2-28：

> *设 $X[0:n-1]$ 和 $Y[0:n-1]$ 为两个数组，每个数组中含有 n 个已排好序的数。设计一个 $O(logn)$ 时间的算法，找出 X 和 Y 的 $2n$ 个数的中位数。*

题目分析：

为简化本题，假设本题中的“已排好序”均为升序排序，即从小到大排序。

看到 $O(logn)$，第一想法就是二分。本题要寻找中位数，由于数组总共有 $2n$ 个数，故中位数必然是 $\frac{x+y}{2}$ 的形式。

那么如何寻找这个中位数呢？显而易见，我们要找的是这 $2n$ 个数中第 n 大和第 n 小的数。我们可以用二分的方法去依次排除大于第 n 个数的数和小于等于第 n 个数的数。同时维护两个变量 a,b，在删除大于等于第 n 个数的数时，取 $a=min(a,\text{被删除的元素中最小的})$；删除小于第 n 个数的数时，$b=max(b,\text{被删除的元素中最大的})$。通过不断地删除，最后将数组删空，剩余的 a,b 的平均值即为所求中位数。

下记 $mid=\frac{n_i}{2}$（$i=X,Y$，n_i 表示数组 i 中剩余的元素数），$Xleft$ 和 $Yleft$ 表示数组 X,Y 现在从下标何处开始，取 X，Y 剩余元素的中位数 $x=X[Xleft+mid],y=Y[Yleft+mid]$。

①：若 n 为奇数，每次选取 x,y 后，均由两种可能：

- 若$x\geq y$，删除数组 X 的右半部分，更新 a；删除数组 Y 的左半部分，更新 b。
- 若$x<y$，删除数组 X 的左半部分，更新 b；删除数组 Y 的有半部分，更新 a。

②：若 n 为偶数，此时相对比较复杂，为了保证“对称性”，要考虑以下情况：

> 先取两个数组第 $\frac{n}{2}$ 个元素进行比较，将小的那一部分的左半部分删除，更新 b。
>
> 再取两个数组第 $\frac{n}{2}+1$ 个元素比较，将大的那一部分的右半部分删除，更新 a。

示例：

```
1 2 5 6
3 4 7 8
```

对于上述示例，首先比较2和4，显然2小，因此将1，2删除。再比较2和4右边的5和7，显然7大，于是将7，8删除，变为如下模样：

~~1 2~~ 5 6

3 4 ~~7 8~~

这种删除方式，要么上下删除相同个数的元素；要么就将一个数组全部删空。若将数组删空，则需考虑情况③。

③：一个数组已被删空，剩下数组 S。数组 S 中元素的个数必为偶数（下证），因此在数组 S 中进行二分，删除左右部分并更新 a,b 即可。

（证明：将数组删空的情况只有两种情况：

①：当 X 和 Y 中元素的个数为偶数的情况。而此种情况剩下的数组元素必为偶数。

②：X 和 Y 都只剩一个元素，两个数组全部删空，则 S 中数组仍为偶数。）

用此方法，每次取两个数组中剩余元素的中位数，实现二分，即可在 $O(logn)$ 的复杂度内找出中位数。

代码实现：

见2-28 Code.cpp

输出示例：

```
输入样例1：
2
1 4
2 3
输出1：2.5
输入样例2：
4
4 5 6 7
1 2 8 9
输出2：5.5
输入样例3：
8
2 3 4 8 9 10 11 12
1 2 3 4 5 6 7 8
输出3：5.5
对此输入排序，为1 2 2 3 3 4 4 5 6 7 8 8 9 10 11 12
```

算法分析：

本算法中，通过不断删除左右部分和更新 a, b 来进行中位数选取。

- 偶数情况：删除的左半部分有 $\frac{n}{2}$ 个元素，右半部分也有 $\frac{n}{2}$个元素，则将规模从 $2n$ 变为 n，规模减半。
- 奇数情况：删除的左右半部分各有 $\lfloor \frac{n}{2} \rfloor + 1$，因此删除的元素数也为 n，规模减半。

因此，此问题每次规模减半，而合并问题及各项其余操作（不包含输入）的复杂度均为 $O(1)$ ，因此算法总复杂度为：

$$T(n) = T(\frac{n}{2}) + O(1) = O(logn)$$