



二分查找算法

2019.5.25

◆引入:

- 1. **yt1242**: 网线主管 (**poj1064**)
- 2. **yt1433**: **【例题1】** 愤怒的牛 (**poj2456, sp297**)

◆ 中国人民银行发行2019年版第五套人民币50元、20元、10元、1元纸币及1元、5角、1角硬币



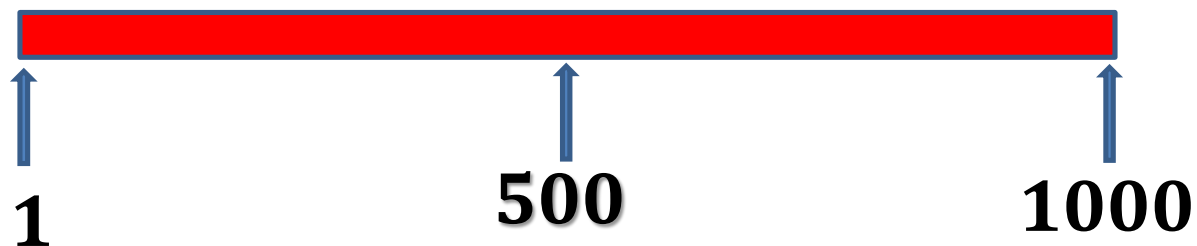
1.找假币:

12个硬币中1个是假的，已知假的比真的**轻**。有一个天平。
问：最少称几次能找出假币？



2.猜数字:

可能的答案1-1000



情况1: 正确



情况2: 小了



情况3: 大了



只要每次询问区间的**正中间**，无论反馈哪种结果，都能使存在答案的区间长度减半。

3.你能举起杠铃的最大重量是多少？

- 能举起的范围 $[0,500]$ 公斤
- 让你举试



龙清泉，2008年北京第29届奥运会56公斤级，抓举132公斤，挺举160公斤，总成绩292公斤夺得冠军

◆ 二分的本质：

- 每一次的查找或询问的结果：
- 找到
- 如果找不到，答案所在的范围能缩小为原来的一半；
- 在答案所在的范围内继续用同样的方法查找或询问
-

◆ 查找算法

- 在数据集合中寻找给定关键字的位置或判断其有无。
- 集合数列{10,20,15,21,31}中
- 15是第几个数?
- 100是第几个数?
- 查找最大值, 最小值等
- 查找算法: 线性查找; 二分查找

◆顺序查找

- 用for循环实现:
- 在n个元素的序列中 $a[0], a[1] \dots a[n-1]$ 查找x:
- 找到输出下标i, 找不到输出-1.
- 输入:
 - 5
 - 10,20,15,21,31
 - 15
- 输出: 2

◆输出x在序列中第一次出现的位置（0开始）,没有输出-1

```
cin>>n;  
for(int i=0;i<n;i++) cin>>a[i];  
int x;  
cin>>x;  
int k=-1;  
for(int i=0;i<n;i++)  
    if(a[i]==x){  
        k=i;  
        break;  
    }  
cout<<k<<endl;
```

自定义线性查找函数：

```
int Find(int x) { //a[0]...a[n-1]
    for(int i=0; i<n; i++)
        if(a[i]==x) return i;
    return -1;
}
```

每次循环：两次判断，一次加法

线性查找的**常数优化**：效率提高常数倍（循环次数不变，循环体运算次数减少或者加快运算效率）



“哨兵”：设置一个特殊元素，借助编程技巧，达到简化控制循环的目的。

查找的元素放在数组末尾，当做“哨兵”，用作查找控制终点。

n 个元素： $a[0], a[1], \dots, a[n-1]$

初始： $i=0$ 开始向后找：找到 ($i < n$); 找不到 ($i == n$)

```
int Find(int x) { //a[0]...a[n-1]
    a[n]=x; //哨兵
    int i=0;
    while(a[i]!=x) i++;
    return i!=n;
}
```

