1.2 b

1.3 e

1.4 c

1.5 思路：将该数组依次对半分割，直至分割成只有两个数的小数组。对这些数组内部进行排序，需要比较 n/2 次。长度为 2 的小数组排好序后，将其两两进行合并，取最大的两个数有序地放在合并后的长度为 4 的数组中，并不断重复此操作直至合并后的数组长度为 n，此时第二个数就是第二大的数。在合并的过程中，每次合并最多只需要进行三次比较

伪代码：

```
secondMax(low, high):
if high – low <= 1:
    return (max(A[high], A[low]), min(A[high], A[low]))
else:
    mid = (high + low) / 2
    (a1, a2) = secondMax(low, mid)
    (b1, b2) = secondMax(mid, high)
    compare a1, a2, b1, b2 to get the max and the second_max
    (x1, x2) = (max, second_max)
    return (x1, x2)
```

2.1 d

2.2 cd

2.3 daceb

2.4 eadbc

2.5 aecbd

3.1 a

3.2 思路：将当前数组由正中的元素分割为两个数组，判断正中的元素是否比它的左右元素大，若满足则找到峰值，否则继续查找元素值大于正中元素的那个数组，并以此类推。

伪代码：

```
FindMax(low, high):
    int mid = (low + high) / 2
```

```
    if A[mid] > A[mid-1] && A[mid] < A[mid+1]:
        return FindMax(mid + 1, high)
    else if A[mid] < A[mid-1] && A[mid] > A[mid+1]:
        return FindMax(low, mid - 1)
    else:
        return A[mid];
```

3.4 思路：通过分治的思想，以矩阵最中间的行和最中间的列将矩阵划分成四个部分，然后确定其中一定存在局部最小数的一部分，在这个子矩阵中继续实行分治，以此类推。

首先查找四条边界和最中间行、最中间列的所有元素中的最小值，判断它是否为局部最小数。若不满足，则通过这个最小值周围的四个方向中的最小数来确定下一步将要选择的子矩阵，继续重复此问题直至矩阵较小，可以直接得到答案为止。在首次分治时，需要查找 6n 个位置，但是在之后的所有子矩阵中，由于边界部分已经在上一次查找中完成，所以只需要访问最中间行和最中间列的元素即可。所以询问次数上限是 8n，时间复杂度为 O(n)

伪代码：

```
solve(rl, rr, cl, cr):
    if rr = rl : return A[rr][cr]
    if rr-rl = 1 :
        compare 2x2 matrix to get the min
        return min
    rmid = (rl + rr) / 2, cmid = (cl + cr) / 2
    compare rows(rr, rl, rmid) and col(cl, cr, cmid) to get the Min and its place(Minr,Minc)
    compare Min and its neighbors to get the nextmin
    if nextmin is in A11: return solve(rl, rmid-1, cl, cmid-1)
    if nextmin is in A12: return solve(rmid+1, rr, cl, cmid-1)
    ……
```