二分法

使用条件

> 用于查找的内容逻辑上来说要有序（从小到大或从大到小，不然一会小一会大没法根据中间值缩小区间）
>
> 查找的数量只能是1个（单纯查找，最大值的最小、最小值的最大这类）

两种写法（用哪种看题目，大部分是左闭右闭）

1.左闭右闭，每次查找的区间在**[l,r]**上：

> 循环条件：`while(l<=r)`
>
> 中间值判断：
>
> `if(mid>target) -> r=mid-1 -> 接下来查找的区间[l,mid-1]`
>
> `if(mid<target) -> l=mid+1 -> 接下来查找的区间[mid+1,r]`

2.左闭右开，每次查找的区间在**[l,r)**上

循环条件：`while(l<r)`

中间值判断：

`if(mid>target) -> r=mid -> 接下来查找的区间[l,mid]`

`if(mid<target) -> l=mid+1 -> 接下来查找的区间[mid+1,r]`

用STL函数实现二分（🥳）

binary_search():

头文件：#include<algorithm>(当然万能头也ooook)

函数：binary_search(arr[],arr[]+size,index)

参数说明：arr[]:数组首地址 size：数组元素个数 index：需要查找的值

功能：在数组中二分查找选定元素，找到了返回为真，否则返回值为假（没法返回下标差评！！）

lower_bound:

模板：lower_bound(arr[],arr[]+size,index)

功能：在first和last前闭后开区间二分查找，返回**大于或等于**index的第一个元素位置（注意是**地址**），如果找不到就返回last位置，**如果由多个相同的index返回第一个index下标**

用法：返回值减去数组首地址就可以得到下标

upper_bound:

模板：upper_bound(arr[],arr[]+size,index)

功能：在first和last前闭后开区间二分查找，返回**大于**index的第一个元素位置（注意是**地址**），如果找不到就返回last位置，如果**多个相同index返回最后一个index下标+1**

用法：返回值减去数组首地址就可以得到下标

求指定元素出现次数：

upper_bound(arr[],arr[]+size,index)-lower_bound(arr[],arr[]+size,index) 长度为size的有序数组arr中元素index的个数

用法举例：

```
#include <algorithm>
#include <iostream>
using namespace std;
int main()
{
    int a[]={1,2,3,4,5,7,8,9};
    printf("%d",lower_bound(a,a+8,6)-a);
    return 0;
}
//输出:5

#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
int main()
{
    vector<int> A;
    A.push_back(1);
    A.push_back(2);
    A.push_back(3);
    A.push_back(4);
    A.push_back(5);
    A.push_back(7);
    A.push_back(8);
    A.push_back(9);
    int pos = lower_bound(A.begin() , A.end() , 6)-A.begin();
    cout << pos << endl;
    return 0;
 }
//输出:5
```

例题（感觉主要还是要从例题去掌握二分法）

【T1】

P1024 [NOIP2001 提高组] 一元三次方程求解

提交答案 加入题单

题目描述

复制Markdown 展开

有形如：$ax^3+bx^2+cx+d=0$ 这样的一个一元三次方程。给出该方程中各项的系数（a,b,c,d 均为实数），并约定该方程存在三个不同实根（根的范围在 -100 至 100 之间），且根与根之差的绝对值 ≥ 1。要求由小到大依次在同一行输出这三个实根(根与根之间留有空格)，并精确到小数点后 2 位。

提示：记方程 $f(x)=0$，若存在 2 个数 x_1 和 x_2，且 $x_1<x_2$，$f(x_1)\times f(x_2)<0$，则在 (x_1,x_2) 之间一定有一个根。

