**《（课程名称）》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2019级计算机科学与技术（卓越）02** | | | **姓名** | **李燕琴** |
| **实验题目** | 有理数的堆排序 | | | | | |
| **实验时间** | **2020/12/01** | | **实验地点** | **DS1401** | | |
| **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）  1、有理数的类定义主要是构造函数的拓展编写，另外有理数逻辑运算函数，通过通分比较分子判断。  2、siftdown函数，通过比较当前元素Heap[pos],和左儿子元素、右儿子元素的大小，三者最大的和pos处元素交换。Siftup函数,通过比较当前元素Heap[pos],和父元素大小，较大者替换和pos处元素交换。Insert函数，将新的元素放在最后一个位置，并通过shifup的原理将新元素放在正确的位置。  3、输出，通过一个找质数的for循环原理，找到两个元素的公约数约掉，直至gcd(n.d)为0。  ---------------------------------------------程序-----------------------------------------------------  #include<bits/stdc++.h>  using namespace std;  int MAX\_N = 1e5 + 10;  class Rational{      friend bool operator<(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 < n2;      }      friend bool operator<=(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 <= n2;      }      friend bool operator>(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 > n2;      }      friend bool operator>=(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 >= n2;      }      friend bool operator==(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 == n2;      }      friend bool operator!=(const Rational& r1, const Rational& r2) {          int n1 = r1.N \* r2.D, n2 = r2.N \* r1.D;          return n1 != n2;      }      friend ostream& operator<<(ostream &output, const Rational& r){          if(r.D==1 || r.N==0) {              output << r.N;              return output;          }          if(r.N%r.D==0) {              output << r.N/r.D;              return output;          }          // 是否有最大公约数 gcd(n,d)=0          int n = r.N, d = r.D;          for (int i = 2; i <= sqrt(min(n, d));i++){              if(n%i==0 && d%i==0) {                  n /= i;                  d /= i;              }          }          output << n<<"/"<< d;          return output;      }  public:      int N; //分子      int D; //分母， 要求大于0      Rational() {}  //default constructor      Rational(int n){ //constructor for integer value          this->N = n;          this->D = 1;      }      Rational(int n, int d) {          this->N = n;          this->D = d;      } //normal constructor      Rational(const Rational& r){          this->N = r.N;          this->D = r.D;      } //copy constructor      Rational& operator=(const Rational& r) {          this->N = r.N;          this->D = r.D;          return \*this;      } // assignment override  };  class Comp{  public:      static bool prior(const Rational& r1, const Rational& r2){          return r1 < r2;      }  };  template<typename E,typename Comp>  class heap{      private:          E \*Heap;          int maxsize; // 堆的最大规格          int n; // 元素个数          void siftdown(int pos){ // 将Pos放入正确的位置中              while(!isLeaf(pos)){                  int j = leftchild(pos);                  int rc = rightchild(pos);                  if((rc<n) && Comp::prior(Heap[rc],Heap[j]))                      j = rc; // 把j赋值给大的那个数                  if(Comp::prior(Heap[pos],Heap[j]))                      return; // 看看pos能否坐稳那个位置，是则返回，否则换位置                  swap(Heap[pos], Heap[j]);                  pos = j; // move down 继续往下查看              }          }          void siftup(int pos){ // 跟他上司比              while(true){                  if(pos==0) return;                  int p = parent(pos);                  if(Comp::prior(Heap[pos],Heap[p]))                      return;                  swap(Heap[pos], Heap[p]);                  pos = p; // move down 继续往下查看              }          }      public:          heap(E\*h,int num,int max){              Heap = h;              n = num;              maxsize = max;              buildHeap();          }          int size() const { return n; }          int isLeaf(int pos) const { return (pos>=n/2) && (pos<n); }          int leftchild(int pos) const { return 2\*pos+1; } // root从0开始          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--) // i = n/2之后的都将是叶结点                  siftdown(i); // 从最下面第一个分支结点往上看，找到合适的位置          }          void insert(const E&it){              if(n > maxsize){                  cout << "Heap is full";                  return;              }              int curr = n++;              Heap[curr] = it;              // 先插入最后一个，不断和父节点比较，若true则代替父节点，反之则退出循环              while((curr!=0 && Comp::prior(Heap[curr],Heap[parent(curr)]))){                  swap(Heap[curr],Heap[parent(curr)]);                  curr = parent(curr); // 代替父节点              }  // 方法二： shilfup(curr);          }          E removefirst(){              // Assert(n > 0, "Heap is empty");              n--;              swap(Heap[0],Heap[n]); // 把最后一个点代替first点              if(n!=0)                  siftdown(0);              return Heap[n]; // 之前把first元素和最后一个元素交换了          }          E remove(int pos){ // 移除某个特定位置的点              // Assert((pos >= 0) && (pos < n), "Position is not in the heap.");              if(pos == (n-1))                  n--;              else{                  n--;                  swap(Heap[pos],Heap[n]);                  while((pos!=0) && (Comp::prior(Heap[pos],Heap[parent(pos)]))){                      swap(Heap[pos], Heap[parent(pos)]); // 如果pos的父节点坐不稳那个位置                      pos = parent(pos); // pos换到父节点那个位置                  }                  if(n!=0)                      siftdown(pos); // 把pos放在合适的位置              }              return Heap[n]; // 返回pos那个要删除的数字          }  };  int main(){      int N;      cin >> N;      Rational num[MAX\_N];      Rational re[MAX\_N];      heap<Rational,Comp> h(num, N, MAX\_N);      for (int i = 0; i < N;i++){          int n, d;          cin >> n >> d;          Rational temp(n,d);          num[i] = temp;          re[i] = temp;      }      h.buildHeap();      for (int i = 0; i < N;i++){          // print(num[i]);          cout<<num[i]<<"  ";      }      cout << endl;      for (int i = 0; i < N;i++){          if(i!=0) cout<<"  ";          cout << h.removefirst();      }      return 0;  } | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  **1、实验结果**  测试样例一：    测试样例二：    **2、程序调试过程**  有理数的逻辑运算函数，最开始的思路是计算出n/d的double值，通过min = 1e-6来去除浮点数运算误差，但是这种复杂化了；后来是直接通过分母通分，因为N\*D<1e^6，在int值范围内，可以直接比较r1.N \* r2.D 与 r2.N \* r1.D的大小，顺利解决；有理数的输出，需要找到最大公约数，最开始没有处理这块，但是后来通过找质数的for循环解决了。因为之前写了一遍Heap的程序，这次实验完成的比较顺利。 | | | | | | |

