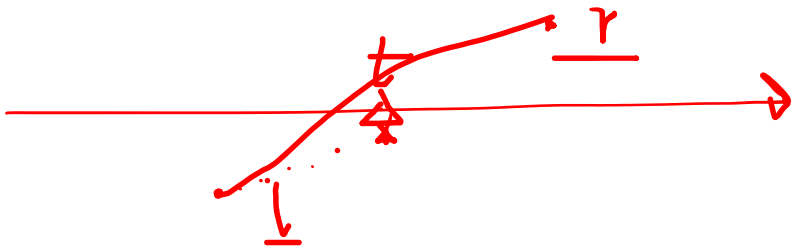
二分选讲

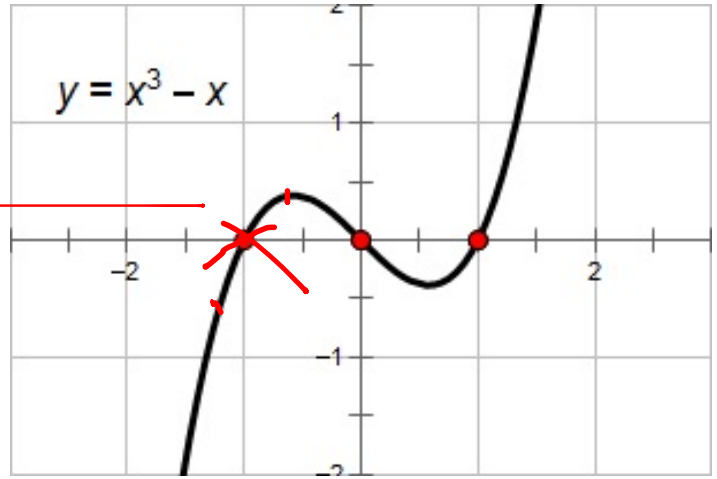
原作者：zcysky

讲师：王骏峣

二分法是什么？



- 高中数学必修一对于二分法求方程近似解的介绍是：
- 对于在区间[l,r]连续的函数f(x)，若使得f(l)*f(r)<0,则该区间内存在一变号零点（零点存在定理）
- 所以每次不断地将这个区间一分为二，每次判断两个子区间是否满足零点存在定理，即可求得零点的一个近似解。
- 但是这样只能求得一组解。
- f[l]<0, f[r]>0
- f[mid]>0->[l,mid]

