Double Divide!

我们仍未知道那天所看见的题要怎么二分

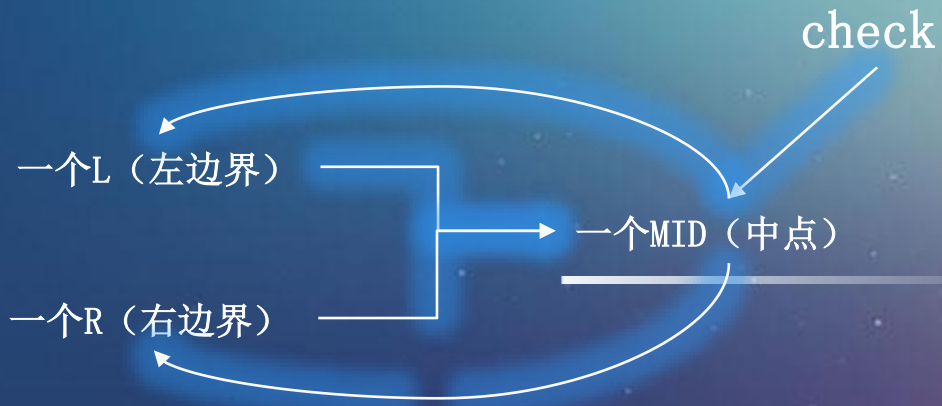
~~这次没有ACM的LOGO了，就算队长要赶我走我也不放了~~

2017

01.基础二分

什么是二分

需要：



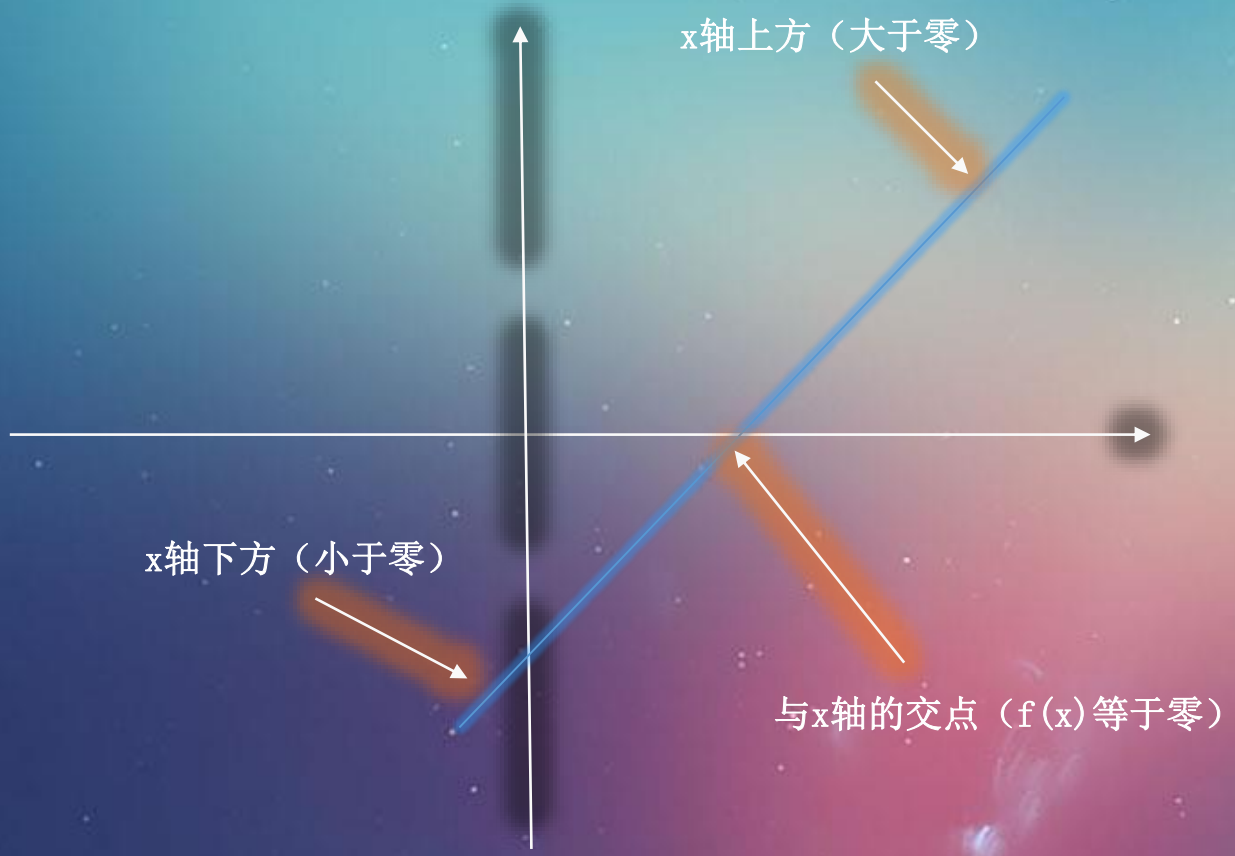
适用条件

- 单调性的序列

- 可判定

举个栗子

O(n) -> O(log n)



