LeetCode 1431. 拥有最多糖果的孩子

1431. 拥有最多糖果的孩子

难度 简单 4 收藏 分享 切换为英文 关注 反馈

给你一个数组 `candies` 和一个整数 `extraCandies` ，其中 `candies[i]` 代表第 `i` 个孩子拥有的糖果数目。

对每一个孩子，检查是否存在一种方案，将额外的 `extraCandies` 个糖果分配给孩子们之后，此孩子有 **最多** 的糖果。注意，允许有多个孩子同时拥有 **最多** 的糖果数目。

示例 1：

输入： candies = [2,3,5,1,3], extraCandies = 3
输出： [true,true,true,false,true]
解释：
孩子 1 有 2 个糖果，如果他得到所有额外的糖果（3个），那么他总共有 5 个糖果，他将成为拥有最多糖果的孩子。
孩子 2 有 3 个糖果，如果他得到至少 2 个额外糖果，那么他将成为拥有最多糖果的孩子。
孩子 3 有 5 个糖果，他已经是拥有最多糖果的孩子。
孩子 4 有 1 个糖果，即使他得到所有额外的糖果，他也只有 4 个糖果，无法成为拥有糖果最多的孩子。
孩子 5 有 3 个糖果，如果他得到至少 2 个额外糖果，那么他将成为拥有最多糖果的孩子。

示例 2：

输入： candies = [4,2,1,1,2], extraCandies = 1
输出： [true,false,false,false,false]
解释： 只有 1 个额外糖果，所以不管额外糖果给谁，只有孩子 1 可以成为拥有糖果最多的孩子。

示例 3：

输入： candies = [12,1,12], extraCandies = 10
输出： [true,false,true]

```cpp
class Solution {
public:
    vector<bool> kidsWithCandies(vector<int>& candies, int extraCandies) {
        int n = candies.size();

        int ma = candies[0];
        for (int i = 1; i < n; i++)
            ma = max(ma, candies[i]);

        vector<bool> ans(n, false);
        for (int i = 0; i < n; i++)
            if (candies[i] + extraCandies >= ma)
                ans[i] = true;

        return ans;
```

```
    }
};
```

1432. 改变一个整数能得到的最大差值

1432. 改变一个整数能得到的最大差值

难度 中等 3 收藏 分享 切换为英文 关注 反馈

给你一个整数 `num` 。你可以对它进行如下步骤恰好 **两次** ：

- 选择一个数字 `x (0 <= x <= 9)`.
- 选择另一个数字 `y (0 <= y <= 9)` 。数字 `y` 可以等于 `x` 。
- 将 `num` 中所有出现 `x` 的数位都用 `y` 替换。
- 得到的新的整数 **不能** 有前导 0 ，得到的新整数也 **不能** 是 0 。

令两次对 `num` 的操作得到的结果分别为 `a` 和 `b` 。

请你返回 `a` 和 `b` 的 **最大差值** 。

示例 1:

```
输入：num = 555
输出：888
解释：第一次选择 x = 5 且 y = 9 ，并把得到的新数字保存在 a 中。
第二次选择 x = 5 且 y = 1 ，并把得到的新数字保存在 b 中。
现在，我们有 a = 999 和 b = 111 ，最大差值为 888
```

示例 2:

```
输入：num = 9
输出：8
解释：第一次选择 x = 9 且 y = 9 ，并把得到的新数字保存在 a 中。
第二次选择 x = 9 且 y = 1 ，并把得到的新数字保存在 b 中。
现在，我们有 a = 9 和 b = 1 ，最大差值为 8
```

示例 3:

```
输入：num = 123456
输出：820000
```

```cpp
class Solution {
public:
    int change(int x, int y, string s) {
        for (char &c : s)
            if (c == x + '0')
                c = y + '0';

        return stoi(s);
    }

    int maxDiff(int num) {
        string s = to_string(num);
        int a = num;
```

```
        for (int i = 0; i < s.size(); i++)
            if (s[i] < '9') {
                a = change(s[i] - '0', 9, s);
                break;
            }

        int b = num;
        if (s[0] > '1')
            b = change(s[0] - '0', 1, s);
        else {
            for (int i = 1; i < s.size(); i++)
                if (s[i] > '1') {
                    b = change(s[i] - '0', 0, s);
                    break;
                }
        }

        return a - b;
    }
};
```

