

Random Pick with Weight

Try to solve the Random Pick with Weight problem.

We'll cover the following

- Statement
- Examples
- Understanding the problem
- Figure it out!
- Try it yourself

Statement

You're given an array of positive integers, `weights`, where `weights[i]` is the weight of the i^{th} index.

Write a function, **Pick Index()**, which performs weighted random selection to return an index from the `weights` array. The larger the value of `weights[i]`, the heavier the weight is, and the higher the chances of its index being picked.

Suppose that the array consists of the weights `[12, 84, 35]`. In this case, the probabilities of picking the indexes will be as follows:

- **Index 0:** $12 / (12 + 84 + 35) = 9.2\%$
- **Index 1:** $84 / (12 + 84 + 35) = 64.1\%$
- **Index 2:** $35 / (12 + 84 + 35) = 26.7\%$

Constraints:

- $1 \leq \text{weights.length} \leq 10^4$
- $1 \leq \text{weights}[i] \leq 10^5$
- **Pick Index()** will be called at most 10^4 times.

Note: Since we're randomly choosing from the options, there is no guarantee that in any *specific* run of the program, any of the elements will be selected with the exact expected frequency.

Examples

Sample example 1

Input

	[0]	[1]	[2]
weights	1	2	3

Index 2 has the highest weight, so its probability of being picked is higher than the other elements.

1 of 3



Understanding the problem

Let's take a moment to make sure you've correctly understood the problem. The quiz below helps you check if you're solving the correct problem:

Random Pick with Weight

1

Given this list of weights, which index has the highest probability of being picked? Note that indexes start at 0.

weights = [5, 6, 10, 8, 9, 7]

A) 0

B) 3

C) 4

D) 2

Submit Answer



Question 1 of 3
0 attempted



Reset Quiz ↺

Figure it out!

We have a game for you to play. Rearrange the logical building blocks to develop a clearer understanding of how to solve this problem.



Drag and drop the cards to rearrange them in the correct sequence.

Generate a list of running sums from the given list of weights.



Generate a random number, where the range for the random number will be from 0 to the maximum number in the list of running sums.

Use binary search to find the index of the first running sum that is greater than the random value.

Return the index found.

Reset

Show Solution

Submit

Try it yourself

Since a good randomized picking function shouldn't result in elements being picked with precisely the same frequencies as predicted mathematically, we print the frequency with which each element is picked as a result of calling the **Pick Index()** function 900 times for each list. Next to the actual frequency, we print the expected frequency. The better our function is, the more closely it matches the expected frequencies over several runs.

You are expected to implement a class whose constructor receives the list of weights and has a method **Pick Index()** that picks an index at random, taking into account the weight of each index.

Java



```
4
5 public RandomPickWithWeight(int[] weights) {
6     // Write your code here
7     // The integer's weight array is passed to the constructor
8
9 }
10
11 public int pickIndex() {
12     // Write your code here
13     // Currently returning the first index
14     // Your function should implement the required solution to this problem
15     return 0;
16 }
17
18 public static int sumW(int[] arr) {
19     int sum = 0;
20     // Loop through the array to calculate sum of elements
21     for (int i = 0; i < arr.length; i++) {
22         sum = sum + arr[i];
23     }
24     return sum;
25 }
26
27 static float round(float var) {
28     float value = (int)(var * 100 + 5);
```



← Back

Solution: First Bad Ver...

Next →

Solution: Random Pick...

☒ Mark as
Completed