每次循环：一次判断，一次加法。少用一次判断i是否越界，因为一定能找到x。

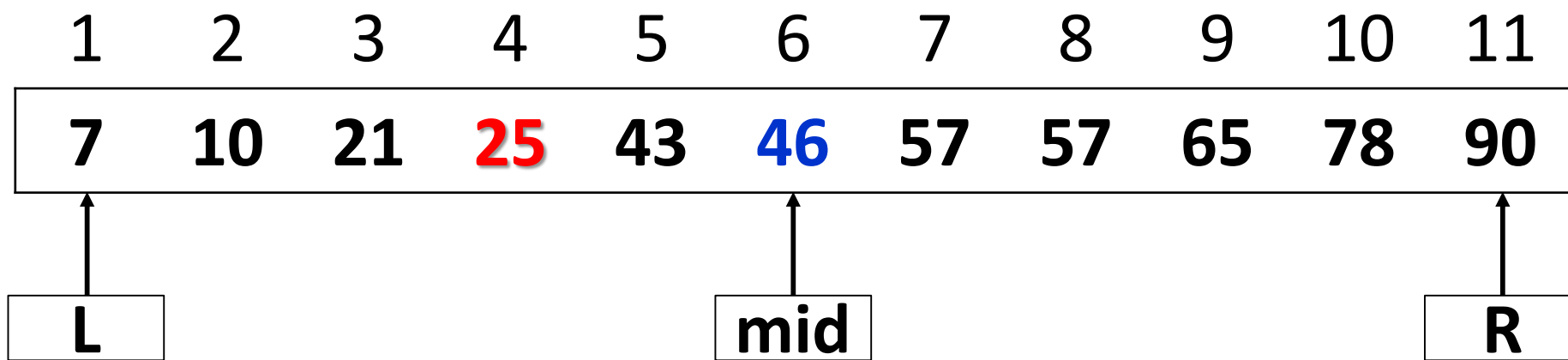
- ◆ 长度为 n 的线性表上线性查找的时间复杂度: $O(n)$
- ◆ 如果 q 次查询则时间复杂度: $O(q*n)$
- ◆ $q \leq 10000, n \leq 100000$?

