

Random number generator in arbitrary probability distribution fashion

Given n numbers, each with some frequency of occurrence. Return a random number with probability proportional to its frequency of occurrence.

Example:

Let following be the given numbers.

```
arr[] = {10, 30, 20, 40}
```

Let following be the frequencies of given numbers.

```
freq[] = {1, 6, 2, 1}
```

The output should be

10 with probability 1/10

30 with probability 6/10

20 with probability 2/10

40 with probability 1/10

It is quite clear that the simple random number generator won't work here as it doesn't keep track of the frequency of occurrence.

We need to somehow transform the problem into a problem whose solution is known to us.

One simple method is to take an auxiliary array (say `aux[]`) and duplicate the numbers according to their frequency of occurrence. Generate a random number (say r) between 0 to $\text{Sum}-1$ (including both), where Sum represents summation of frequency array (`freq[]` in above example). Return the random number `aux[r]` (Implementation of this method is left as an exercise to the readers).

The limitation of the above method discussed above is huge memory

consumption when frequency of occurrence is high. If the input is 997, 8761 and 1, this method is clearly not efficient.

How can we reduce the memory consumption? Following is detailed algorithm that uses $O(n)$ extra space where n is number of elements in input arrays.

1. Take an auxiliary array (say `prefix[]`) of size n .
2. Populate it with prefix sum, such that `prefix[i]` represents sum of numbers from 0 to i .
3. Generate a random number (say r) between 1 to Sum (including both), where Sum represents summation of input frequency array.
4. Find index of Ceil of random number generated in step #3 in the prefix array. Let the index be `indexc`.
5. Return the random number `arr[indexc]`, where `arr[]` contains the input n numbers.

Before we go to the implementation part, let us have quick look at the algorithm with an example:

`arr[]`: {10, 20, 30}

`freq[]`: {2, 3, 1}

`Prefix[]`: {2, 5, 6}

Since last entry in prefix is 6, all possible values of r are [1, 2, 3, 4, 5, 6]

- 1: Ceil is 2. Random number generated is 10.
- 2: Ceil is 2. Random number generated is 10.
- 3: Ceil is 5. Random number generated is 20.
- 4: Ceil is 5. Random number generated is 20.
- 5: Ceil is 5. Random number generated is 20.
- 6: Ceil is 6. Random number generated is 30.

In the above example

- 10 is generated with probability $2/6$.
- 20 is generated with probability $3/6$.
- 30 is generated with probability $1/6$.

How does this work?

Any number `input[i]` is generated as many times as its frequency of occurrence because there exists count of integers in range(`prefix[i - 1]`, `prefix[i]`) is `input[i]`.

```
//C program to generate random numbers according to given range
#include <stdio.h>
#include <stdlib.h>
```

```
// The main function that returns a random number from a
// distribution array defined by freq[]. n is size of arr
int myRand(int arr[], int freq[], int n)
{
    // Create and fill prefix array
    int prefix[n], i;
    prefix[0] = freq[0];
    for (i = 1; i < n; ++i)
        prefix[i] = prefix[i - 1] + freq[i];

    // prefix[n-1] is sum of all frequencies. Generate a
    // with value from 1 to this sum
    int r = (rand() % prefix[n - 1]) + 1;


    // Find index of ceiling of r in prefix array
    int indexc = findCeil(prefix, r, 0, n - 1);
    return arr[indexc];
}
```

data:text/html;charset=utf-8,%3Cdiv%20class%3D%22post-title-info%22%20style%3D%22float%3A%20left%3B%20font-size%... 3/4

```
srand(time(NULL));

// Let us generate 10 random numbers according to
// given distribution
for (i = 0; i < 5; i++)
    printf("%d\n", myRand(arr, freq, n));

return 0;
}
```



Output: May be different for different runs

```
4
3
4
4
4
```