Maths 1

Gyanu Mayank

17th February, 2023

Agenda

1. Problems Based on Randomization
2. Concept of Greatest Common Divisor

Problems

1. **Given rand6() implement rand9()**

Given a function rand6() that returns random numbers from 1 to 6 with equal probability, implement the one-liner function rand9() using rand6() that returns random numbers from 1 to 9 with equal probability.

Approach

Let's say that we are provided with the rand6() function that generates values uniformly from 1 to 6. We can generate values uniformly from

generate numbers in the range 0 to 36 using the following formula

(rand6() – 1) + 6 * (rand6())

Now, we want the range to be [1,2,...,9] so we will do v%9 then the range will be [0,1,...,8] if we add 1 to it [1,2,...,9] so we should return **v%9+1.**

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int rand6()
{

    return (rand() % 6) + 1;
}
int rand9()
{
    return ((rand6() + (rand6() - 1) * 6) %
}

int main()
{
    srand(time(NULL));
    int N = 9;
    int freq[N + 1] = {0};
    for (int i = 0; i < N; i++)
        freq[rand9()]++;
    for (int i = 1; i <= N; i++)
        cout << freq[i] << " ";
    return 0;
}
```

numbers from 1 to 5 with equal probability, implement the one-liner function rand7() using rand5() that returns random numbers from 1 to 5 with equal probability.

Approach

Let's say that we are provided with the rand5() function that generates values uniformly from 1 to 5. We can generate values uniformly from 0 to 4 by subtracting 1 from the result of rand5(). Now using rand5() we can uniformly generate numbers in the range 0 to 25 using the following formula

(rand5()) + 5 * (rand5() – 1)

Now to generate numbers uniformly from 1 to 7, we use rejection sampling. If the above formula generates a random number in the range of 0 to 22, we accept the number. If the formula generates a number in the range of 23 to 25, then we reject the number and re-try the formula till we get a number in the range of 0 to 22. We then find the remainder when the number in the range 0 to 22 is divided by 7. The remainder will be uniformly distributed in the range of 0 to 6. We then add 1 to the remainder to get the result which will be uniformly distributed in the range of 1-7. The rejection sampling in this case can be visualised with a 5 * 5 grid, where in if we throw a dart and the dart randomly lands on a cell, the number in the grid is accepted for some cells and rejected for others.

3	1	2	3	4	5
4	6	reject	reject	reject	reject

Implementation

```
#include <bits/stdc++.h>
using namespace std;
int rand5()
{

    return (rand() % 5) + 1;
}
int rand7()
{
    int x = (rand5() - 1) * 5 + rand5();
    if (x >= 22)
        return rand7();
    return x % 7 + 1;
}

int main()
{
    srand(time(NULL));
    int N = 7;

    int freq[N + 1] = {0};

    for (int i = 0; i < N; i++)
        freq[rand7()]++;

    for (int i = 1; i <= N; i++)
        cout << freq[i] << " ";

    return 0;
}
```

(Highest Common Factor) of two numbers is the largest number that divides both of them. For example, GCD of 20 and 28 is 4 and GCD of 98 and 56 is 14.
A simple and old approach is the **Euclidean algorithm subtraction** is a process of repeat subtraction, carrying the result forward each time until the result is equal to any one number being subtracted. If the answer is greater than 1, there is a GCD (besides 1). If the answer is 1, there is no common divisor (besides 1), and so both numbers are coprime.

Simple Solution :
For finding the GCD of two numbers we will first find the minimum of the two numbers and then find the highest common factor of that minimum which is also the factor of the other number.
Implementation

```
#include <bits/stdc++.h>
using namespace std;

int gcd(int a, int b)
{
    int result = min(a, b);
    while (result > 0)
    {
        if (a % result == 0 && b % result =
        {
            break;
        }
        result--;
```

```
}

int main()
{
    int a, b;
    cin >> a >> b;
    cout << gcd(a, b);
    return 0;
}
```

Optimized Approach

An **efficient solution** is to use the **Euclidean algorithm** which is the main algorithm used for this purpose. The idea is, the GCD of two numbers doesn't change if a smaller number is subtracted from a bigger number.

```
#include <bits/stdc++.h>
using namespace std;

int gcd(int a, int b)
{
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int main()
{
    int a, b;
    cin >> a >> b;
    cout << gcd(a, b);
    return 0;
}
```

4. GCD of more than two numbers

Given an array of numbers, find the GCD of the array elements

Example:

Input: n=3, arr[] = {1, 2, 3}

Output:1

The GCD of three or more numbers equals the product of the prime factors common to all the numbers, but it can also be calculated by repeatedly taking the GCDs of pairs of numbers.

gcd(a, b, c) = gcd(a, gcd(b, c)) = gcd(gcd(a, b), c) = gcd(gcd(a, c), b)

Implementation

```
#include <bits/stdc++.h>
using namespace std;

int gcd(int a, int b)
{
    if (a == 0)
        return b;
    return gcd(b % a, a);
}

int findGCD(int arr[], int n)
{
    int result = arr[0];
    for (int i = 1; i < n; i++)
    {
        result = gcd(arr[i], result);
```

```
            return i;
        }
    }
    return result;
}

int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    cout << findGCD(arr, n) << endl;
    return 0;
}
```

5. **Subsequence with gcd 1**

Given an array find if there is a subsequence with gcd 1 if it is not there print -1 else print 1

Approach

If there is any subsequence with gcd 1 then the gcd of the whole array will be 1 so find the gcd of the whole array if it is 1 then return 1 else return -1

Implementation

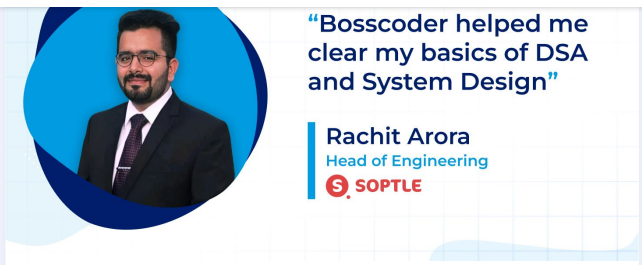
```
#include <bits/stdc++.h>
using namespace std;

int gcd(int a, int b)
{
    if (a == 0)
        return b;
```

```
int findGCD(int arr[], int n)
{
    int result = arr[0];
    for (int i = 1; i < n; i++)
    {
        result = gcd(arr[i], result);

        if (result == 1)
        {
            return 1;
        }
    }
    return result;
}

int main()
{
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
        cin >> arr[i];
    if (findGCD(arr, n) == 1)
        cout << 1;
    else
        cout << -1;
    return 0;
}
```

Bosscoder had everything I was looking for

Read

Date: 12th May, 2023



Bosscoder helped me clear my basics of DSA and System Design

Read

Date: 5th May, 2023

View All blogs