有序（递增或递减）表上的查询：除了线性查找，还可以用二分！

查找25

1	2	3	4	5	6	7	8	9	10	11
7	10	21	25	43	46	57	57	65	78	90

查找25



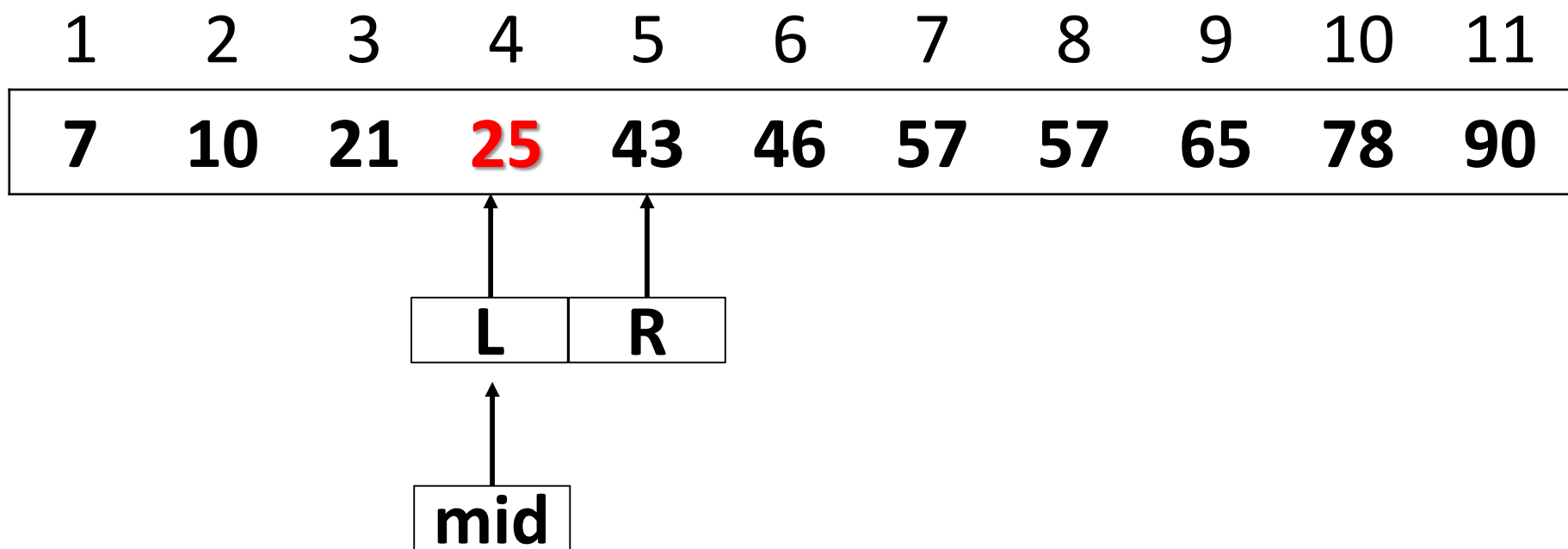
查找25

1	2	3	4	5	6	7	8	9	10	11
7	10	21	25	43	46	57	57	65	78	90
L		mid		R						

查找25

1	2	3	4	5	6	7	8	9	10	11
7	10	21	25	43	46	57	57	65	78	90
L		mid		R						

查找25



◆ 上述进行的前提：

- 查找的序列是**有序**的
- 如果无序可以**先排序**

查找60

1	2	3	4	5	6	7	8	9	10	11
7	10	21	25	43	46	57	57	65	78	90

◆实现:

//a[1..n] 中查找是否有x, 找到返回下标, 找不到返回-1

```
int Find(int x) {  
    int l=1,r=n,mid;  
    while (l<=r) {  
        mid=(l+r)/2;  
        if (a[mid]==x) return mid;  
        else  
            if (a[mid]>x) r=mid-1; // [l, mid-1]  
            else l=mid+1; // [mid+1, r]  
    }  
    return -1; // No Found  
}
```

◆思考下列问题:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	33	44	44	44	46	55	55	55	66	77	88	88	99

- 1. 大于等于x的最小值(如果多个和x相等, 返回第一个)是哪个? 大于等于x的有多少个?
- 2. 小于等于x中最大值(如果多个和x相等, 返回最后一个)是哪个? 小于等于x的有多少个?
- 3. 如果多个值和x相等, 有多少个?
- 4. 如果找不到x, 与x最接近的是几?

1. 大于等于x的最小值(如果多个和x相等, 返回第一个)是哪个?
大于等于x的有多少个?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	33	44	44	44	46	55	55	55	66	77	88	88	99
<X		>=X											

x=44, L=3 (找到)

x=40, L=3 (没找到)

➤ $L = \min(i | a[i] \geq x)$

➤ 满足 $a[i] \geq x$ 条件的最小的i。

查找方法1:

int l, r, mid;

l=1; r=n;

while (l < r) {

mid = (l + r) / 2;

if (a[mid] >= x) r = mid;

else l = mid + 1;

}

return l;

2. 小于等于x中最大值(如果多个和x相等, 返回最后一个)是哪个?
 小于等于x的有多少个?

1	2	3	4	5	6	7	8	9	10	11	12	13	14
22	33	44	44	44	46	55	55	55	66	77	88	88	99

$\leq x$

$> x$

$x=55, L=9$

$x=60, L=9$

- $L = \max\{i \mid a[i] \leq x\}$
- 满足 $a[i] \leq x$ 条件的最大的 i 。

查找方法2:

$l=1; r=n;$

while($l < r$) {

$mid = (l + r + 1) / 2;$

 if ($a[mid] \leq x$) $l = mid;$

 else $r = mid - 1;$

}

return $l;$

//if $a[l] == x$ 找到

◆总结:

➤ 答案保证在区间 $[l, r]$ 以内；循环 $l=r$ 时结束；

查找 $\geq x$ 的数中最小的一个（ x ，或 x 后继）：用于就最小值

```
while (l < r) {
    mid = (l + r) / 2;
    if (a[mid] >= x) r = mid; // 答案区间[l, mid]
    else l = mid + 1;         // 答案区间[mid + 1, r]
}
return l;
```

查找 $\leq x$ 的数中最大的一个（ x ，或 x 前驱）：用于求最大值

```
int l = 1, r = n, mid;
while (l < r) {
    mid = (l + r + 1) / 2;
    if (a[mid] <= x) l = mid; // 答案区间[mid, r]
    else r = mid - 1;         // 答案区间[l, mid - 1]
}
return l;
```