你知道整数二分有多少种写法嘛

- 取整方式：向下取整，向上取整
- 区间开闭：闭区间 左闭右开区间 左开右闭区间 开区间
- 问题类型：
  对于不下降序列a，求最小的i，使得a[i] = key
  对于不下降序列a，求最大的i，使得a[i] = key
  对于不下降序列a，求最小的i，使得a[i] > key
  对于不下降序列a，求最大的i，使得a[i] < key
  对于不上升序列a，求最小的i，使得a[i] = key
  对于不上升序列a，求最大的i，使得a[i] = key
  对于不上升序列a，求最小的i，使得a[i] < key
  对于不上升序列a，求最大的i，使得a[i] > key

这个时候就上链接：https://www.zhihu.com/question/36132386

几个重点

不要以 L == R 做终结条件。有时会被跳过的。

不相信在 [1, 5] 里找 0 试试？

正确的终结条件是： L > R 即搜索空间为空

(L + R) / 2的过程中L + R 爆范围了怎么办？

L + (R - L) / 2

02.二分答案

先来一个栗子

一个数字num，使其1~num中有n个2的倍数，m个三的倍数，且这n+m个数字均不相同，输出满足条件的num的最小值。

n = 1, m = 3
ans = 9　　(2, 3, 6, 9)

n = 3, m = 2
ans = 8　　(2, 4, 8, 3, 6)

怎么做？

n = 1, m = 3
ans = 9 (2, 3, 6, 9)

n = 3, m = 2
ans = 8 (2, 4, 8, 3, 6)

贪心？

优先先把2确定？

样例2过不了QAQ

n = 4, m = 2
ans = 10 (2, 4, 8, 10, 3, 6) ?
ans = 9 (2, 4, 6, 8, 3, 9) !