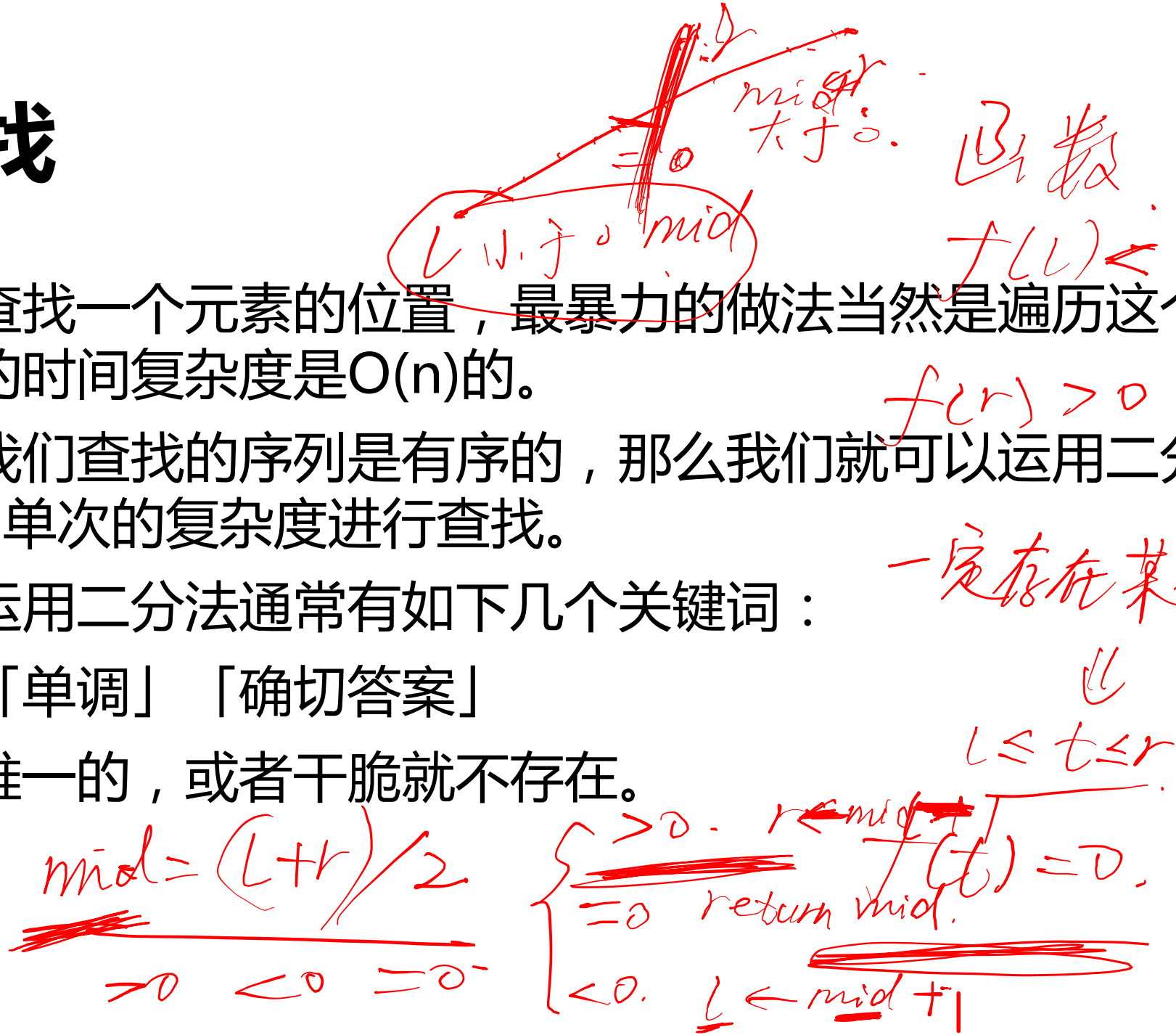


代码实现

```
inline double find(double l,double r){
    while(r-l>eps){
        double mid=(l+r)/2;
        if(f(mid)==0)return mid;
        if(f(l)*f(mid)<0)r=mid;
        else l=mid;
    }
    return l;
}
```

二分查找

- 在序列上查找一个元素的位置，最暴力的做法当然是遍历这个序列，最坏的时间复杂度是O(n)的。
- 但是如果我们查找的序列是有序的，那么我们就可以运用二分法以O(logn)单次的复杂度进行查找。
- 请注意，运用二分法通常有如下几个关键词：
- 「连续」「单调」「确切答案」
- 解通常是唯一的，或者干脆就不存在。



二分查找

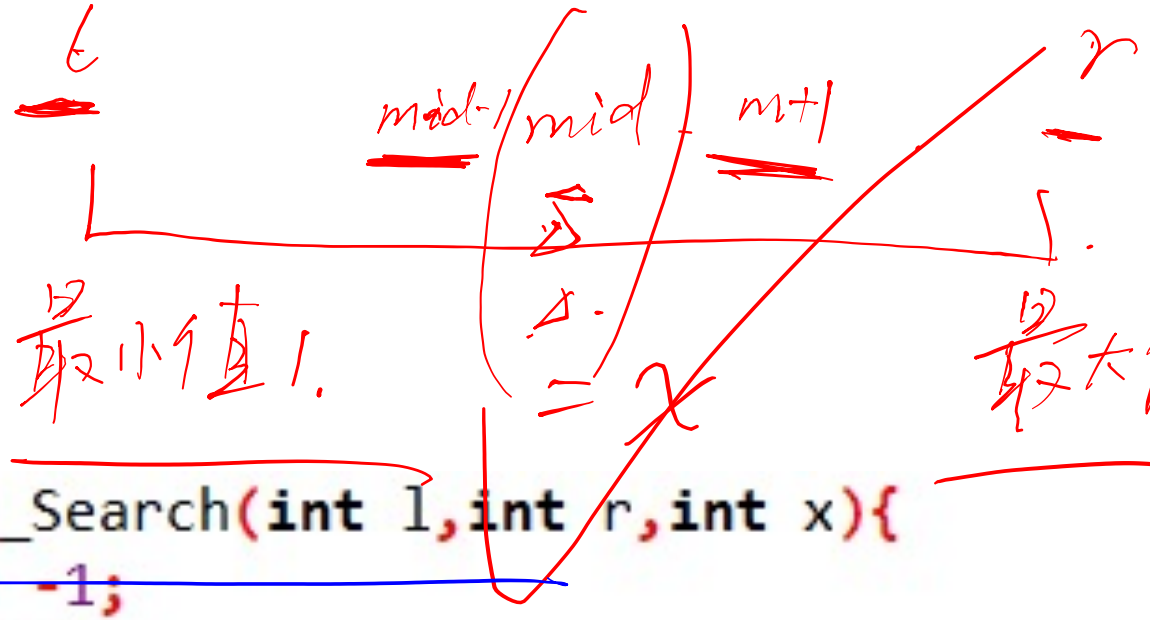
- 在序列上查找一个元素的位置，最暴力的做法当然是遍历这个序列，最坏的时间复杂度是O(n)的。
- 但是如果我们查找的序列是有序的，那么我们就可以运用二分法以O(logn)单次的复杂度进行查找。
- 请注意，运用二分法通常有如下几个关键词：
- 「连续」「单调」「确切答案」
- 解通常是唯一的，或者干脆就不存在。