查找 x 也可以用上述两种方法，最后检查 $a[l]$ 是否等于 x 即可。

- 3.如果多个值和 x 相等，有多少个？
- 4.如果找不到 x ，与 x 最接近的是几？

◆二分查找作用：不仅仅是查找

- 1. 查找元素是否存在（位置或值）
- 2. 求满足条件的**最值**（最大值/最小值）

二分查找算法：

- 在一个**单调有序**(递增或递减)的区间 $[a_1, a_n]$ 中查找元素 x , 每次将区间分为左右长度相等的两部分, 判断解在哪个区间中并调整区间上下界, 重复直至找到 x 。
- **找不到有时候更有用。**
- 一次查找的时间复杂度: $O(\log n)$ 优于顺序查找 $O(n)$
- q 次查找的时间复杂度: $O(q * \log n)$
- $q \leq 10000, n \leq 100000$ 问题完美解决

- 实数区间上的二分查找
- 整数区间上的二分查找及其应用

整数区间上的二分查找:

1.有序序列上的二分查找(分的下标标号)

2.查找最终有两种结果:

1.找到了。（如果有多个，找的是哪一个？）

2.没有找到。（找不到，最终停在那里？）

*3.求最优值问题（最大值最小，最小值最大）重点:

二分答案转换为判定问题

➤ 训练题目：

- 1. yt1244： 和为给定数
- 2. 查找最接近的元素（oj 1.11.01）

整数二分答案（最值问题转化为判定性问题）

- ◆ 有些问题，如求最大值最小值问题：尤其是：
- ◆ 最大值最小；
- ◆ 最小值最大；
- ◆ 直接求不容易，可以确定答案范围，然后用二分答案的方法，转为判断性问题。

◆你能举起杠铃的最大重量是多少？

➤ [0,500]公斤

➤ 令：

– 条件 $\text{check}(x)$ =能举起重量为 x 的杠铃。

➤ 问题转化：

– 求满足条件 $\text{check}(x)$ 成立的最大 x 。

```
➤ L=0; r=500;  
➤ While (l<r) {  
➤     mid=(l+r+1) div 2;  
➤     if check(mid) then l=mid;  
➤     else r=mid-1;  
➤ } //l=r停止查找  
➤ return l;
```

主要任务: **check(x)**=能举起重量为**x**的杠铃。
1:能; 0:不能

◆最大值最小：

你要想考上复旦，6门科最高的最少要考多少分？复旦不低于680分

(如果您最高的考140分，最多考840（其实最多720）：再少点还行；

如果您最高的才考90分，6门最多考540分，山师也考不上：太少了，要多点)

➤ 六门课：语数外各150，另外的3门各100，总分750.

➤ `l=0, r=150;`

➤ `While (l<r) {`

➤ `mid=(l+r) div 2;`

➤ `if check(mid) then r=mid; //[1,mid]`

➤ `else l=mid+1; //[mid+1, r]`

➤ `}` //l=r停止二分, l就是答案

➤ `//return l;`

➤ 验证最低分mid: `check(mid):` mid分能考上返回1，考不上返回0.

➤ `int check(int x){return 6*x>=680;}`

```
4 int check(int x) { //能考上返回1, 考不上返回0
5     return 6*x>=680;
6 }
7 int main() {
8     int l=0, r=150, mid;
9     while (l<r) {
10         mid=(l+r)/2;
11         if (check(mid)) r=mid; //答案区间[l,mid]
12         else l=mid+1;         //答案区间[mid+1,r]
13     }
14     cout<<l<<endl;
15     return 0;
16 }
```

◆最小值最大:

100个苹果分给7个人，分的最少的那个人最多分多少个？

(如果最少的那个人要是分10个苹果，70个就够分的：还可以多点；

如果最少的那个人要是分20个苹果，那至少要有140个才够分：太多了，要少点)

- `l=0, r=100;`
- `While (l<r) {`
- `mid=(l+r+1) div 2;`
- `if check(mid) then l=mid; //[mid, r]`
- `else r=mid-1; //[1, mid-1]`
- `}` `//l=r`停止二分, `l`就是答案
- `//return l;`
- 验证最少的分最多`mid`个: `check(mid)`: `mid`个够分的返回1，不够分的返回0.
- `int check(int x){return 7*x<=100;}`

```
4 int check(int x){ //够分的返回1, 不够分的返回0
5     return 7*x<=100;
6 }
7 int main(){
8     int l=0,r=100,mid;
9     while(l<r){
10         mid=(l+r+1)/2;
11         if(check(mid)) l=mid; //答案区间 [mid,r]
12         else r=mid-1;        //答案区间 [l,mid-1]
13     }
14     cout<<l<<endl;
15     return 0;
16 }
```

◆求最值的两种模板（二分答案）：

- 1.求满足条件的最小值（想想你能考上复旦吧）：
 - 答案所在的范围 $[l, r]$
 - While ($l < r$) {
 - $mid = (l + r) \div 2;$
 - if **check**(**mid**) then $r = mid;$ $// [l, mid]$
 - else $l = mid + 1;$ $// [mid + 1, r]$
 - } $// l = r$ 停止二分, l 就是答案
 - $// return\ l;$
 - **check**(**mid**): **mid**满足条件返回1, 不满足返回0.

