排名	用户名	得分	完成时间	题目1 (3)	题目2 (4)	题目3 (5)	题目4 (6)
1094 / 7500	河流之王塔图姆	12	0:32:46	0:04:02	0:08:35 1	0:27:46	

2373. 矩阵中的局部最大值

直接在原数组上求解，只需去除最后的两行两列，而不用再构造用于保存结果的二维数组。

```
class Solution {
public:
    vector<vector<int>> largestLocal(vector<vector<int>>& grid) {
        int n = grid.size();
        for(int i = 0; i < n - 2; i++) {
            for(int j = 0; j < n - 2; j++) {
                for(int x = 0; x < 3; x++) {
                    for(int y = 0; y < 3; y++) {
                        grid[i][j] = max(grid[i][j], grid[i + x][j + y]);
                    }
                }
            }
            grid[i].pop_back();
            grid[i].pop_back();
        }
        grid.pop_back();
        grid.pop_back();
        return grid;
    }
};
```

2374. 边积分最高的节点

数据范围 n <= **10^5**, 极端情况下会达到 **n^2**， 因此需要使用**long long**。

```
class Solution {
public:
    int edgeScore(vector<int>& edges) {
        int n = edges.size(), res = 0;
        long long maxn = 0;
        vector<long long> scores(n, 0);
        for(int i = 0; i < n; i++) {
            scores[edges[i]] += i;
        }
        for(int i = 0; i < n; i++) {
            if(maxn < scores[i]) {
                maxn = scores[i];
                res = i;
            }
        }
        return res;
    }
```

```
};
```

2375. 根据模式串构造最小数字

可以采用**深搜+回溯**的方式，结合**贪心**思想，每次选择符合升降条件的最小未被选择的数字，得到的第一个结果即为字典序最小的答案。

本题也可以直接采用**贪心**思想，保证遍历模式字符串第`i`位时，生成的长度为`i + 1`的中间字符串是由`1~ i + 1`构成的字典序最小的字符串，直至最终遍历模式串结束。因此我们可以先生成一个长度为`n + 1`的最小字符串，在遍历模式串时遇到`D`进行***reverse***操作，连续`k`个`'D'`就翻转对应位置上长度为`k + 1`的子串。

```
class Solution {
public:
    string smallestNumber(string pattern) {
        int n = pattern.size(), pre = 0;
        string res;
        for(int i = 0; i <= n; i++){
            res += '1' + i;
        }
        for(int i = 0; i < n; i++) {
            if(pattern[i] == 'I') {
                reverse(res.begin() + pre, res.begin() + i + 1);
                pre = i + 1;
            }
        }
        reverse(res.begin() + pre, res.end());
        return res;
    }
};
```

2376. 统计特殊整数

数位DP题目，感谢零神的模板

采用**记忆化搜索**解决数位DP问题。为了方便获取每一位，可以将数字转化为字符串。明确一点，只有**遍历到边界上，下一位才会有范围限制。**举例来说`n = 123`，当前搜索到`12`时，下一位的取值范围才会被限制为`0 ~ 3`，否则没有限制，为`0 ~ 9`

> int dfs(int k, int mask, bool is_limit, bool is_num);

- `k`记录搜索深度，即遍历到哪一位
- `mask`用来满足题目中的要求，在此题中采用**位运算**的方式保证每位上数字不能重复**（可选）**
- `is_limit`表示当前搜索是否在边界上
- `is_num`记录是否取过数字，即排除前导0对本题目中数字不重复要求的影响 **（可选）**

时间复杂度为 **状态个数 * 转移个数**，在本题中为`log(n) * 1024 * 10`

```
class Solution {
public:
    string s;
```

```
    int dp[10][1 << 10], len;
    int dfs(int k, int mask, bool is_limit, bool is_num) {
        if(k == len) return is_num;
        if(!is_limit && is_num && dp[k][mask] != -1) return dp[k][mask];

        int res = 0;
        //若当前还未取过数字，可以继续不取数字
        if(!is_num) res += dfs(k + 1, 0, false, false);
        //取数字
        for(int i = 1 - is_num, up = is_limit ? s[k] - '0' : 9; i <= up; i++) {
            if((mask >> i & 1) == 0) {
                res += dfs(k + 1, mask | (1 << i), is_limit && i == up, true);
            }
        }
        //搜索到边界或者没有取过数字的情况只会出现一次，因此不需要记忆化
        if(!is_limit && is_num) dp[k][mask] = res;
        return res;
    }
    int countSpecialNumbers(int n) {
        s = to_string(n);
        len = s.size();
        memset(dp, -1, sizeof(dp));
        return dfs(0, 0, true, false);
    }
};
```

扩展题目：

233. 数字 1 的个数

600. 不含连续1的非负整数

902. 最大为 N 的数字组合