优先先把3确定？

Wrong Answer

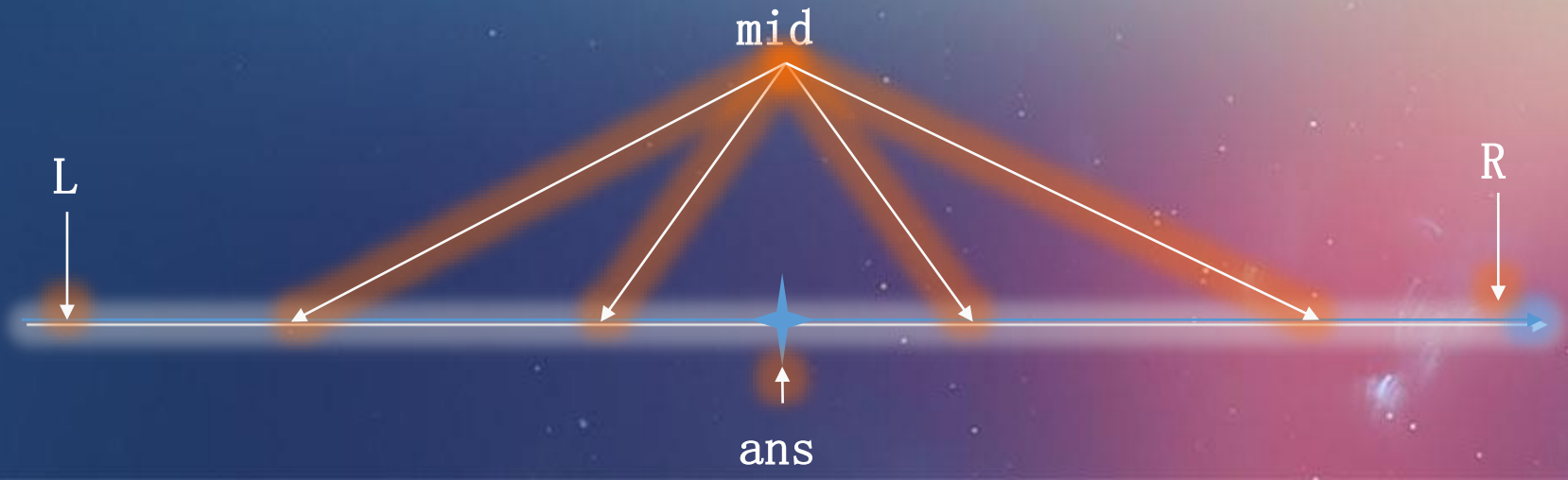
能过样例，好像可行？

其实是能贪心过的，但是我们重点在二分。

一直判断下去！

单调性？　当mid >= ans的时候，能满足题意

当mid < ans的时候，不能满足题意



大力出奇迹

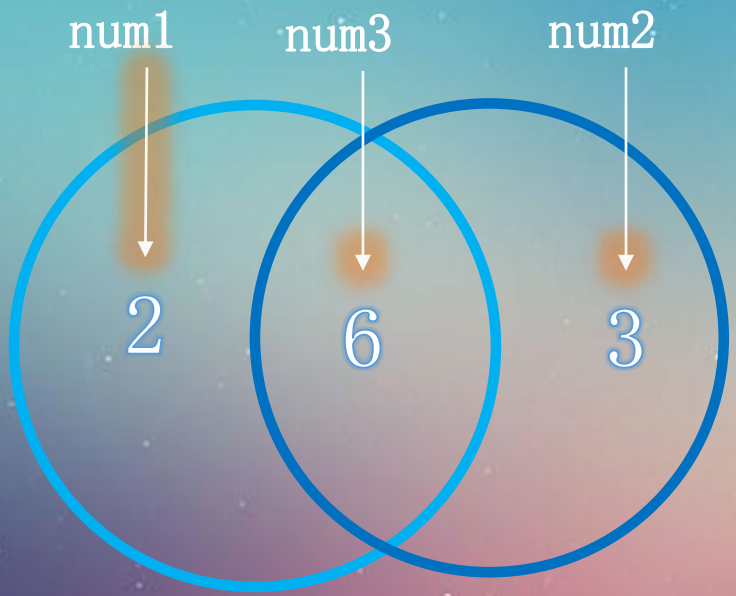
怎么check？

怎么表示两个圈的并？

怎么表示满足条件？

num1 + num2 - num3 >= n + m

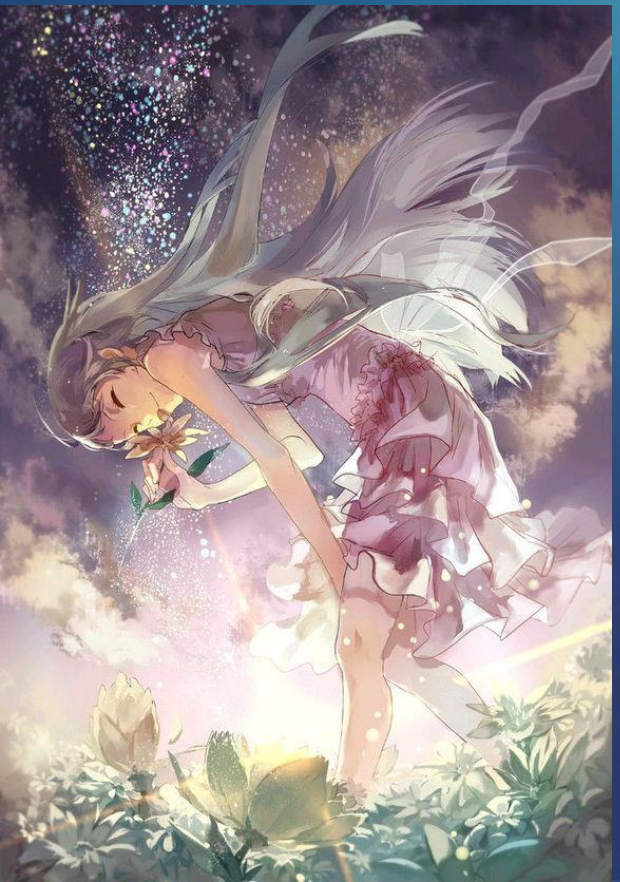
```
bool check(int mid)
{
    int a = mid / 2;
    int b = mid / 3;
    int c = mid / 6;
    if(n > a || m > b) return false;
    if(a + b - c >= n + m) return true;
    else return false;
}
```



