Daily Coding Problem #152

Problem

This problem was asked by Triplebyte.

You are given n numbers as well as n probabilities that sum up to 1. Write a function to generate one of the numbers with its corresponding probability.

For example, given the numbers `[1, 2, 3, 4]` and probabilities `[0.1, 0.5, 0.2, 0.2]`, your function should return 1 10% of the time, 2 50% of the time, and 3 and 4 20% of the time.

You can generate random numbers between 0 and 1 uniformly.

Solution

One way to solve this problem would be to imagine all the probabilities as distinct, disjoint intervals. For example, given the probabilities `[0.1, 0.5, 0.2, 0.2]`, you would get the intervals `[0, 0.1)`, `[0.1, 0.6)`, `[0.6, 0.8)`, `[0.8, 1]`. Then we generate a uniform random between 0 and 1 and select whichever value corresponding to the interval we fall in. That would look like this:

```
from random import random

def distribute(nums, probs):
    r = random()

    s = 0
    for num, prob in zip(nums, probs):
        s += prob
        if s >= r:
            return num
```

This would take O(n) time and constant space. However, we can speed this up by preprocessing our list of probabilities into an array, and then binary searching over the array to find our value.

```
from random import random
from bisect import bisect_left

def preprocess(probs):
    lst = []

    current_val = 0
    for p in probs:
        current_val += p
        lst.append(current_val)

    return lst

def distribute(nums, arr):
    r = random()
    i = bisect_left(arr, r)
    return nums[i]
```

The preprocessing takes O(n) time and space, but after that, queries take only O(log n) time.