# 20221008 数据结构与算法 解题报告

## 检查一个序列是否构成堆

检查数列是否满足堆的性质：

$$val\_i<val\_{i\*2}\ \and\ val\_i<val\_{i\*2+1}\\
val\_i>val\_{i\*2}\ \and\ val\_i>val\_{i\*2+1}$$

遍历检查即可。

#include <bits/stdc++.h>  
using namespace std;  
  
bool cmp1(int a, int b) {return a <= b;}  
bool cmp2(int a, int b) {return a >= b;}  
  
bool judge(vector<int> &val, bool(\*cmp)(int, int)) {  
 int size = val.size();  
 for (int i = 0; i < size; i++) {  
 if (i \* 2 + 1 < size && !cmp(val[i], val[i \* 2 + 1])) return false;  
 if (i \* 2 + 2 < size && !cmp(val[i], val[i \* 2 + 2])) return false;  
 }  
 return true;  
}  
  
int main(int argc, char const \*argv[]) {  
 // freopen("init.in", "r", stdin);  
 for (int n; cin >> n;) {  
 vector<int> val(n); for (auto &i : val) cin >> i;  
 if (judge(val, cmp1) && judge(val, cmp2)) cout << "both" << endl;  
 else if (judge(val, cmp1)) cout << "min heap" << endl;  
 else if (judge(val, cmp2)) cout << "max heap" << endl;  
 else cout << "no" << endl;  
 }  
 return 0;  
}

## 奖学金

创建类Student储存信息，利用堆（priority\_queue）取前五输出即可。

#include <bits/stdc++.h>  
using namespace std;  
  
struct Student {  
 int a, b, c, tot, id;  
 Student(int \_a, int \_b, int \_c, int \_id) : a(\_a), b(\_b), c(\_c), tot(\_a + \_b + \_c), id(\_id) {}  
 friend bool operator < (const Student &p, const Student &q) {  
 if (p.tot == q.tot)  
 if (p.a == q.a) return p.id > q.id;  
 else return p.a < q.a;  
 else return p.tot < q.tot;  
 }  
};  
  
int main(int argc, char const \*argv[]) {  
 // freopen("init.in", "r", stdin);  
 for (int n; cin >> n;) {  
 priority\_queue<Student> hep;  
 for (int i = 0; i < n; i++) {  
 int a, b, c; cin >> a >> b >> c;  
 hep.push(Student(a, b, c, i + 1));  
 }  
 for (int i = 0; i < 5; i++) {  
 auto tmp = hep.top(); hep.pop();  
 cout << tmp.id << " " << tmp.tot << endl;  
 }  
 }  
 return 0;  
}

## heap

Push：将新元素插入到堆底，进行shiftUp操作进行维护。

Pop：将堆顶元素与堆底元素swap，弹出堆底（n--），对堆顶进行shiftDown操作进行维护。

#include <bits/stdc++.h>  
#include "heap.h"  
using namespace std;  
  
#define INF 1E9  
  
void heap::push(int val) {  
 int pos; h[pos = ++n] = val;  
 for (; pos != 1 && h[pos] < h[pos / 2]; pos /= 2) swap(h[pos], h[pos / 2]);  
}  
  
void heap::pop() {  
 if (!n) return;  
 swap(h[n], h[1]); n--; int pos = 1;  
 for (; pos \* 2 <= n;) {  
 int c = pos \* 2;  
 if (c + 1 <= n && h[c + 1] < h[c]) c++;  
 if (h[c] < h[pos]) {swap(h[c], h[pos]); pos = c;}  
 else break;  
 }  
}