1433. 检查一个字符串是否可以打破另一个字符串

1433. 检查一个字符串是否可以打破另一个字符串

难度 中等 4 收藏 分享 切换为英文 关注 反馈

给你两个字符串 `s1` 和 `s2` ，它们长度相等，请你检查是否存在一个 `s1` 的排列可以打破 `s2` 的一个排列，或者是否存在一个 `s2` 的排列可以打破 `s1` 的一个排列。

字符串 `x` 可以打破字符串 `y` （两者长度都为 `n` ）需满足对于所有 `i` （在 `0` 到 `n - 1` 之间）都有 `x[i] >= y[i]` （字典序意义下的顺序）。

示例 1：

输入： `s1 = "abc", s2 = "xya"`
输出： `true`
解释： `"ayx" 是 s2="xya" 的一个排列，"abc" 是字符串 s1="abc" 的一个排列，且 "ayx" 可以打破 "abc" 。`

示例 2：

输入： `s1 = "abe", s2 = "acd"`
输出： `false`
解释： `s1="abe" 的所有排列包括："abe"，"aeb"，"bae"，"bea"，"eab" 和 "eba" ，s2="acd" 的所有排列包括："acd"，"adc"，"cad"，"cda"，"dac" 和 "dca"。然而没有任何 s1 的排列可以打破 s2 的排列。也没有 s2 的排列能打破 s1 的排列。`

示例 3：

输入： `s1 = "leetcodee", s2 = "interview"`
输出： `true`

```cpp
class Solution {
public:
    void csort(string &s) {
        int n = s.size();
        vector<int> c(26, 0);
        for (int i = 0; i < n; i++)
            c[s[i] - 'a']++;

        for (int i = 25; i >= 0; i--)
            while (c[i]--)
                s[--n] = i + 'a';
    }

    bool checkIfCanBreak(string s1, string s2) {
        csort(s1);
        csort(s2);

        int n = s1.size();
        bool f1 = true, f2 = true;
        for (int i = 0; i < n; i++)
            if (s1[i] < s2[i]) f1 = false;
            else if (s1[i] > s2[i]) f2 = false;

        return f1 || f2;
    }
};
```

1434. 每个人戴不同帽子的方案数

1434. 每个人戴不同帽子的方案数

难度 困难 20 收藏 分享 切换为英文 关注 反馈

总共有 `n` 个人和 `40` 种不同的帽子，帽子编号从 `1` 到 `40` 。

给你一个整数列表的列表 `hats` ，其中 `hats[i]` 是第 `i` 个人所有喜欢帽子的列表。

请你给每个人安排一顶他喜欢的帽子，确保每个人戴的帽子跟别人都不一样，并返回方案数。

由于答案可能很大，请返回它对 `10^9 + 7` 取余后的结果。

示例 1：

```
输入：hats = [[3,4],[4,5],[5]]
输出：1
解释：给定条件下只有一种方法选择帽子。
第一个人选择帽子 3，第二个人选择帽子 4，最后一个人选择帽子 5。
```

示例 2：

```
输入：hats = [[3,5,1],[3,5]]
输出：4
解释：总共有 4 种安排帽子的方法：
(3,5)，(5,3)，(1,3) 和 (1,5)
```

示例 3：

```
输入：hats = [[1,2,3,4],[1,2,3,4],[1,2,3,4],[1,2,3,4]]
输出：24
解释：每个人都可以从编号为 1 到 4 的帽子中选。
(1,2,3,4) 4 个帽子的排列方案数为 24 。
```

示例 4：

```
输入：hats = [[1,2,3],[2,3,5,6],[1,3,7,9],[1,8,9],[2,5,7]]
输出：111
```

```cpp
class Solution {
public:
    int numberWays(vector<vector<int>>& hats) {
        const int mod = 1000000007;
        int n = hats.size();
        vector<vector<int>> f(41, vector<int>(1 << n, 0));
        vector<vector<bool>> h(n, vector<bool>(41, false));

        for (int i = 0; i < n; i++)
            for (int j : hats[i])
                h[i][j] = true;

        f[0][0] = 1;

        for (int i = 1; i <= 40; i++)
            for (int s = 0; s < (1 << n); s++) {
                f[i][s] = (f[i][s] + f[i - 1][s]) % mod;
                for (int j = 0; j < n; j++)
```

```
                    if ((s & (1 << j)) && h[j][i])
                        f[i][s] = (f[i][s] + f[i - 1][s ^ (1 << j)]) % mod;
            }

        return f[40][(1 << n) - 1];
    }
};
```