非递归实现

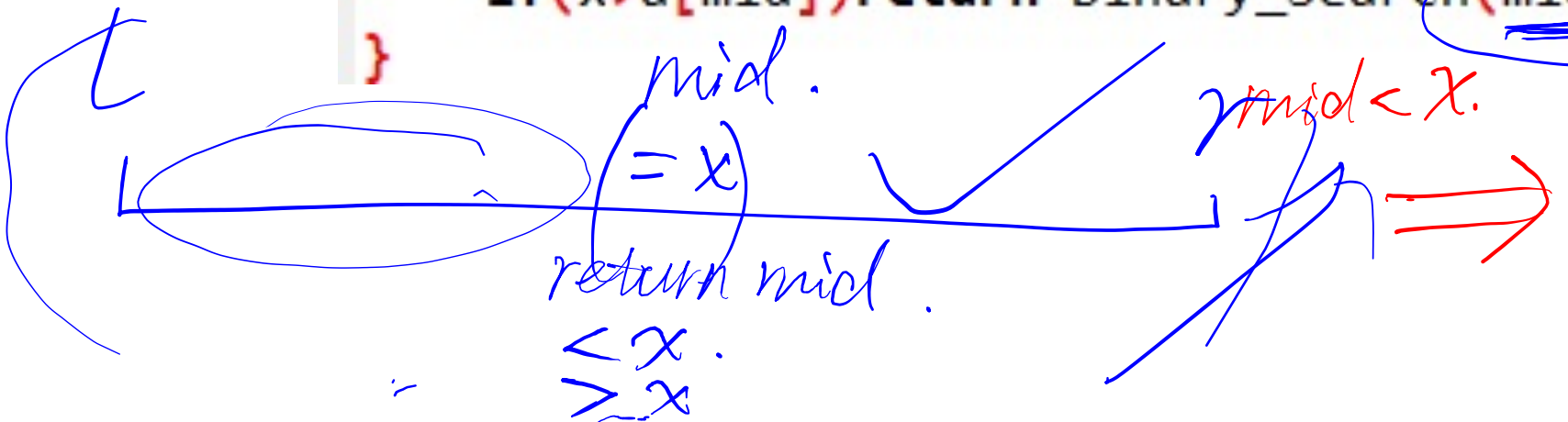
```
int Binary_Search(int l,int r,int x)//l~r的区间里找到x出现的位置
{
    while(l<=r)
    {
        int mid=(l+r)/2;
//考虑l~mid-1 mid mid+1~r
        if(a[mid]==x)//刚好mid就是x的位置
            return mid;
        if(a[mid]>x)//连mid都比x大，那么比mid大的都比x大
            r=mid-1;//l~mid-1
        else//(a[mid]<x)
            l=mid+1;//mid+1~r
    }
    //l>r了，区间不再成立
1~5 a[1],a[2],a[3],a[4],a[5]
return -1;
}
```

```
inline int Binary_Search(int l,int r,int x){
    while(l<=r){
        int mid=(l+r)>>1;
        if(a[mid]==x)return mid;
        if(x<a[mid])r=mid-1;
        else l=mid+1;
    }
    return -1;
}
```

代码实现




```
inline int Binary_Search(int l,int r,int x){
    if(l>r)return -1;
    int mid=(l+r)>>1;
    if(a[mid]==x)return mid;
    if(x<a[mid])return Binary_Search(l,mid-1,x);
    if(x>a[mid])return Binary_Search(mid+1,r,x);
}
```



