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 Sum-1(including both), where Sum represents summation of frequency array (freq[] in above example). Return the

Google™ Custom Search

GeeksforGeeks

53,525 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

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 index**c**.
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]. Like in the above example 3 is

Popular Posts

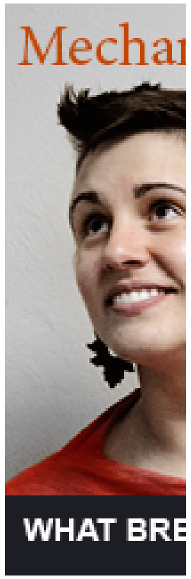
generated thrice, as there exists 3 integers 3, 4 and 5 whose ceil is 5.

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

// Utility function to find ceiling of r in arr[l..h]
int findCeil(int arr[], int r, int l, int h)
{
    int mid;
    while (l < h)
    {
         mid = l + ((h - l) >> 1);  // Same as mid = (l+h)/2
        (r > arr[mid]) ? (l = mid + 1) : (h = mid);
    }
    return (arr[l] >= r) ? l : -1;
}

// The main function that returns a random number from arr[] according
// distribution array defined by freq[]. n is size of arrays.
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 random number
    // with value from 1 to this sum
    int r = (rand() % prefix[n - 1]) + 1;

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

// Driver program to test above functions
int main()
{
    int arr[]  = {1, 2, 3, 4};
    int freq[] = {10, 5, 20, 100};
    int i, n = sizeof(arr) / sizeof(arr[0]);

    // Use a different seed value for every run.
    srand(time(NULL));
    // Let us generate 10 random numbers accroding 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
```

This article is compiled by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 12 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 52 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 56 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

23 Tweet 4 2

Writing code in comment? Please use **ideone.com** and share the link here.

10 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Alok Kumar • a month ago

Your non recursive binary search is good, but it's always O(logn). I mean you
you have just 2 elements. You can improve it, for average and good cases by
must agree solving it this way makes code much easier to understand.

Meenal • 4 months ago

I have a doubt in above code. When i run above code I am not getting values a
debugged and found out random number r is not generated with equal probabi
anyone help me out with this issue.
Code used is same as above.
Please help if I am missing anything.
I tried to run it for
arr[]: {10, 20, 30} freq[]: {2, 3, 1}
Prefix[]: {2, 5, 6}

and out put was not according to frequency.

spheroid • 11 months ago

Nice Article.Below is Implementation left as an exercise

```
 #include<stdio.h>
#include<iostream>
#include<stdlib.h>
#include<time.h>
using namespace std;

int main()
{
    int sum=0,k=0,opt=1;
    int arr[]={10, 30, 20, 40} ;
    int  freq[] = {1, 6, 2, 1} ;
    int l=sizeof(freq)/sizeof(int);
    for( k=0;k<l;k++)
    {
             sum+=freq[k];
    }
```

see more

spheroid ➦ spheroid • 11 months ago

One Correction

int x=rand()%(sum-1);
should be
int x=rand()%(sum);//x in range 0 to (sum-1)

```
/* Paste your code here (You may delete these lines if not wr
```

Prateek • 11 months ago

```
 /* Paste your code here (You may delete these lines if not writing co
int findCeil(int arr[], int r, int l, int h)
{
    int mid;
    while (l < h)
    {
         mid = l + ((h - l) >> 1);  // Same as mid = (l+h)/2
        (r > arr[mid]) ? (l = mid + 1) : (h = mid);
    }
    return (arr[l] >= r) ? l : -1;
}

if above function return -1 and later this value is used as array ind

int indexc = findCeil(prefix, r, 0, n - 1);
    return arr[indexc];

Does it sounds good??
```

Aashish ➦ Prateek • 11 months ago

Please take a closer look at the algorithm. The findCeil() will never retu

Thanks for inputs. We will be adding this point to the original post.

Zohreh Karimi • 11 months ago

Time Complexity can be improved by binary search.
Run time can be improved by generating the ceil array once and using it many
(Note: code might have some flaws.)

```
[sourcecode language="C"]
void Arr::weightedRandom(vector<int> a, vector<float> prob) {.
vector<float> ceil;
ceil.push_back(0);
for (int i=1; I < a.size(); i++) {.
ceil.push_back(prob[i]*100 + ceil[i-1]);.
}

vector<int> sum;
for (int i=0; I <= a.size(); i++) {.
sum.push_back(0);
}

for (int i=0; I < 10000; i++) {.
int rIndex = genRandomNo(ceil);.
```

see more

Mahesh Gs • 11 months ago

Nice one

GeeksforGeeks • 11 months ago

Please mail your experience to contribute@geeksforgeeks.org

Subscribe   Add Disqus to your site