把无从下手的操作转化成check操作

03.浮点数二分

还是一个栗子



N条绳子长度分别为len_i，从中切割出K条长度相同的绳子，K条绳子每条最长有多长？

数据与浮点数相关

考虑二分

单调性 ： 当mid <= ans，能分割的条数 >= k

当mid > ans，能分割的条数 < k

每个mid可check ：

```
bool check(double mid)
{
    int num = 0;

    for(int i = 0; i < n; i ++){
        num += (int)(len[i] / mid);
    }

    if(num >= x){
        return true;
    }
    else{
        return false;
    }
}
```

二分怎么写？

```
while(r - l >= eps){     ///eps根据题目要求精度而决定
    int mid = (l + r) / 2;

    if(check(mid)){
        r = mid - eps;
    }
    else{
        l = mid + eps;
    }
}
```

两个问题：

- 题目要求精度为1e-5时，我们的eps需要设置到1e-8甚至更多
- eps会因为太小加上浮点数精度问题陷入死循环

怎么办？

通过循环强行控制二分次数

100次循环能达到1e-30的精度

```
for(int i = 0; i < 100; i ++){
    double mid = (l + r) >> 1;
    if(check(mid)){
        l = mid;
    }
    else{
        r = mid;
    }
}
```

04.STL与二分

新的问题来了

查找有序序列中数字的个数怎么办？

例如从1，2，2，2，5中寻找2

lower_lound和upper_bound

前提是有序！

lower_lound

函数lower_bound(first, last, val)在first和last中的前闭后开区间进行二分查找，返回大于或等于val的第一个元素位置。如果所有元素都小于val，则返回last的位置

今天栗子有点多　　一个序列：4, 10, 11, 30, 69, 70, 96, 100

pos = lower_bound(number, number + 8, 3) - number;

pos = 0，即number数组的下标为0的位置。

pos = lower_bound(number, number + 8, 9) - number;

pos = 1，即number数组的下标为1的位置（即10所在的位置）。

pos = lower_bound(number, number + 8, 111) - number;

pos = 8，即number数组的下标为8的位置
（但下标上限为7，所以返回最后一个元素的下一个元素）。

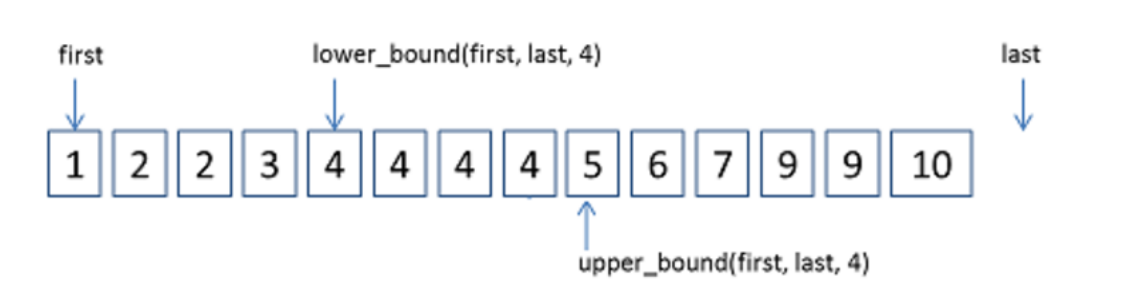
STL源码

```
//这个算法中，first是最终要返回的位置
int lower_bound(int *array, int size, int key)
{
    int first = 0, middle;
    int half, len;
    len = size;

    while(len > 0) {
        half = len >> 1;
        middle = first + half;
        if(array[middle] < key) {
            first = middle + 1;
            len = len-half-1;        //在右边子序列中查找
        }
        else
            len = half;              //在左边子序列（包含middle）中查找
    }
    return first;
}
```

upper_lound

```
int upper_bound(int *array, int size, int key)
{
    int first = 0, len = size-1;
    int half, middle;

    while(len > 0){
        half = len >> 1;
        middle = first + half;
        if(array[middle] > key)      //中位数大于key,在包含last的左半边序列中查找。
            len = half;
        else{
            first = middle + 1;     //中位数小于等于key,在右半边序列中查找。
            len = len - half - 1;
        }
    }
    return first;
}
```

一张图告诉你我之前都是废话



回到刚开始的问题

有序列中某一元素的个数

```
cnt = upper_bound(ma, ma + n, k) - lower_bound(ma, ma + n, k);
```

谢谢