  for(int i=0;i<num;i++)  {  int n,d;  cin>>n>>d;  Rational r(n,d);  ra[i]=r;  }  heap<Rational,minIntCompare> h (ra,num,100);  output1(ra,num);  output2(h,num);  return 0;  } | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  1.OJ截图    2.运行结果  C:\Users\ASUS\AppData\Roaming\Tencent\Users\3350789950\QQ\WinTemp\RichOle\U@~`8E@}6)V3%D009N6}]`E.png | | | | | | |