输入格式

一行，4 个实数 a,b,c,d。

输出格式

一行，3 个实根，从小到大输出，并精确到小数点后 2 位。

输入输出样例

输入 #1 复制

```
1 -5 -4 20
```

输出 #1 复制

```
-2.00 2.00 5.00
```

说明/提示

【题目来源】

NOIP 2001 提高组第一题

```cpp
#include<bits/stdc++.h>
using namespace std;
double a,b,c,d;
double check(double x)//算出对应的y
{
    double y;
    y=a*x*x*x+b*x*x+c*x+d;
    return y;
}
int main()
{
    cin>>a>>b>>c>>d;
    for(int i=-100;i<100;i++)
    {
        double l=i,r=(i+1);
        double mid;
```

```
        if(check(l)==0)cout<<setprecision(2)<<setiosflags(ios::fixed)<<l<<' ';//考虑左端点为解的情况
        else if(check(l)*check(r)<0)//说明l，r之间有解
        {
            while((r-l)>=0.001)//注意精度设置，保留两位小数要差值小于0.01
            {
                mid=(l+r)/2;//开始二分
                if(check(mid)*check(l)<=0)r=mid;
                else l=mid;
            }
            cout<<setprecision(2)<<setiosflags(ios::fixed)<<l<<' ';
        }
    }
}
```

【T2】

P1824 进击的奶牛

提交答案 加入题单

题目描述

复制Markdown 展开

Farmer John 建造了一个有 $N(2 \leq N \leq 100000)$ 个隔间的牛棚，这些隔间分布在一条直线上，坐标是 $x_1,...,x_N$ $(0 \leq x_i \leq 1000000000)$。

他的 $C(2 \leq C \leq N)$ 头牛不满于隔间的位置分布，它们为牛棚里其他的牛的存在而愤怒。为了防止牛之间的互相打斗，Farmer John 想把这些牛安置在指定的隔间，所有牛中相邻两头的最近距离越大越好。那么，这个最大的最近距离是多少呢?

输入格式

第 1 行：两个用空格隔开的数字 N 和 C。

第 $2 \sim N+1$ 行：每行一个整数，表示每个隔间的坐标。

输出格式

输出只有一行，即相邻两头牛最大的最近距离。

输入输出样例

输入 #1 复制

```
5 3
1
2
8
4
9
```

输出 #1 复制

```
3
```

```cpp
#include<bits/stdc++.h>//用二分法枚举答案
using namespace std;
long arr[100001];
long n,c,mid;
long ans=0;
bool check(long m)//检查当距离为mid时符不符合要求
{
        long cnt=0;//实际能装几头牛(cnt--计数器)
        long d;
        long l=arr[0];//初始化隔间为arr[0]
        for(int i=1;i<n;i++)
        {
                d=arr[i]-l;//相邻隔间距离
                if(d>=m)//如果两个相邻隔间距离大于m就符合要求
```

```
                {
                        cnt++;
                        l=arr[i];//更新隔间号
                }
        }
        if(cnt>=c-1)return true;//实际能装的比需要的多就符合题意
        else return false;
}
int main()
{
        cin>>n>>c;
        for(int i=0;i<n;i++)
        cin>>arr[i];
        sort(arr,arr+n);//用二分法之前先排序
        long l=arr[0],r=arr[n-1];//为啥不能l=0, r=n-1??--因为要求的是具体的值不是下标
        while(l<=r)//二分法
        {
                long mid=(l+r)/2;
                if(check(mid))
                {
                        ans=mid;
                        l=mid+1;
                }
                else r=mid-1;
        }
        cout<<ans;
}
```

【T3】

P1873 [COCI 2011/2012 #5] EKO / 砍树

提交答案 加入题单

题目描述

复制Markdown 展开

伐木工人 Mirko 需要砍 M 米长的木材。对 Mirko 来说这是很简单的工作，因为他有一个漂亮的新伐木机，可以如野火一般砍伐森林。不过，Mirko 只被允许砍伐一排树。

Mirko 的伐木机工作流程如下：Mirko 设置一个高度参数 H（米），伐木机升起一个巨大的锯片到高度 H，并锯掉所有树比 H 高的部分（当然，树木不高于 H 米的部分保持不变）。Mirko 就得到树木被锯下的部分。例如，如果一排树的高度分别为 $20, 15, 10$ 和 17，Mirko 把锯片升到 15 米的高度，切割后树木剩下的高度将是 $15, 15, 10$ 和 15，而 Mirko 将从第 1 棵树得到 5 米，从第 4 棵树得到 2 米，共得到 7 米木材。