猜数字题号

P2333

```
inline int Binary_Search(int l,int r,int x){
    if(l>r)return -1;
    int mid=(l+r)>>1;
    if(a[mid]==x)return mid;
    if(x<a[mid])return Binary_Search(l,mid-1,x);
    if(x>a[mid])return Binary_Search(mid+1,r,x);
}
```

```
inline int Binary_Search(int l,int r,int x){
    while(l<=r){
        int mid=(l+r)>>1;
        if(a[mid]==x)return mid;
        if(x<a[mid])r=mid-1;
        else l=mid+1;
    }
    return -1;
}
```

$10^9/2=5\times10^8/2=2.5\times10^8/2=1.25\times10^8$

$1024/2=512\to256/2=128$

check(mid)

安全 不安全

$=6.25\times10^7$

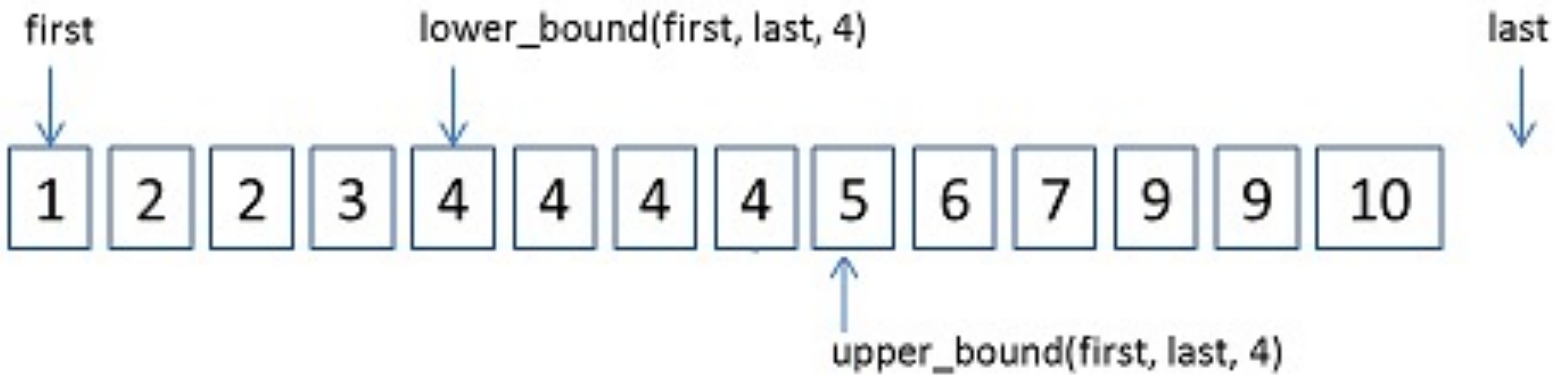
$64/2=32/2=16/2=8/2=4/2=2/2=1$

30.

STL中的二分查找

- `std::lower_bound()` 返回有序表里第一个大于等于给定值的指针或迭代器；
- `std::upper_bound()` 返回有序表里第一个大于给定值的指针或迭代器；
- `std::binary_search()` 返回给定值是否在有序表里存在。

STL中的二分查找


first
lower_bound(first, last, 4)
last
1
2
2
3
4
4
4
4
5
6
7
9
9
10
upper_bound(first, last, 4)

STL中的二分查找

- 用法举例：
- int x=lower_bound(a+1,a+n+1,y)-a-1;
- 一个非常常见的用途：离散化。

```
inline void init(){
    for(int i=1;i<=n;i++)a[i]=read(),b[i]=a[i];
    sort(b+1,b+n+1);
    for(int i=1;i<=n;i++)a[i]=lower_bound(b+1,b+n+1,a[i])-a-1;
}
```

- 可以在值域较大的情况，在保证相对大小不变的情况下缩小值域
- 这样很多值域相关的操作就可以进行，例如树状数组逆序对。

用法举例

- `int a[5] = {0, 2, 4, 6, 8};`
- `printf("%d %d %d\n", lower_bound(a, a + 5, 2) - a, upper_bound(a, a + 5, 2) - a, binary_search(a, a + 5, 2));`
- `printf("%d %d %d\n", lower_bound(a, a + 5, 3) - a, upper_bound(a, a + 5, 3) - a, binary_search(a, a + 5, 3));`
- 输出结果：

```
1 2 1
2 2 0
```

