# 题目

给定一个正整数数组 w ，其中 w[i] 代表下标 i 的权重（下标从 0 开始），请写一个函数 pickIndex ，它可以随机地获取下标 i，选取下标 i 的概率与 w[i] 成正比。

例如，对于 w = [1, 3]，挑选下标 0 的概率为 1 / (1 + 3) = 0.25 （即，25%），而选取下标 1 的概率为 3 / (1 + 3) = 0.75（即，75%）。

也就是说，选取下标 i 的概率为 w[i] / sum(w) 。

示例 1：

输入：

["Solution","pickIndex"]

[[[1]],[]]

输出：

[null,0]

解释：

Solution solution = new Solution([1]);

solution.pickIndex(); // 返回 0，因为数组中只有一个元素，所以唯一的选择是返回下标 0。

示例 2：

输入：

["Solution","pickIndex","pickIndex","pickIndex","pickIndex","pickIndex"]

[[[1,3]],[],[],[],[],[]]

输出：

[null,1,1,1,1,0]

解释：

Solution solution = new Solution([1, 3]);

solution.pickIndex(); // 返回 1，返回下标 1，返回该下标概率为 3/4 。

solution.pickIndex(); // 返回 1

solution.pickIndex(); // 返回 1

solution.pickIndex(); // 返回 1

solution.pickIndex(); // 返回 0，返回下标 0，返回该下标概率为 1/4 。

由于这是一个随机问题，允许多个答案，因此下列输出都可以被认为是正确的:

[null,1,1,1,1,0]

[null,1,1,1,1,1]

[null,1,1,1,0,0]

[null,1,1,1,0,1]

[null,1,0,1,0,0]

......

诸若此类。

提示：

1 <= w.length <= 10000

1 <= w[i] <= 10^5

pickIndex 将被调用不超过 10000 次

# 分析

class Solution {

public:

int range = 0;

vector<int> weights;

Solution(vector<int>& w) {

for (auto w\_ : w) {

weights.push\_back(range);

range += w\_;

}

}

int pickIndex() {

int rnd = rand() % range;

int i = 0;

for (i = 1; i < weights.size(); i++) {

if (rnd < weights[i]) {

break;

}

}

return i-1;

}

};

另一种写法：

class Solution {

public:

Solution(vector<int>& w) {

n = w.size();

dp[0] = w[0];

for(int i = 1; i < n; ++i) dp[i] = w[i] + dp[i - 1]; //此处求出前缀和

}

int dp[10001];

int n;

int pickIndex() {

return lower\_bound(dp, dp + n, (random() % dp[n - 1]) + 1) - dp;

}

};