Mirko 非常关注生态保护，所以他不会砍掉过多的木材。这也是他尽可能高地设定伐木机锯片的原因。请帮助 Mirko 找到伐木机锯片的最大的整数高度 H，使得他能得到的木材至少为 M 米。换句话说，如果再升高 1 米，他将得不到 M 米木材。

输入格式

第 1 行 2 个整数 N 和 M，N 表示树木的数量，M 表示需要的木材总长度。

第 2 行 N 个整数表示每棵树的高度。

输出格式

1 个整数，表示锯片的最高高度。

输入输出样例

输入 #1 复制

```
4 7
20 15 10 17
```

输出 #1 复制

```
15
```

输入 #2 复制

```
5 20
4 42 40 26 46
```

输出 #2 复制

```
36
```

说明/提示

对于 100% 的测试数据，$1 \le N \le 10^6$，$1 \le M \le 2 \times 10^9$，树的高度 $< 10^9$，所有树的高度总和 $> M$。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n,m,a[1000005];
int main()
{
    long long sum;
    long long l=0,r=0;
    cin>>n>>m;
    for(long long i=0;i<n;i++)
    {
        cin>>a[i];
        if(a[i]>r)r=a[i];//选出最长的数，当作右端点
    }                    //取右端点的意思就是全砍光

    while(l<=r)
    {
        long long mid=(l+r)/2;
        sum=0;
        for(long long i=0;i<n;i++)//假设mid就是最大高度
        {                         //然后看这个高度下看下来的够不够
            if(a[i]>mid)sum+=a[i]-mid;
        }
        if(sum<m)r=mid-1;
        else l=mid+1;
    }
    cout<<r;
}
```

【T4】

P2249 【深基13.例1】查找

提交答案 加入题单

题目描述

复制Markdown 展开

输入 n 个不超过 10^9 的单调不减的（就是后面的数字不小于前面的数字）非负整数 $a_1, a_2, \ldots, a_n$，然后进行 m 次询问。对于每次询问，给出一个整数 q，要求输出这个数字在序列中第一次出现的编号，如果没有找到的话输出 -1 。

输入格式

第一行 2 个整数 n 和 m，表示数字个数和询问次数。

第二行 n 个整数，表示这些待查询的数字。

第三行 m 个整数，表示询问这些数字的编号，从 1 开始编号。

输出格式

输出一行，m 个整数，以空格隔开，表示答案。

输入输出样例

输入 #1 复制

```
11 3
1 3 3 3 5 7 9 11 13 15 15
1 3 6
```

输出 #1 复制

```
1 2 -1
```

说明/提示

数据保证，$1 \le n \le 10^6$，$0 \le a_i, q \le 10^9$，$1 \le m \le 10^5$

本题输入输出量较大，请使用较快的 IO 方式。

```cpp
#include<bits/stdc++.h>
using namespace std;

long n,m,q;
int main()//p.s这题好像可以直接用lower_bound做
{
    cin>>n>>m;
    long arr[n];
    for(int i=0;i<n;i++)
    cin>>arr[i];
    long q;
    while(m>0)
```

```
    {
        cin>>q;
        long l=0,r=n;
        long mid;
        int b=-1;
        while(l<=r)//这里要取等(?)
        {
                mid=(l+r)/2;//第一次二分法求符合要求的一个元素
                if(arr[mid]==q)
                {
                        if(mid>0&&arr[mid-1]==q)
                        {
                                long high=mid,low=0;
                                while(low<high)//这里不取等(?)
                                {
                                        long middle=(low+high)/2;//第二次二分法求第一个值为所求数的元素
                                        if(arr[middle]==q)high=middle;
                                        if(arr[middle]<q)low=middle+1;//跟前面一次二分法有区别
                                }
                                cout<<high+1<<' ';//注意输出的时候+1（因为数组下表从0起）
                        }
                        else cout<<mid+1<<' ';
                        b=1;
                        break;//记得要退出循环
                }
                else if(arr[mid]>q)r=mid-1;
                else if(arr[mid]<q) l=mid+1;
        }
        if(b==-1)cout<<"-1 ";//q不在数组里的情况
        m--;
    }
}
```

【T5】

题目背景

复制Markdown 展开

一年一度的"跳石头"比赛又要开始了!

题目描述

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

输入格式

第一行包含三个整数 L, N, M，分别表示起点到终点的距离，起点和终点之间的岩石数，以及组委会至多移走的岩石数。保证 $L \geq 1$ 且 $N \geq M \geq 0$。

接下来 N 行，每行一个整数，第 i 行的整数 $D_i(0 < D_i < L)$，表示第 i 块岩石与起点的距离。这些岩石按与起点距离从小到大的顺序给出，且不会有两个岩石出现在同一个位置。

输出格式

一个整数，即最短跳跃距离的最大值。

输入输出样例

输入 #1 复制

