```
                m[tR][tC + i] = m[dR - i][tC];
                m[dR - i][tC] = m[dR][dC - i];
                m[dR][dC - i] = m[tR + i][dC];
                m[tR + i][dC] = tmp;
            }
        }
```

“之”字形打印矩阵

【题目】

给定一个矩阵 matrix，按照“之”字形的方式打印这个矩阵，例如：

1　　2　　3　　4

5　　6　　7　　8

9　　10　　11　　12

“之”字形打印的结果为：1，2，5，9，6，3，4，7，10，11，8，12

【要求】

额外空间复杂度为 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

本书提供的实现方法是这样处理的：

1．上坐标(tR,tC)初始为(0,0)，先沿着矩阵第一行移动(tC++)，当到达第一行最右边的元素后，再沿着矩阵最后一列移动(tR++)。

2．下坐标(dR,dC)初始为(0,0)，先沿着矩阵第一列移动(dR++)，当到达第一列最下边的元素时，再沿着矩阵最后一行移动(dC++)。

3．上坐标与下坐标同步移动，每次移动后的上坐标与下坐标的连线就是矩阵中的一条斜线，打印斜线上的元素即可。

4．如果上次斜线是从左下向右上打印的，这次一定是从右上向左下打印，反之亦然。总之，可以把打印的方向用 boolean 值表示，每次取反即可。

具体请参看如下代码中的 printMatrixZigZag 方法。

```
public void printMatrixZigZag(int[][] matrix) {
        int tR = 0;
        int tC = 0;
        int dR = 0;
        int dC = 0;
        int endR = matrix.length - 1;
        int endC = matrix[0].length - 1;
        boolean fromUp = false;
        while (tR != endR + 1) {
                printLevel(matrix, tR, tC, dR, dC, fromUp);
                tR = tC == endC ? tR + 1 : tR;
                tC = tC == endC ? tC : tC + 1;
                dC = dR == endR ? dC + 1 : dC;
                dR = dR == endR ? dR : dR + 1;
                fromUp = !fromUp;
        }
        System.out.println();
}

public void printLevel(int[][] m, int tR, int tC, int dR, int dC, boolean f) {
        if (f) {
                while (tR != dR + 1) {
                        System.out.print(m[tR++][tC--] + " ");
                }
        } else {
                while (dR != tR - 1) {
                        System.out.print(m[dR--][dC++] + " ");
                }
        }
}
```

找到无序数组中最小的 k 个数

【题目】

给定一个无序的整型数组 arr，找到其中最小的 k 个数。

【要求】

如果数组 arr 的长度为 N，排序之后自然可以得到最小的 k 个数，此时时间复杂度与排序的时间复杂度相同，均为 $O(N\log N)$。本题要求读者实现时间复杂度为 $O(N\log k)$和 $O(N)$的方法。

【难度】

$O(N\log k)$的方法 尉 ★★☆☆