**Avoid Flood in The City**

**Medium**

Your country has an infinite number of lakes. Initially, all the lakes are empty, but when it rains over the nth lake, the nth lake becomes full of water. If it rains over a lake which is **full of water**, there will be a **flood**. Your goal is to avoid the flood in any lake.

Given an integer array rains where:

* rains[i] > 0 means there will be rains over the rains[i] lake.
* rains[i] == 0 means there are no rains this day and you can choose **one lake** this day and **dry it**.

Return *an array ans* where:

* ans.length == rains.length
* ans[i] == -1 if rains[i] > 0.
* ans[i] is the lake you choose to dry in the ith day if rains[i] == 0.

If there are multiple valid answers return **any** of them. If it is impossible to avoid flood return **an empty array**.

Notice that if you chose to dry a full lake, it becomes empty, but if you chose to dry an empty lake, nothing changes. (see example 4)

**Example 1:**

**Input:** rains = [1,2,3,4]

**Output:** [-1,-1,-1,-1]

**Explanation:** After the first day full lakes are [1]

After the second day full lakes are [1,2]

After the third day full lakes are [1,2,3]

After the fourth day full lakes are [1,2,3,4]

There's no day to dry any lake and there is no flood in any lake.

**Example 2:**

**Input:** rains = [1,2,0,0,2,1]

**Output:** [-1,-1,2,1,-1,-1]

**Explanation:** After the first day full lakes are [1]

After the second day full lakes are [1,2]

After the third day, we dry lake 2. Full lakes are [1]

After the fourth day, we dry lake 1. There is no full lakes.

After the fifth day, full lakes are [2].

After the sixth day, full lakes are [1,2].

It is easy that this scenario is flood-free. [-1,-1,1,2,-1,-1] is another acceptable scenario.

**Example 3:**

**Input:** rains = [1,2,0,1,2]

**Output:** []

**Explanation:** After the second day, full lakes are [1,2]. We have to dry one lake in the third day.

After that, it will rain over lakes [1,2]. It's easy to prove that no matter which lake you choose to dry in the 3rd day, the other one will flood.

**Constraints:**

* 1 <= rains.length <= 105
* 0 <= rains[i] <= 109
* class Solution {
* public:
* vector<int> avoidFlood(vector<int>& rains) {
* int count=0, n=rains.size(), non=1;
* unordered\_map<int, int> m;
* set<int> s;
* vector<int> ans(n, 0);
* for (int i=0; i<n; i++) {
* if (rains[i]==0) {
* s.insert(i);
* continue;
* }
* if (m.count(rains[i])==0) m[rains[i]]=i;
* else {
* auto it=s.lower\_bound(m[rains[i]]);
* if (it==s.end()) return {};
* else {
* ans[\*it]=rains[i];
* m[rains[i]]=i;
* s.erase(it);
* }
* }
* ans[i]=-1;
* non=rains[i];
* }
* for (int i=0; i<n; i++) {
* if (ans[i]==0) ans[i]=non;
* }
* return ans;
* }
* };