LeetCode 1436. 旅行终点站

1436. 旅行终点站

难度 简单 2 收藏 分享 切换为英文 关注 反馈

给你一份旅游线路图，该线路图中的旅行线路用数组 `paths` 表示，其中 `paths[i] = [cityA_i, cityB_i]` 表示该线路将会从 `cityA_i` 直接前往 `cityB_i` 。请你找出这次旅行的终点站，即没有任何可以通往其他城市的线路的城市。

题目数据保证线路图会形成一条不存在循环的线路，因此只会有一个旅行终点站。

示例 1：

输入： paths = [["London","New York"],["New York","Lima"],["Lima","Sao Paulo"]]
输出： "Sao Paulo"
解释： 从 "London" 出发，最后抵达终点站 "Sao Paulo" 。本次旅行的路线是 "London" -> "New York" -> "Lima" -> "Sao Paulo" 。

示例 2：

输入： paths = [["B","C"],["D","B"],["C","A"]]
输出： "A"
解释： 所有可能的线路是：
"D" -> "B" -> "C" -> "A".
"B" -> "C" -> "A".
"C" -> "A".
"A".
显然，旅行终点站是 "A" 。

示例 3：

输入： paths = [["A","Z"]]
输出： "Z"

```cpp
class Solution {
public:
    string destCity(vector<vector<string>>& paths) {
        unordered_set<string> s;

        for (const auto &v : paths)
            s.insert(v[0]);

        for (const auto &v : paths)
            if (s.find(v[1]) == s.end())
                return v[1];

        return "";
    }
};
```

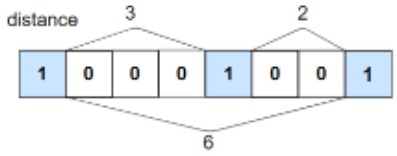
1437. 是否所有 1 都至少相隔 k 个元素

1437. 是否所有 1 都至少相隔 k 个元素

难度 中等 1 收藏 分享 切换为英文 关注 反馈

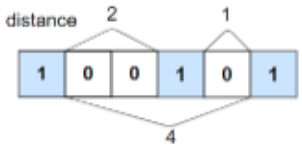
给你一个由若干 `0` 和 `1` 组成的数组 `nums` 以及整数 `k`。如果所有 `1` 都至少相隔 `k` 个元素，则返回 `True` ；否则，返回 `False` 。

示例 1：




```
输入：nums = [1,0,0,0,1,0,0,1], k = 2
输出：true
解释：每个 1 都至少相隔 2 个元素。
```

示例 2：




```
输入：nums = [1,0,0,1,0,1], k = 2
输出：false
解释：第二个 1 和第三个 1 之间只隔了 1 个元素。
```

示例 3：

```
输入：nums = [1,1,1,1,1], k = 0
输出：true
```

```cpp
class Solution {
public:
    bool kLengthApart(vector<int>& nums, int k) {
        int n = nums.size();
        int last = -1;
        for (int i = 0; i < n; i++)
            if (nums[i] == 1) {
                if (last != -1 && i - last <= k)
                    return false;
                last = i;
            }

        return true;
    }
};
```

1438. 绝对差不超过限制的最长连续子数组

1438. 绝对差不超过限制的最长连续子数组

难度 中等 19 收藏 分享 切换为英文 关注 反馈

给你一个整数数组 `nums` ，和一个表示限制的整数 `limit`，请你返回最长连续子数组的长度，该子数组中的任意两个元素之间的绝对差必须小于或者等于 `limit` 。

如果不存在满足条件的子数组，则返回 `0` 。

示例 1：

输入： nums = [8,2,4,7], limit = 4
输出： 2
解释： 所有子数组如下：
[8] 最大绝对差 |8-8| = 0 <= 4.
[8,2] 最大绝对差 |8-2| = 6 > 4.
[8,2,4] 最大绝对差 |8-2| = 6 > 4.
[8,2,4,7] 最大绝对差 |8-2| = 6 > 4.
[2] 最大绝对差 |2-2| = 0 <= 4.
[2,4] 最大绝对差 |2-4| = 2 <= 4.
[2,4,7] 最大绝对差 |2-7| = 5 > 4.
[4] 最大绝对差 |4-4| = 0 <= 4.
[4,7] 最大绝对差 |4-7| = 3 <= 4.
[7] 最大绝对差 |7-7| = 0 <= 4.
因此，满足题意的最长子数组的长度为 2 。

