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 $O(K\log N)$。

【难度】

尉 ★★☆☆

【解答】

本题的解法是利用堆结构和堆排序的过程完成的，具体过程如下：

1．构建一个大小为 N 的大根堆 heap，建堆的过程就是把每一个数组中的最后一个值，也就是该数组的最大值，依次加入到堆里，这个过程是建堆时的调整过程(heapInsert)。

2．建好堆之后，此时 heap 堆顶的元素是所有数组的最大值中最大的那个，打印堆顶元素。

3．假设堆顶元素来自 a 数组的 i 位置。那么接下来就把堆顶的前一个数（即 a[i-1]）放在 heap 的头部，也就是用 a[i-1]替换原本的堆顶，然后从堆的头部开始调整堆，使其重新变为大根堆（heapify 过程）。

4．这样每次都可以得到一个堆顶元素 max，在打印完成后都经历步骤 3 的调整过程。整体打印 k 次，就是从大到小全部的 Top K。

5．在重复步骤 3 的过程中，如果 max 来自的那个数组（仍假设是 a 数组）已经没有元素。也就是说，max 已经是 a[0]，再往左没有数了。那么就把 heap 中最后一个元素放在 heap 头部的位置，然后把 heap 的大小减 1(heapSize-1)，最后依然是从堆的头部开始调整堆，使其重新变为大根堆(堆大小减 1 之后的 heapify 过程)。

6．直到打印了 k 个数，过程结束。

为了知道每一次的 max 来自什么数组的什么位置，放在堆里的元素是如下的 HeapNode 类：

```
public class HeapNode {
        public int value; // 值是什么
        public int arrNum; // 来自哪个数组
        public int index; // 来自数组的哪个位置

        public HeapNode(int value, int arrNum, int index) {
                this.value = value;
                this.arrNum = arrNum;
                this.index = index;
        }
}
```

整个打印过程请参看如下代码中的 printTopK 方法。

```
public void printTopK(int[][] matrix, int topK) {
```

```
        int heapSize = matrix.length;
        HeapNode[] heap = new HeapNode[heapSize];
        for (int i = 0; i != heapSize; i++) {
                int index = matrix[i].length - 1;
                heap[i] = new HeapNode(matrix[i][index], i, index);
                heapInsert(heap, i);
        }
        System.out.println("TOP " + topK + " : ");
        for (int i = 0; i != topK; i++) {
                if (heapSize == 0) {
                        break;
                }
                System.out.print(heap[0].value + " ");
                if (heap[0].index != 0) {
                        heap[0].value = matrix[heap[0].arrNum][--heap[0].index];
                } else {
                        swap(heap, 0, --heapSize);
                }
                heapify(heap, 0, heapSize);
        }
}

public void heapInsert(HeapNode[] heap, int index) {
        while (index != 0) {
                int parent = (index - 1) / 2;
                if (heap[parent].value < heap[index].value) {
                        swap(heap, parent, index);
                        index = parent;
                } else {
                        break;
                }
        }
}

public void heapify(HeapNode[] heap, int index, int heapSize) {
        int left = index * 2 + 1;
        int right = index * 2 + 2;
        int largest = index;
        while (left < heapSize) {
            if (heap[left].value > heap[index].value) {
                largest = left;
            }
            if (right < heapSize && heap[right].value > heap[largest].value) {
                largest = right;
            }
            if (largest != index) {
                swap(heap, largest, index);
            } else {
                break;
            }
            index = largest;
            left = index * 2 + 1;
            right = index * 2 + 2;
```

```
        }
    }

    public void swap(HeapNode[] heap, int index1, int index2) {
        HeapNode tmp = heap[index1];
        heap[index1] = heap[index2];
        heap[index2] = tmp;
    }
```

边界都是 1 的最大正方形大小

【题目】

给定一个 $N \times N$ 的矩阵 matrix，在这个矩阵中，只有 0 和 1 两种值，返回边框全是 1 的最大正方形的边长长度。

例如：

0	1	1	1	1
0	1	0	0	1
0	1	0	0	1
0	1	1	1	1
0	1	0	1	1

其中，边框全是 1 的最大正方形的大小为 4×4，所以返回 4。

【难度】

尉 ★★☆☆

【解答】

先介绍一个比较容易理解的解法：

1. 矩阵中一共有 $N \times N$ 个位置。$O(N^2)$

2. 对每一个位置都看是否可以成为边长为 N~1 的正方形左上角。比如，对于(0,0)位置，依次检查是否是边长为 5 的正方形左上角，然后检查边长为 4、3 等。$O(N)$

3. 如何检查一个位置是否可以成为边长为 N 的正方形的左上角呢？遍历这个边长为 N 的正方形边界看是否只由 1 构成，也就是走过 4 个边的长度(4N)。$O(N)$

所以普通方法总的时间复杂度为 $O(N^2) \times O(N) \times O(N) = O(N^4)$。

本书提供的方法的时间复杂度为 $O(N^3)$，基本过程也是如上三个步骤。但是对于步骤 3，