# C++方向编程题答案

## 第八周

### day48

> **25268 左右最值最大差**
>
> 链接：https://www.nowcoder.com/questionTerminal/f5805cc389394cf69d89b29c0430ff27
>
> **【题目解析】：**
>
> 要求两个最大值的差, 所以肯定可以找到整个数组的最大值, 然后就可以确定一边的最大值了;
>
> 另一边的最大值就是要找左右两端最小那个数, 因为无论他们怎么往外扩都只能增大,不能减小
>
> 所以最大差值就是第一次找出来的最大值和左右两边小的那个数的差了
>
> **【解题思路】：**
>
> 1. 按照题意, 将i为0到n-1的情况都罗列出来即可.
> 2. 创建 mxL 记录i位左侧最大值, 创建 mxR 记录i位右侧最大值. 这是在找最大值.
> 3. 核心要体会到 第 i 位左侧的最大值就是 i - 1 位最大值和 i 的值的较大值.
>
>    例如输入数据为 [2,7,3,1,1] mxL 中的值为: [2,7,7,7,7] mxR 中的值为: [7,7,3,1,1]
> 4. 然后再拿着每个位置的最大值依次去和数组的每个元素做减法, 来找到差值的最大. 牢记, 最大差值就是最大值和左右两边最小的那个数的差

```
class MaxGap {
public:
 int findMaxGap(vector<int> A, int n) {
     vector<int> mxL(n,0);//记录i位左侧最大值
     vector<int> mxR(n,0);//记录i位右侧最大值
     int ma=INT_MIN;
     for(int i=0;i<n;i++){//左侧
         if(i==0)
             mxL[i]=A[i];
         else
            mxL[i]=max(A[i],mxL[i-1]); // 第 i 位左侧的最大值就是 i - 1 位最大值和 i 的值的较
大值
     }
     for(int i=n-1;i>=0;i--){//右侧
         if(i==n-1)
             mxR[i]=A[i];
         else
             mxR[i]=max(A[i],mxR[i+1]);
     }
     // 例如输入数据为 [2,7,3,1,1]
        // mxL 中的值为: [2,7,7,7,7]
        // mxR 中的值为: [7,7,3,1,1]
     int res=INT_MIN;
     // 接下来的循环就依次求差找最大值即可.
```

```
        for(int i=0;i<n-1;i++){//求差
            res=max(res,abs(mxL[i]-mxR[i+1]));
        }
        return res;
    }
};
```

**25282 顺时针打印矩阵**

链接: https://www.nowcoder.com/questionTerminal/97e7a475d2a84eacb60ee545597a8407

**【题目解析】：**

问题本身不复杂, 主要是考虑周全各种边界条件.

**【解题思路】：**

1. 先记录左上角和右下角坐标(这两个坐标就描述了一个矩形)
2. 然后先按照顺时针打印这个矩形边上的元素
3. 缩小矩形(也就是调整左上和右下坐标位置)
4. 再次顺时针打印. 一直缩小到这个矩形为空即可.

```
// 问题本身不复杂，将思路考虑周全即可.
class Printer {
public:
    vector<int> clockwisePrint(vector<vector<int> > mat, int n, int m) {
        vector<int> ret;
        int x1 = 0, y1 = 0;       //左上角坐标
        int x2 = n-1, y2 = m-1;   //右下角坐标
        // 这两个坐标表示了一个矩形(最开始的矩阵)
        // 然后按照要求打印最外圈的数据.
        // 打印完毕之后，缩小矩形的大小.
        while(x1 <= x2 && y1 <= y2){
            for(int j = y1; j <= y2; ++j)
                ret.push_back(mat[x1][j]);
            for(int i = x1+1; i < x2; ++i)
                ret.push_back(mat[i][y2]);
            for(int j = y2; x1 < x2 && j >= y1; --j) //x1 < x2: 只有在不是单一的横行时才执行这个循环
                ret.push_back(mat[x2][j]);
            for(int i = x2-1; y1 < y2 && i > x1; --i)//y1 < y2: 只有在不是单一的竖列时才执行这个循环
                ret.push_back(mat[i][y1]);
            x1++; y1++;
            x2--; y2--;
        }
        return ret;
    }
};
```