- 2.求满足条件的**最大值**(想想你能举起的最大重量):
 - 答案所在的范围[1, r]
 - While (l<r) {
 - mid=(l+r+1) div 2;
 - if **check(mid)** then l=mid; // [mid, r]
 - else r=mid-1; // [1, mid-1]
 - } // l=r停止二分, l就是答案
 - //return l;
 - check(mid): mid满足条件返回1, 不满足返回0.

二分最容易出现问题:

1.确定答案所在区间 $[l, r]$

2.循环条件: $\text{while}(l < r)$

3.注意两种配套写法: 最容易死循环。二分的关键

最小值: $r = \text{mid}, l = \text{mid} + 1, \text{mid} = (l + r) / 2$

最大值: $l = \text{mid}, r = \text{mid} - 1, \text{mid} = (l + r + 1) / 2$

4. $l == r$ 时结束, 注意找不到的情况, 最后判断 $a[l]$ 是否等于 x

5. $\text{check}(\text{mid})$ 一般用贪心实现。

6.如果查找的范围是 $[1, n]$:

尤其注意最后查找结果 $l = 1$ 或者 $l = n$ 的情况

◆训练::

- 1.Cable master(poj1064,yt1242): 求最大值。

整数二分或实数二分

实数二分

也可转化为整数二分

精度问题,不能四舍五入

%.2f %.2lf(pojudge上这里用2f? ? ?)

yt1242第4个数据需考虑精度,读入时调整+0.5

◆二分答案

```
int l=0,r=100000000,mid;
while(l<r){
    mid=(l+r+1)/2;
    if(check(mid)==1) l=mid;
    else r=mid-1;
}
printf("%.2f\n",l/100.0);
```

- 2. Aggressive cows 愤怒的牛(poj2456,sp297,yt1433)
- 最小值最大：最大值

➤ 3.Copying Books poj1505 p1281（最大值最小化：最小值）

◆noip考试题:

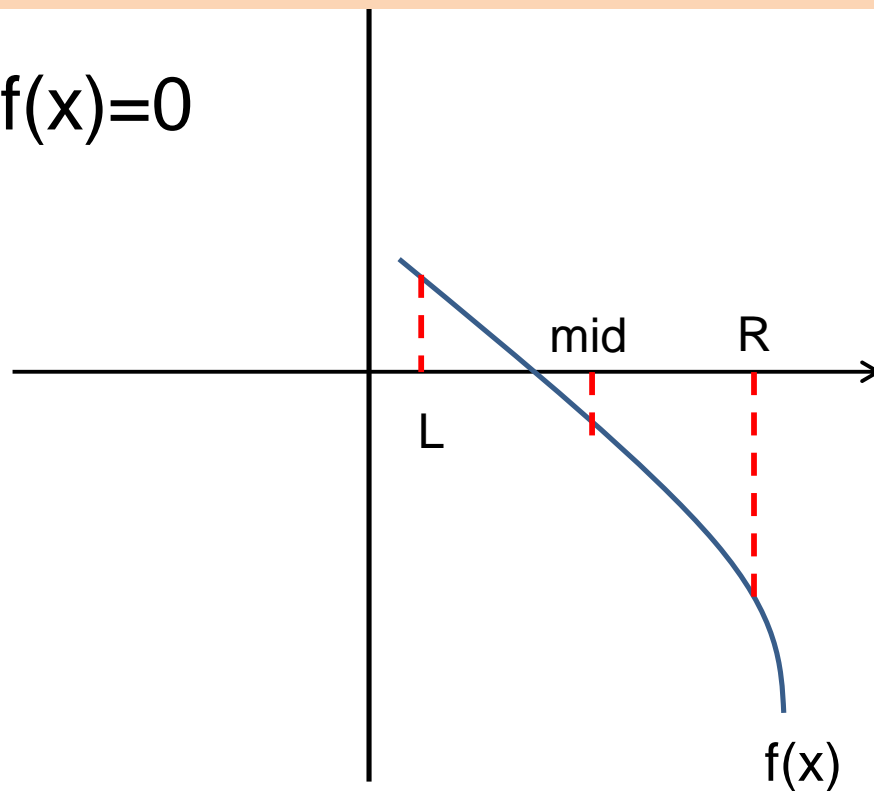
- 1.NOIP2015跳石头P2678(最小值最大)
- 2.NOIP2011聪明的检查员(找离某个值最近的点)
- 3.NOIp2010关押罪犯（最大值最小化）

