DAA – Lab 2
Jan 18, 2017

Vishal Gauba
1410110501

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.

Source Code

```
#include "stdio.h"
#include "math.h"
#include <time.h>

int main(int argc, char const *argv[])
{
	/* declaration of variables */
	int arr[3]={0}, numarr[6]={0}, size=3, i, j, sample=100000;

	/* generate n sample values */
	for (j = 0; j < sample; ++j)
	{
		int num=0;

		/* flipping a 2-sided coin 3 times*/
		for (i = 0; i < size; ++i){
			arr[i] = rand()%2;
		}

		/* converting binary number to decimal*/
		for (i = 0; i < size; ++i){
			num+=arr[i]*pow(2,i);
		}

		/* discarding if 0 or 7 */
		if(num == 0 || num == 7){
			j--;
		} else { /* updating count of the number generated by simulated toss */
			numarr[num-1] += 1;
		}
	}

	/* calculating probabilities of each number 0 to 6 */
	for (i = 0; i < 6; ++i)
	{
		printf("%lf\n", (float) numarr[i]/sample);
	}

	return 0;
}
```

Result Table

Sample space is the following: {1, 2, 3, 4, 5, 6}

Probability of each event, while generating 1000 samples points:

Event	Probability of event
1	**0.171000**
2	**0.175000**
3	**0.184000**
4	**0.163000**
5	**0.156000**
6	**0.151000**

Analysis

Did the result meet the expectation?

- **Not quite. The expected ideal probability was 1/6 = 0.166. While the results are close to this value, for a real world pseudo-random generator a more distributed probability is required.**

If no, can you think of an improvement?

- **Increasing the sample size tenfold (10000) gives much better probabilities. Another improvement could be using time as a seed every time the random 0 to 1 is generated (2-sided coin is tossed).**