Design and Analysis of Algorithm: Lab2

Prasanna Natarajan

1410110298

Problem Statement:

Problem Statement: How to simulate an n-sided coin using a 2-sided coin. (Solve for n=6).

Algorithm:

Solution: You can simulate an n-sided coin using a two-sided coin as follows:

Let m = $\lceil \log n \rceil$. The base is always 2. (Example, for n = 6, m = 3)

Flip a 2-sided coin m times and record the result of every flip. (HHT may be represented as 110)

Convert the binary number generated to a decimal number. (Example: $(110)_2 = (6)_{10}$)

Repeat for the number of sample points required.

Challenge:

If m =3, the numbers generated will be in the range 0 to 7, whereas we need the numbers in the range (1, 6).

A possible Solution-

- When you get a number not in range, ignore it and regenerate another number in range.
  In this example – When you generate a 0 or a 7, ignore it and generate another number till you get a number in the range and record that.

Note: When n = 6, we can simulate a dice using a 2-sided coin.

C Code

```
/*
Name        :   Prasanna Natarajan
Roll Number :   1410110298
Inputs      :   number of sides of coin
Outputs     :   Distribution of occurences in the number of sample
points

*/

#include<stdio.h>
#include<math.h>
#include<time.h>
// declaring functions
int toDecimal(int a[], int n); // a is a binary array, n is the size
of the array

int main(){
    int *a;
    int n; // number of sides
    int i = 0;
    time_t t;
    srand((unsigned)time(&t));
    printf("Enter the number of sides: ");
    scanf("%d",&n);
    int num_sample_points = 1000;
    int j=0;
    int m;
    int counter[] = {0,0,0,0,0,0};
    int dec=0;
    for(j=0;j<num_sample_points;j++){
        m = ceil(log2(n));
        //printf("m = %d\n",m);
        a = (int *)malloc( sizeof(int)*m );
        for(i=0;i<m;i++){
            a[i] = rand()%2;
            //printf("a[%d] = %d\n",i,a[i]);
        }
        dec = toDecimal(a,m);
        //printf("dec = %d\n",dec);
        if(dec<7 && dec>0){
            //printf("in the range");
            counter[dec-1]++;
        }

        else j--;
    }

    for(i=0;i<6;i++){
        printf("\ncounter[%d] = %d\n",i+1,counter[i]);
    }
    return 0;
}

int toDecimal(int a[], int n){
    int ret = 0;
    int i=0;
    for(i=1;i<=n;i++){
        //printf("This is a[n-i] = %d n= %d\n",a[n-i],n);
        ret += (a[n-i]*pow(2,i-1));
    }
    return ret;

}
```

Result Table

Sample space is the following: {1, 2, 3, 4, 5, 6}
Find the probability of each event, while generating 1000 samples points.

Event	Probability of event
1	0.175
2	0.17
3	0.161
4	0.173
5	0.153
6	0.168

Analysis

Did the result meet the expectation?
The expectation of the result is to see the probability to be exactly 0.166. The observed values are in the acceptable range of error. As seen in the above table, all the values are close to 0.166.

If no, can you think of an improvement?
To improve the performance, we could try to distribute the probability of 0 and 7 to first half and second half of the six numbers.