## Simulating Observations

- Simulating observations of a discrete random variable with a prescribed set of probabilities. We denote by $f_x(x) = P(X=x)$.
- C.D.F = cumulative distribution function. For a real-valued random variable X, $F_X(x) = P(X \leq x)$.
- The probability that X is in the interval [a,b] is $F_X(b)-F_X(a)$.
- We can use the CDF to generate values for the variable X with the given distribution of probabilities.



## General Idea

- We can pre-compute the values of the CDF and store them in an array. These values are between 0 and 1, and in ascending order.
- Store the possible values of X in a different array at corresponding positions.
- To generate a value for X, we generate a random number between 0 and 1, u.
- Find the smallest value in the CDF array that is greater or equal than u.
- Return the value of X at that position in the second array.

## Example

- Suppose that we want to generate the sum of two independent fair dice.
- We can do the following:

```
int sum_of_faces()
{
    int r = randint(1,6);
    int s = randint(1,6);
    return  r + s;
}
```



## CDF

- The value of the sum can be from 2 to 12.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| x | | | | | | | | | | |
| $f_x(x)$ | | | | | | | | | | |
| $F_x(x)$ | | | | | | | | | | |

```
int sum_of_faces()
{
  static bool called_previously = false;
  static double  cdf[11];
  static int     X_value[11];
  if (!called_previously)  {
    called_previously = true;
    compute_cdf(cdf, X);
  }
  double u = randouble();
  int i = 0;
  while (u > cdf[i])
    ++i;
  return X_value[i];
}
```



```
void compute_cdf(float cdf[], float X[])
{
  int i;
  cdf[0]     = 1.0/36.0;
  X_value[0] = 2;
  for (i = 1; i <= 5; ++i)  {
    cdf[i] = cdf[i-1] + (i+1)/36.0;
    X_value[i] = i + 2;
  }
  for (i = 6; i <= 9; ++i) {
    cdf[i] = cdf[i-1] + (11-i)/36.0;
    X_value[i] = i + 2;
  }
  cdf[10] = 1.0;
  X_value[10] = 12;
}
```

```
/* Compute the cdf array knowing
  the simple distribution of
  probabilities f. */

void compute_cdf(float cdf[],
  float f[], int n)
{
  float sum = 0;
  for (int x=0; x<n; x++) {
    sum += f[x];
    cdf[x] = sum;
  }
}
```



```
/* Simulate an outcome from 0 to
  n-1 given the f array. */

void simulate(float f[], int n)
{
  float sum = f[0];
  u = randouble(); // in [0,1)
  x = 0;
  while (sum < u && x < n) {
    x++;
    sum += f[x];
  }
  return x;
}
```

## Infinite Values

- Simulate the number of times that it takes for a fair coin to come up heads.
- There is no limit to the number of attempts to have the coin come up face – X can have an infinite number of values.
- The probability decreases as the number of attempts increases.
- We have to choose a limit for the variable, high enough that the probability to have a higher value than that is very small. The function will only generate values within that range.



## CDF

- X = {1, 2, 3, …}
- $f_x(1) = ½$ $F_x(1) = ½$
- $f_x(2) = ¼$ $F_x(2) = ½+¼ = 3/4$
- $f_x(3) = 1/8$ $F_x(2) = ½+¼+1/8=7/8$
- $f_x(k) = 1/2^k$ …
- $F_x(k) = \sum^k_{i=1} 1/2^i = (2^k-1)/2^k$
- For k=30, the probability that the number of attempts is bigger than that is less than 1 over a billion.