```
25 5 2
2
11
14
17
21
```

输出 #1 复制

```
4
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int d,n,m;
const int N = 5 * 1e5 + 10;
int arr[N];
bool check(int u)
{
    int cnt=0;//搬走的石头总数
    int nex=0;//下一步将要去哪
    int cur=0;//目前位置
    while(nex<n+1)//注意不能取等（还是因为终点不参与运算）
    {
        nex++;//首先让arr[0](起点)和arr[1]算距离
        if(arr[nex]-arr[cur]<u)cnt++;
        else cur=nex;
    }
    if(cnt>m)return false;//搬走的石头不能超过给的m
```

```
     return true;
}
int main()
{
    int ans;
    cin>>d>>n>>m;
    for(int i = 1; i <= n; i ++) cin>>arr[i];//注意要从1开始，因为arr[0]不能搬走（不参与函数运算）
    arr[n+1] = d;//把终点放到数组里
    int l = 0,r = d;
    while(l<=r)//典型二分法（注意l可等于r）
    {
      int mid = (l + r )/2;
        if(check(mid))
        {
            l=mid+1;
            ans = mid;
        }
        else r = mid - 1;
    }
    cout<<ans<<endl;
    return 0;
}
```

【T6】

P3382 【模板】三分法

提交答案 加入题单

题目描述

复制Markdown 展开

如题，给出一个 N 次函数，保证在范围 $[l,r]$ 内存在一点 x，使得 $[l,x]$ 上单调增，$[x,r]$ 上单调减。试求出 x 的值。

输入格式

第一行一次包含一个正整数 N 和两个实数 l,r，含义如题目描述所示。

第二行包含 $N+1$ 个实数，从高到低依次表示该 N 次函数各项的系数。

输出格式

输出为一行，包含一个实数，即为 x 的值。若你的答案与标准答案的相对或绝对误差不超过 10^{-5} 则算正确。

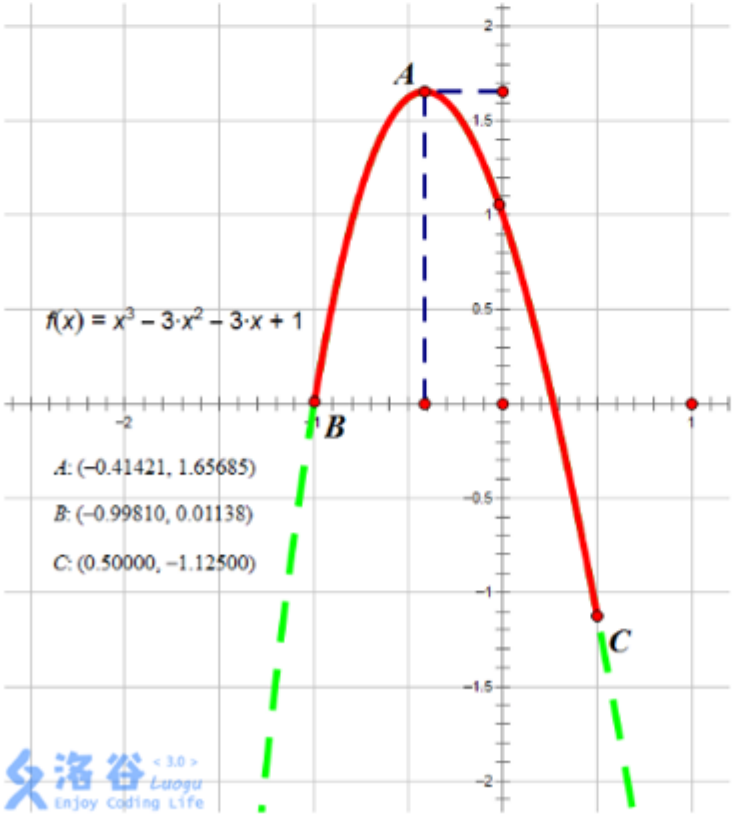
输入输出样例

输入 #1	输出 #1
3 -0.9981 0.5 1 -3 -3 1	-0.41421

说明/提示

对于 100% 的数据，$6 \le N \le 13$，函数系数均在 $[-100,100]$ 内且至多 15 位小数，$|l|,|r| \le 10$ 且至多 15 位小数。$l \le r$。

【样例解释】



如图所示，红色段即为该函数 $f(x)=x^3-3x^2-3x+1$ 在区间 $[-0.9981,0.5]$ 上的图像。

当 $x=-0.41421$ 时图像位于最高点，故此时函数在 $[l,x]$ 上单调增，$[x,r]$ 上单调减，故 $x=-0.41421$，输出 -0.41421。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define epx 1e-7
double a[100];
double n,l,r,midl,midr;
double f(double x)
{
    double num=0;
    for(int i=0;i<=n;i++)
        num+=a[i]*pow(x,n-i);//算出x对应的y
    return num;
}
int main()
{
    cin>>n>>l>>r;
    for(int i=0;i<=n;i++)
        cin>>a[i];
     while(fabs(r-l)>=epx)//三分法的精髓
    {
        midl=l+(r-l)/3.0;
        midr=r-(r-l)/3.0;
        if(f(midl)>f(midr))
        r=midr;
        else
        l=midl;
    }//三分法的精髓
    cout<<setprecision(5)<<setiosflags(ios::fixed)<<l;

}
```