实数单调区间上的二分查找:

函数 $f(x)$ 图像在区间 $[l,r]$ 与 x 轴的交点, $f(x)=0$

前提: $f(x)$ 在该区间 $[l,r]$ 内单调

- $L=1.5$;
- $R=2.4$;
- $f(L)*f(R) \leq 0$
- 解的区间: $[1.5, 2.4]$
- 当区间 $[l, r]$ 足够小, 满足精度要求即可停止。



1.二分法求函数的零点 (一本通1241) (noi1.11.02)

有函数:

$$f(x) = x^5 - 15 * x^4 + 85 * x^3 - 225 * x^2 + 274 * x - 121$$

已知 $f(1.5) > 0$, $f(2.4) < 0$

且方程 $f(x) = 0$ 在区间 $[1.5, 2.4]$ 有且只有一个根.

求出这个根。

要求四舍五入到小数点后6位

$L=1.5;$

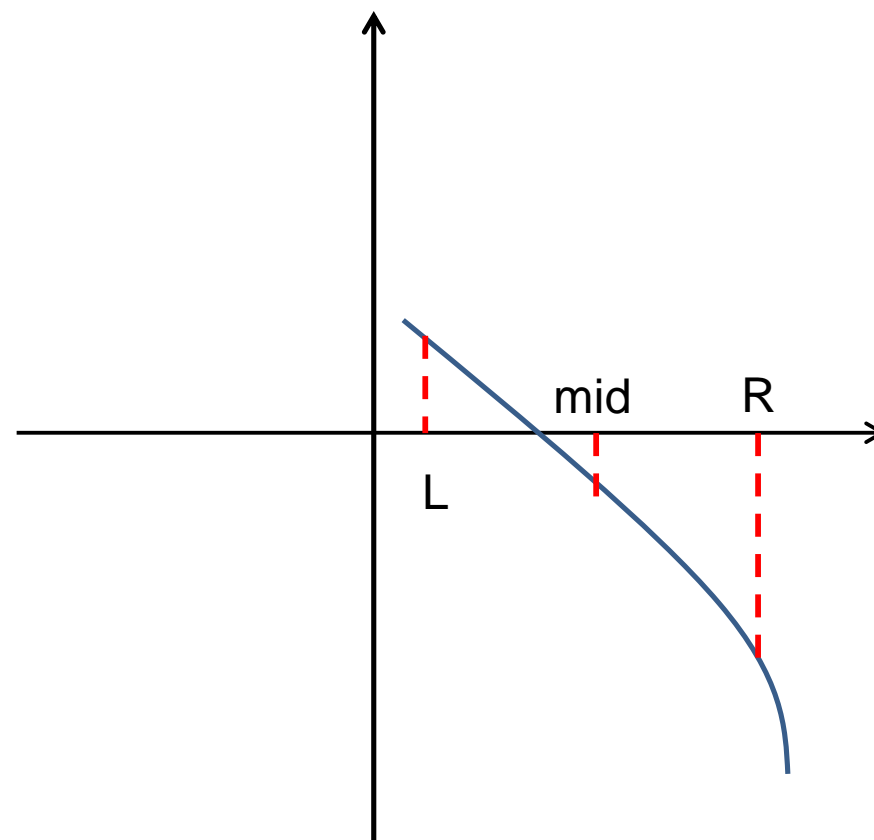
➤ $R=2.4;$

➤ $f(L) * f(R) \leq 0$

➤ 解的区间: $[1.5, 2.4]$

➤

➤



$L=1.5;$

➤ $R=2.4;$

➤ $f(L)*f(R) \leq 0$

➤ 解的区间: $[1.5, 2.4]$



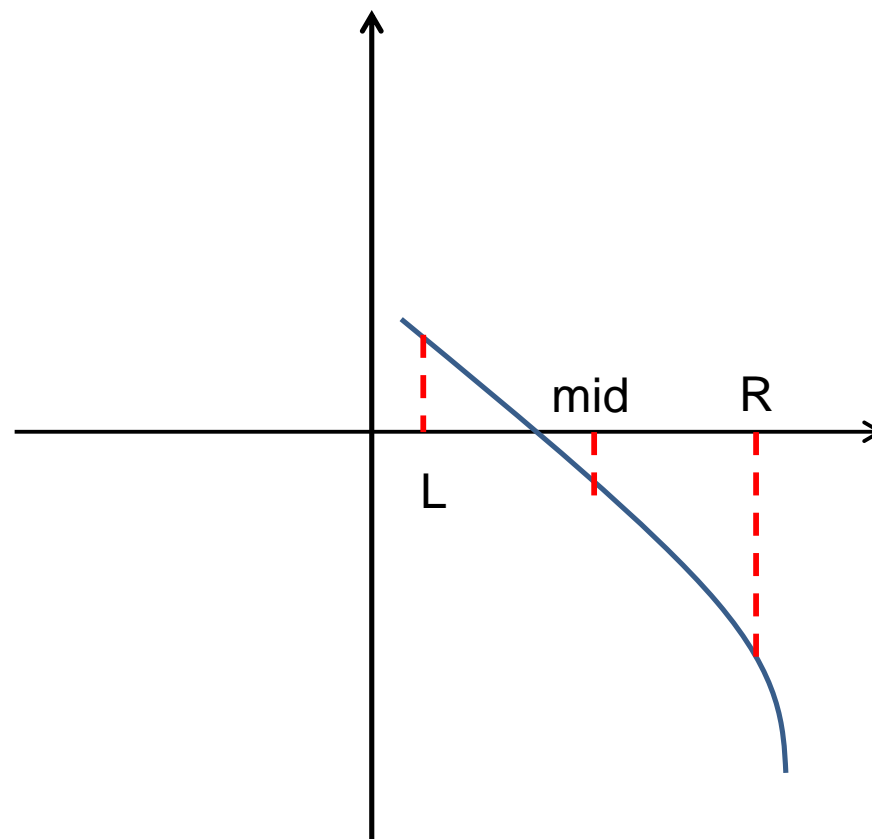
➤ $mid = (L+R)/2;$

➤ If $f(mid)*f(R) \leq 0$

➤ 答案在 $[mid, R]$

➤ Else

➤ 答案在 $[L, mid]$



```
double mid, l=1.5, r=2.4;