备注：

1、教师在布置需撰写实验报告的实验前，应先将报告书上的“实验题目”、“实验性质”、“实验目的”、“实验项目内容”等项目填写完成，然后再下发给学生。

2、教师在布置需撰写报告的实验项目时，应告知学生提交实验报告的最后期限。

3、学生应按照要求正确地撰写实验报告：

* 1. 在实验报告上正确地填写“实验时间”、“实验地点”等栏目。
  2. 将实验所涉及的源程序文件内容（实验操作步骤或者算法）填写在“实验过程或算法（源程序）”栏目中。
  3. 将实验所涉及源程序调试过程（输入数据和输出结果）或者实验的分析内容填写在“实验结果及分析和（或）源程序调试过程”栏目中。
  4. 在实验报告页脚的“报告创建时间：”处插入完成实验报告时的日期和时间。
  5. 学生将每个实验完成后，按实验要求的文件名通过网络提交（上载）到指定的服务器所规定的共享文件夹中。每个实验一个电子文档，如果实验中有多个电子文档（如源程序或图形等），则用WinRAR压缩成一个压缩包文档提交，压缩包文件名同实验报告文件名（见下条）。
  6. 提交的实验报告电子文档命名为：“年级（两位数字不要“级”字）专业（缩写：计算机科学与技术专业（计科）、网络工程专业（网络）、信息安全专业（信息）、物联网工程（物联网））班级（两位数字）学号（八位数字）姓名实验序号（一位数字）．doc。如学号为20155676、年级为2015级、专业为“计算机科学与技术”专业、班级为“02班”、姓名为“王宇”的学生，完成的第一次实验命名为： 15计科02班20155676王宇1．Doc，以后几次实验的报告名称以此类推。

4、教师（或助教）在评价学生实验时，应根据其提交的其他实验相关资料（例如源程序文件等）对实验报告进行仔细评价。评价后应完成的项目有:

1. 在“成绩”栏中填写实验成绩。每个项目的实验成绩按照总分10评分。
2. 在“教师评价”栏中用符号标注评价项目结果（用√表示正确，用×表示错误，用≈表示 半对半错）。
3. 在“教师评价”栏中“评价教师签名”填写评价教师（或助教）姓名。将评价后的实验报告转换为PDF格式文件归档。
4. 课程实验环节结束后，任课教师将自己教学班的实验报告文件夹进行清理。在提交文件夹中，文件总数为实验次数×教学班学生人数（如，教学班人数为90人，实验项目为5，其文件数为：90×5=450）。任课教师一定要认真清理，总数相符，否则学生该实验项目不能得分。最后将学生提交的实验报告刻光盘连同实验成绩一起放入试卷袋存档。