示例 2：

输入： nums = [10,1,2,4,7,2], limit = 5
输出： 4
解释： 满足题意的最长子数组是 [2,4,7,2]，其最大绝对差 |2-7| = 5 <= 5 。

示例 3：

输入： nums = [4,2,2,2,4,4,2,2], limit = 0
输出： 3

```
class Solution {
public:
    int longestSubarray(vector<int>& nums, int limit) {
        int n = nums.size();
        deque<int> m1, m2;

        int ans = 0;
        for (int i = 0, j = 0; i < n; i++) {
            while (!m1.empty() && nums[i] >= nums[m1.back()])
                m1.pop_back();

            m1.push_back(i);

            while (!m2.empty() && nums[i] <= nums[m2.back()])
                m2.pop_back();

            m2.push_back(i);
```

```
            while (!m1.empty() && !m2.empty()
                   && nums[m1.front()] - nums[m2.front()] > limit) {
                j++;
                while (!m1.empty() && m1.front() < j)
                    m1.pop_front();

                while (!m2.empty() && m2.front() < j)
                    m2.pop_front();
            }

            ans = max(ans, i - j + 1);
        }

        return ans;
    }
};
```

1439. 有序矩阵中的第 k 个最小数组和

1439. 有序矩阵中的第 k 个最小数组和

难度 困难 10 收藏 分享 切换为英文 关注 反馈

给你一个 `m * n` 的矩阵 `mat`，以及一个整数 `k`，矩阵中的每一行都以非递减的顺序排列。

你可以从每一行中选出 1 个元素形成一个数组。返回所有可能数组中的第 k 个 **最小** 数组和。

示例 1：

输入： `mat = [[1,3,11],[2,4,6]], k = 5`
输出： `7`
解释： 从每一行中选出一个元素，前 `k` 个和最小的数组分别是：
`[1,2], [1,4], [3,2], [3,4], [1,6]`。其中第 `5` 个的和是 `7` 。

示例 2：

输入： `mat = [[1,3,11],[2,4,6]], k = 9`
输出： `17`

示例 3：

输入： `mat = [[1,10,10],[1,4,5],[2,3,6]], k = 7`
输出： `9`
解释： 从每一行中选出一个元素，前 `k` 个和最小的数组分别是：
`[1,1,2], [1,1,3], [1,4,2], [1,4,3], [1,1,6], [1,5,2], [1,5,3]`。其中第 `7` 个的和是 `9` 。

示例 4：

输入： `mat = [[1,1,10],[2,2,9]], k = 7`
输出： `12`

```
class Solution {
public:
```

```cpp
    int kthSmallest(vector<vector<int>>& mat, int k) {
        int m = mat.size();
        int n = mat[0].size();

        #define PIV pair<int, vector<int>>

        priority_queue<PIV, vector<PIV>, greater<PIV>> q;
        set<vector<int>> vis;

        vector<int> idx(m, 0);
        int sum = 0;

        for (int i = 0; i < m; i++)
            sum += mat[i][0];

        q.push(make_pair(sum, idx));

        while (--k) {
            sum = q.top().first;
            idx = q.top().second;

            q.pop();

            for (int i = 0; i < m; i++) {
                int j = idx[i];
                if (j + 1 < n) {
                    idx[i]++;
                    if (vis.find(idx) == vis.end()) {
                        vis.insert(idx);
                        q.push(make_pair(sum - mat[i][j] + mat[i][j + 1], idx));
                    }
                    idx[i]--;
                }
            }
        }

        return q.top().first;
    }
};
```