Question: Given a positive integer num, write a function which returns True if num is a perfect square else False.
Solution: class Solution { public: bool isPerfectSquare(int num) { int lo = 0, hi = num; while (lo <= hi) { int mid = lo + (hi - lo) / 2; long long s = (long long)mid * mid; if (s == num) { return true; } else if (s > num) { hi = mid - 1; } else { lo = mid + 1; } } return (long long)hi * hi == num; } };
Question:You have a total of n coins that you want to form in a staircase shape, where every k-th row must have exactly k coins. Given n, find the total number of full staircase rows that can be formed. n is a non-negative integer and fits within the range of a 32-bit signed integer.
Solution:class Solution { public: int arrangeCoins(int n) { long lo = 0, hi = n; while (lo <= hi) { long mid = lo + (hi - lo) / 2; if ((mid + 1) * mid / 2 > n) { hi = mid - 1; } else { lo = mid + 1; } } return lo - 1; }
Question:You are given a map in form of a two-dimensional integer grid where 1 represents land and 0 represents water. Grid cells are connected horizontally/vertically (not diagonally). The grid is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells). The island doesn't have "lakes" (water inside that isn't connected to the water around the island). One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.
Solution:class Solution { public: int islandPerimeter(vector>& grid) { int ret = 0, m = grid.size(), n = grid[0].size(); for (int i = 0; i < m; ++i) { for (int j = 0; j < n; ++j) { if (grid[i][j] == 1) { if (i - 1 < 0) ++ret; if (i + 1 == m) ++ret; if (i - 1 >= 0 && grid[i - 1][j] == 0) ++ret; if (i + 1 < m && grid[i + 1][j] == 0) ++ret; if (j - 1 < 0) ++ret; if (j + 1 == n) ++ret; if (j - 1 >= 0 && grid[i][j - 1] == 0) ++ret; if (j + 1 < n && grid[i][j + 1] == 0) ++ret; } } } return ret; } };
Question:We define a harmounious array as an array where the difference between its maximum value and its minimum value is exactly 1. Now, given an integer array, you need to find the length of its longest harmonious subsequence among all its possible subsequences.
Solution:class Solution { public: int findLHS(vector& nums) { if (nums.empty()) return 0; map cnt; for (auto& n : nums) ++cnt[n]; int ret = 0; for (auto& m : cnt) if (cnt[m.first - 1] > 0 && cnt[m.first] > 0) ret = max(ret, cnt[m.first - 1] + cnt[m.first]); return ret; } };
Question: Given a Binary Search Tree and a target number, return true if there exist two elements in the BST such that their sum is equal to the given target.
Solution:class Solution { public: bool findTarget(TreeNode* root, int k) { vector nums; tree2vector(root, nums); for (int l = 0, r = nums.size() - 1; l < r;) { int s = nums[l] + nums[r]; if (s == k) { return true; } else if (s < k) { ++l; } else { --r; } } return false; } void tree2vector(TreeNode* root, vector& nums) { if (root == nullptr) return; tree2vector(root->left, nums); nums.push_back(root->val); tree2vector(root->right, nums); } };
Question:Given an array of integers A sorted in non-decreasing order, return an array of the squares of each number, also in sorted non-decreasing order.
Solution: class Solution { public: vector sortedSquares(vector& A) {vector ret; for (int l = 0, r = A.size() - 1; l <= r;) { if (abs(A[l]) > abs(A[r])) { ret.push_back(pow(A[l], 2)); ++l; } else { ret.push_back(pow(A[r], 2)); --r; } } reverse(ret.begin(), ret.end()); return ret; } };