在vector上的用法

- `lower_bound(a.begin(), a.end(), 2)`
- vector的迭代器可减，也就是说可以和数组一样减去数组名来获取下标。

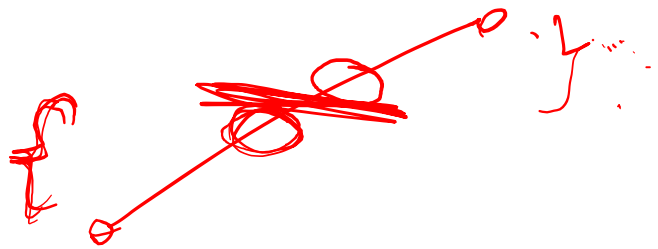
在map/set里面的用法

- `a.lower_bound(2)`
- 注意STL的迭代器不可减
- 毕竟这个本质是平衡树

二分查找的实质

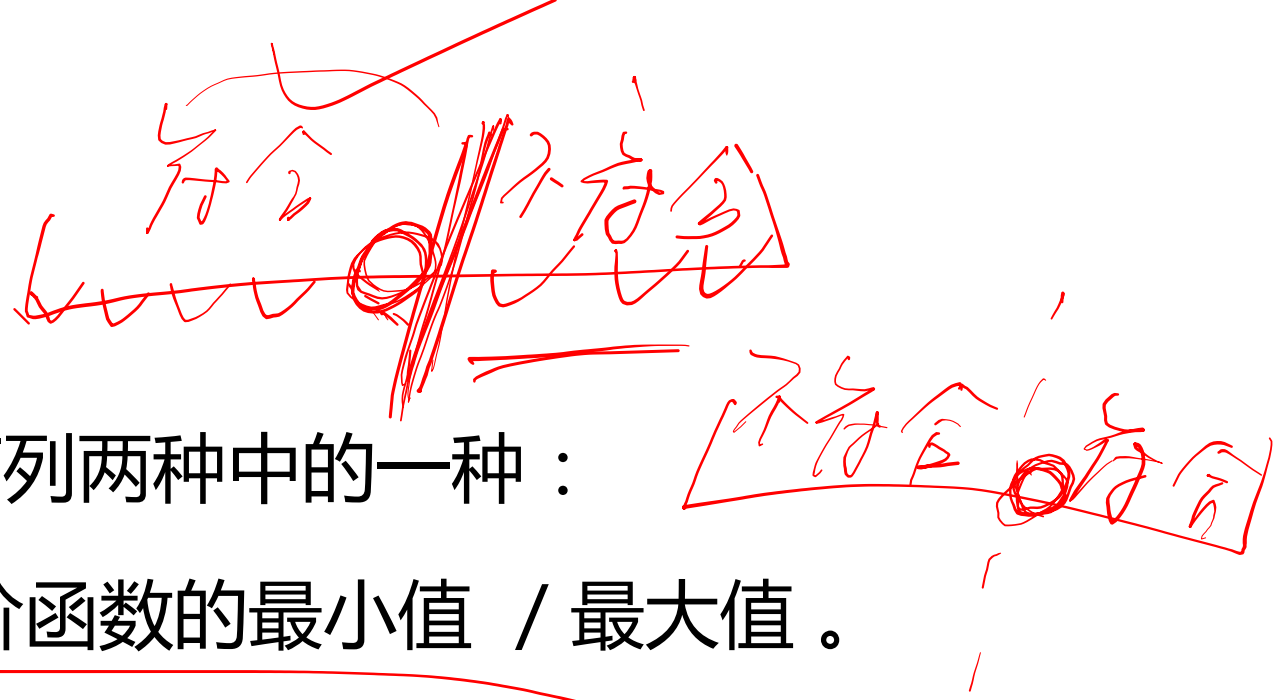
- 可以把数组看做值离散的函数
- 二分查找的实质就是定义在（定义域为整数的）函数上的二分法
- 但是相比之下，二分查找对数组有单调性的要求，并且保证可以找到答案

二分答案



- 可以使用二分答案的题目的答案，会具有某种单调性。
- 以及一些提示性的关键词：“最大值最小”，“最小值最大”等
- 这类题目的思想大概就是，我们知道了答案满足这样的单调性，所以我们在答案的取值范围内进行二分，每次验证一个解。
- 算法模板和之前的二分法求方程近似解非常像，不过会套一个奇奇怪怪的函数检查生成的答案的正确性。

二分答案



- 通常来说，二分答案题目为下列两种中的一种：
  1. 给定一个评价函数，求评价函数的最小值 / 最大值 。
  2. 给定一个条件，要求在满足条件的同时，使得代价最小。
- 计算函数或者判断符合条件需要消耗极长的时间；
- 或者评价函数的值域太大了，不能一一计算。