

```
    }  
    return left;  
}
```

数组中子数组的最大累乘积

【题目】

给定一个 `double` 类型的数组 `arr`，其中的元素可正、可负、可 0，返回子数组累乘的最大乘积。例如，`arr=[-2.5, 4, 0, 3, 0.5, 8, -1]`，子数组`[3, 0.5, 8]`累乘可以获得最大的乘积 12，所以返回 12。

【难度】

尉 ★★★☆☆

【解答】

本题可以做到时间复杂度为 $O(N)$ 、额外空间复杂度为 $O(1)$ 。所有的子数组都会以某一个位置结束，所以，如果求出以每一个位置结尾的子数组最大的累乘积，在这么多最大累乘积中最大的那个就是最终的结果。也就是说，结果= $\text{Max}\{\text{以 arr}[0]\text{结尾的所有子数组的最大累乘积, 以 arr}[1]\text{结尾的所有子数组的最大累乘积}\cdots\cdots\text{以 arr}[\text{arr.length}-1]\text{结尾的所有子数组的最大累乘积}\}$ 。

如何快速求出所有以 i 位置结尾(`arr[i]`)的子数组的最大乘积呢？假设以 `arr[i-1]` 结尾的最小累乘积为 `min`，以 `arr[i-1]` 结尾的最大累乘积为 `max`。那么，以 `arr[i]` 结尾的最大累乘积只有以下三种可能：

- 可能是 `max*arr[i]`。`max` 既然表示以 `arr[i-1]` 结尾的最大累乘积，那么当然有可能以 `arr[i]` 结尾的最大累乘积是 `max*arr[i]`。例如，`[3,4,5]`在算到 5 的时候。
- 可能是 `min*arr[i]`。`min` 既然表示以 `arr[i-1]` 结尾的最小累乘积，当然有可能 `min` 是负数，而如果 `arr[i]`也是负数，两个负数相乘的结果也可能很大。例如，`[-2,3,-4]`在算到-4 的时候。
- 可能仅是 `arr[i]`的值。以 `arr[i]`结尾的最大累乘积并不一定非要包含 `arr[i]`之前的数。例如，`[0.1,0.1,100]`在算到 100 的时候。

这三种可能的值中最大的那个就作为以 i 位置结尾的最大累乘积，最小的作为最小累

乘积，然后继续计算以 $i+1$ 位置结尾的时候，如此重复，直到计算结束。

具体过程请参看如下代码中的 `maxProduct` 方法。

```
public double maxProduct(double[] arr) {
    if (arr == null || arr.length == 0) {
        return 0;
    }
    double max = arr[0];
    double min = arr[0];
    double res = arr[0];
    double maxEnd = 0;
    double minEnd = 0;
    for (int i = 1; i < arr.length; ++i) {
        maxEnd = max * arr[i];
        minEnd = min * arr[i];
        max = Math.max(Math.max(maxEnd, minEnd), arr[i]);
        min = Math.min(Math.min(maxEnd, minEnd), arr[i]);
        res = Math.max(res, max);
    }
    return res;
}
```

打印 N 个数组整体最大的 Top K

【题目】

有 N 个长度不一的数组，所有的数组都是有序的，请从大到小打印这 N 个数组整体最大的前 K 个数。

例如，输入含有 N 行元素的二维数组可以代表 N 个一维数组。

219,405,538,845,971

148,558

52,99,348,691

再输入整数 $k=5$ ，则打印：

Top 5: 971,845,691,558,538

【要求】

1. 如果所有数组的元素个数小于 K ，则从大到小打印所有的数。
2. 要求时间复杂度为 $O(K\log N)$ 。