【T7】

7-3 山头狙击战 分数 300

全屏浏览题目 切换布局 作者 neuqAcmClub 单位 东北大学秦皇岛分校

题目描述

小明为了掩护大部队，单枪匹马同敌人周旋，后来被敌人包围在某山头……等等，为什么怎么听怎么像狼牙山五壮士！不过不用着急，这次小明携带了足够的弹药，完全可以将涌上来的敌人一个一个干掉。小明是个神枪手，只要他的枪膛中有子弹，他就能将在他射程m（用从敌人位置到山头的直线距离算）以内的一个敌人瞬间射杀。但如果在射程内没有敌人，出于节约子弹考虑和面子问题，小明会等待敌人靠近然后射击。
正当小明为自己的强大而自我膨胀时，他忽然发现了一个致命的失误：他携带的枪是单发枪，每射出一发子弹都必须花k秒钟的时间装子弹。而凶残的敌人才不会花时间等你换子弹呢。他们始终在以1m/s的速度接近山头。而如果在一个敌人到达山头时小明无法将他击毙，那么我们可怜的小明就将牺牲在敌人的刺刀下。现在小明用心灵感应向你发出求助：要保住自己的性命并且歼灭所有敌人，小明最多只能用多少时间给枪装上一发子弹？
说明：假设一开始小明的枪中就有一发子弹，并且一旦确定一个装弹时间，小明始终会用这个时间完成子弹的装卸。希望你能帮助小明脱离险境。

输入格式

每组输入数据，第一行有两个整数n和m，（2≤n≤100,000; 1≤m≤10,000,000）n代表敌人个数，m代表小明的射程。
接下来有n行，每行一个整数mi,（1≤mi≤10,000,000），代表每个敌人一开始相对山头的距离（单位为米）。