double eps=1e-8; //至少比精度要求多1位, 2位最好
while (l+eps<r) {
    mid=(l+r)/2;
    if (f(l)*f(mid)<=0) r=mid;
    else l=mid;
}
printf("%.6lf\n", mid);
```

一般要求保留 k 位有效小数时, 所需精度 $\text{eps}=1\text{e}-(k+2)$ 即可
就是区间范围距离 (越小精度越高)

如果精度不好确定时，干脆直接使用固定循环次数的二分，精度一般比设置了eps的更高，更好理解。

```
double mid, l=1.5, r=2.4;
for (int i=0; i<100; i++) {
    mid=(l+r)/2;
    if (f(l)*f(mid)<=0) r=mid;
    else l=mid;
}
printf("%.6lf\n", mid);
```

◆ 2. P1024 一元三次方程求解 (NOIP2001)

- 形如： $ax^3+bx^2+cx+d=0$ 这样的的一元三次方程。
- 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。
- 【输入样例】
- 1.0 -5.0 -4.0 20.0
- 【输出样例】
- -2.00 2.00 5.00

◆注意事项:

- 整点不要计算重复

◆ 3. Cable master(poj1064,yt1242)

有 n 条电缆，他们的长度分别为 $l[i]$ 。如果从 n 条电缆中切割出 K 条长度相同的电缆的话，这 k 条电缆每条最长能多长？答案小数点后保留两位有效数字。

输入：

n, k

n 行： $l[i]$

Sample Input

4 11

8.02

7.43

4.57

5.39

Sample Output

2.00

◆二分答案

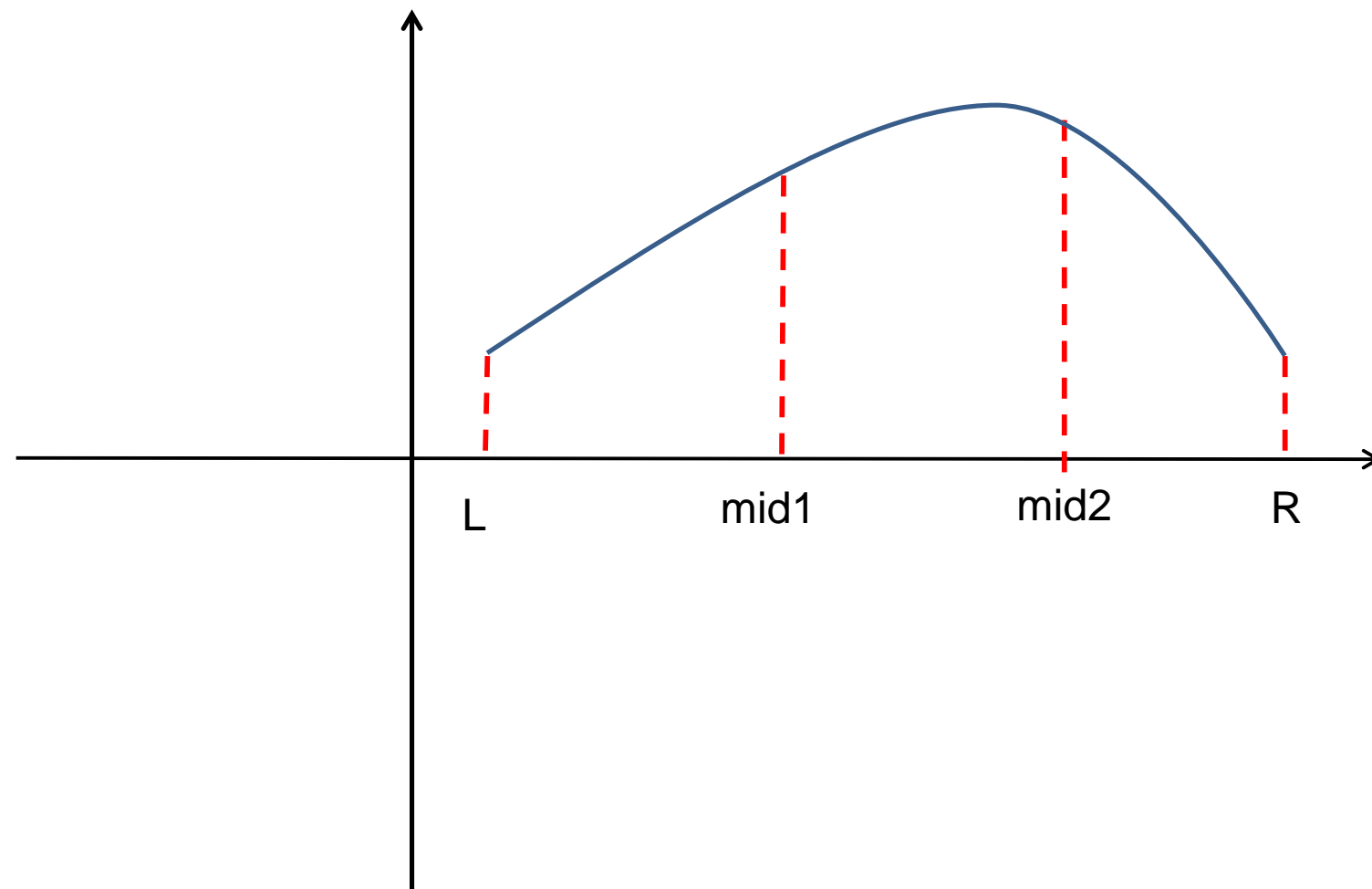
```
double l=0,r=100000,mid;  
for(int i=0;i<100;i++){  
    mid=(l+r)/2;  
    if(check(mid)==1) l=mid;  
    else r=mid;  
}  
printf("%.2f\n",floor(100*l)/100);
```

◆ 4. Best Cow Fences poj2018, yt1343

➤ 没做?

凸（凹）区间上的最大（小）值：三分法

凸区间上的最大值
凹区间上的最小值
只有一个凸或凹



◆训练：

- 1.ZOJ 3203 Light Bulb
- 2.hdu 2438 Turn the corner