输出格式

每组输出数据仅有一个整数，代表小明的换弹时间（单位为秒）。

样例输入

```
6 100
236
120
120
120
120
120
```

样例输出

```
25
```

```cpp
#include<bits/stdc++.h>
using namespace std;
const long Max=1e7;
long arr[Max];
long n,m;
queue<long>enemy;//用队列处理(栈应该也可以)
bool check(int x)//x就是换弹时间
{
        long cnt=0;//统计总路程
        int win=1;//标识符，失败置为0
        for(int i=0;i<n;i++) enemy.push(arr[i]);
        if(enemy.front()>m)//第一个敌人单独考虑
        cnt+=enemy.front()-m;
        enemy.pop();
        while(win)
```

```
        {
                cnt+=x;//换弹
                if(enemy.front()-cnt<0)
                {
                        win=0;
                        break;
                }       //代表敌人已经到达，失败
                if(enemy.front()-cnt>m)//如果换完弹后理他最近的敌人还没进入射程
                {
                        long time=enemy.front()-cnt-m;//等到最近的敌人刚刚进入射程花费
的时间
                        cnt+=time;//把等待时间加到总时间里
                        enemy.pop();//击杀
                }
                else//这个时候射程范围内已经有敌人
                {
                        enemy.pop();//击杀距离他最近的敌人
                }
                if(enemy.empty())break;
        }
        while(!enemy.empty())enemy.pop();//一定要记得每次判断完要清空队列
        if(win==0)return false;
        else return true;
}
int main()
{
        int ans;
        cin>>n>>m;
        for(int i=0;i<n;i++) cin>>arr[i];
        sort(arr,arr+n);//排序，方便比较
        for(int i=1;;i++)//这里也可以用二分法(🤪)
        {
                if(check(i))
                ans=i;
                else break;//只要一次不符合就能判定上一个是最大时间
        }
        cout<<ans;
}
```

【T8】

7-5 一元三次方程 分数 300

全屏浏览题目 切换布局 作者 neuqAcmClub 单位 东北大学秦皇岛分校

给定一个形如$ax^3+bx^2+cx+d=0$的一元三次方程。

已知该方程有三个不同的实数根（根与根之差的绝对值$\geq 10^{-6}$），且根范围均在$[p,q]$之间，你需要解出这个方程的三个根。

输入格式:

第一行一个整数$T(1\leq T\leq 1000)$，表示有T组数据

接下来T行，每行6个实数，分别表示a,b,c,d,p,q

数据保证：$-10^2\leq p,q\leq 10^2$，且对于$\forall x\in[p,q],-10^6\leq f(x)\leq 10^6$

输出格式:

输出三个实数，表示方程的三个解。

你的答案可以以任意顺序输出。

一个答案被认为是正确的，当且仅当其与标准答案的绝对误差不超过10^{-6}

输入样例:

在这里给出一组输入。例如:

```
1
1.000000 -5.000000 -4.000000 20.000000 -10.000000 10.000000
```

输出样例:

在这里给出相应的输出。例如:

```
-2.000000 2.000000 5.000000
```

```
#include<bits/stdc++.h>
using namespace std;
int n;
const double eps=1e-6;
double a,b,c,d,p,q;
double x1,x2;
double f(double x)//算出x对应的y
{
        double y=a*x*x*x+b*x*x+c*x+d;
        return y;
```

```
}
double find(double l,double r)//二分查找
{
        double mid=(l+r)/2;
        while(fabs(f(mid)) > eps)//不论递增递减都可以
        {
                if(f(mid)*f(r)>0)r=mid;//mid和r在同一侧，所以r缩到mid
                if(f(mid)*f(l)>0)l=mid;//mid和l在同一侧，所以l增到mid
                mid=(l+r)/2;//注意要更新mid
        }
        return mid;
}
int main()
{
        cin>>n;
        for(int i=0;i<n;i++)
        {
                cin>>a>>b>>c>>d>>p>>q;
                x1=(-b-sqrt(b*b-3*a*c))/(3*a);//找出两个极值点，分成三段单调的区间
                x2=(-b+sqrt(b*b-3*a*c))/(3*a);
                if(x1>x2)swap(x1,x2);//保证总小到大的区间查找顺序
                cout<<setprecision(6)<<setiosflags(ios::fixed)<<find(p,x1)<<'
'<<find(x1,x2)<<' '<<find(x2,q)<<endl;

        }
}
```