Login: student

Password: prawdo22

Probability and statistics, 2021, Computer Science Algorithmics,
Undergraduate Course, Part II, lecturer: Mirosław Kutyłowski

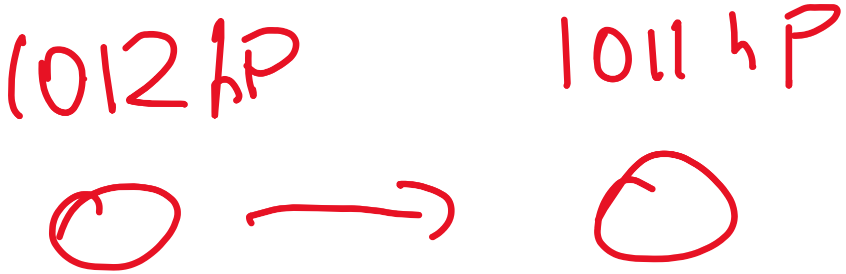
I. Generating Random Numbers for a given probability distribution

Chapter: 5.2 from Byron

goal:
- simulations (e.g. pharma industry, weather forcast, system testing ...)

Weather simulations:

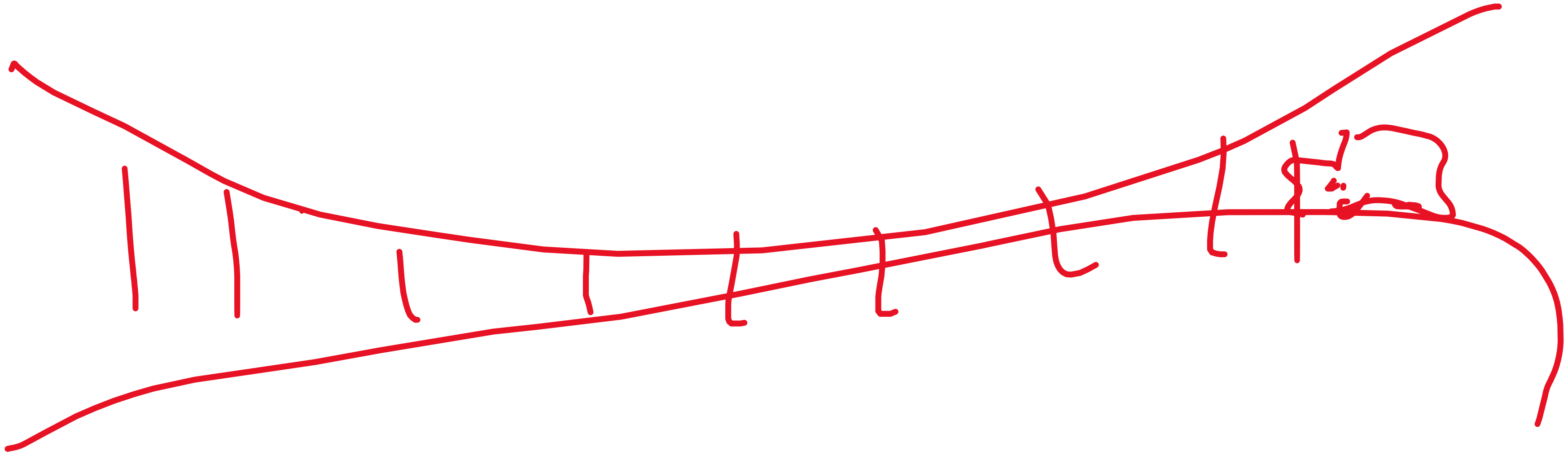
stara metoda: modelowanie



Model: fluktuacje

Złożony losowy proces

Modelling behavior of a bridge:



The traffic on the bridge is a random proces: we need to generate these random events

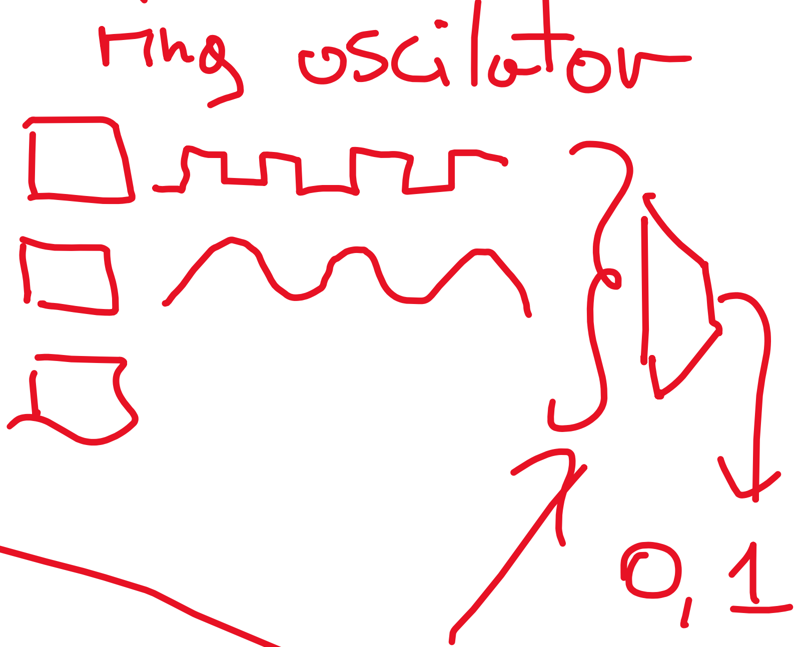
Simulations for new chemical products, pharmaceuticals:

projekt. leków

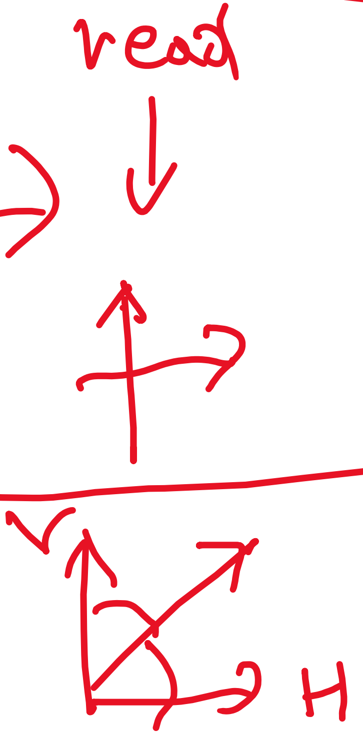
złożoność obliczeniowa

Physical sources: examples:

1) Electronics (bistable)



3) quantum generators



3) noise

Problems of physical sources:

1) bias – obciążony

60% to 0, 40% bitów 1 | #0 = #1

2) memory: dependence on history

ograniczenia na częstości próbkowania

3) external influence

generowanie losowego hasła

output przewidywalny

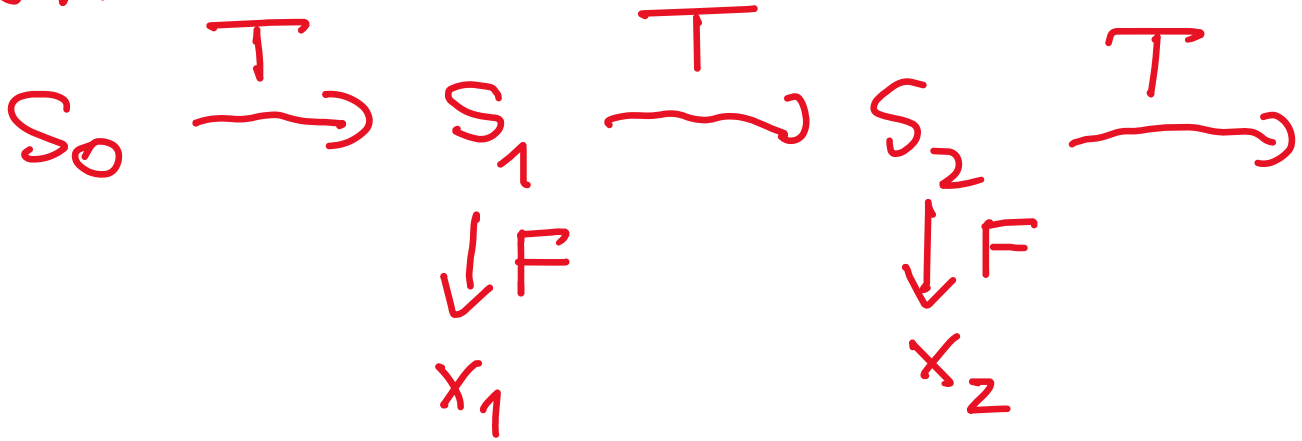
deterministic random number generators:

DRNG: **NIST, recommendations,**

rozkład jednostajny, ciąg bitów „niezależnych"

architecture: PRNG(seed) yields: bits

stan:



Idealny:

licznik

DRNG:

basic property: not distinguishable from coin flipping

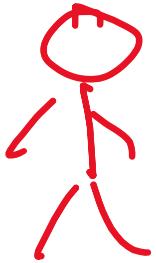
what does it mean?:

NIST tests

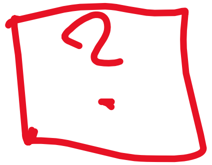
kompromis!

szybkie i w miarę skuteczne

left-or-right game

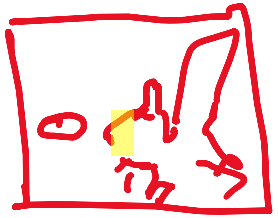


generator



A: DRNG

B:



w środku dinozaur rzuca monetą

secure PRNG: unpredictability

forwards:

znając $x_1, \ldots, x_n$ nie mogę zgadnąć

$x_{n+1}, x_{n+2}, \ldots$

backwards:

szyfrowanie rozmów telefonicznych:

podsłuchana rozmowa: $M \text{ XOR } S_1$

później chwila milczenia: $0\ldots0 \text{ XOR } S_2$

atakujący: zna S'

wyprowadza na podstawie S' ciąg S

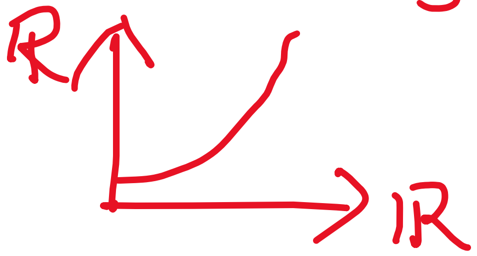
odczytuje M ← tak nie powinno być!

realizations:

families of PRNG (based on residual arithmetic and algebraic expressions

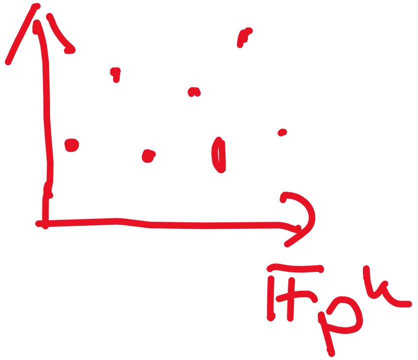
$$a \cdot x^2 + b \cdot x + c \bmod p$$

a, b - parametry



cryptographic generators: e.g. based on encryption

$$trunc(32, Enc_K(1)),\ trunc(32, Enc_K(2)),\ \ldots$$

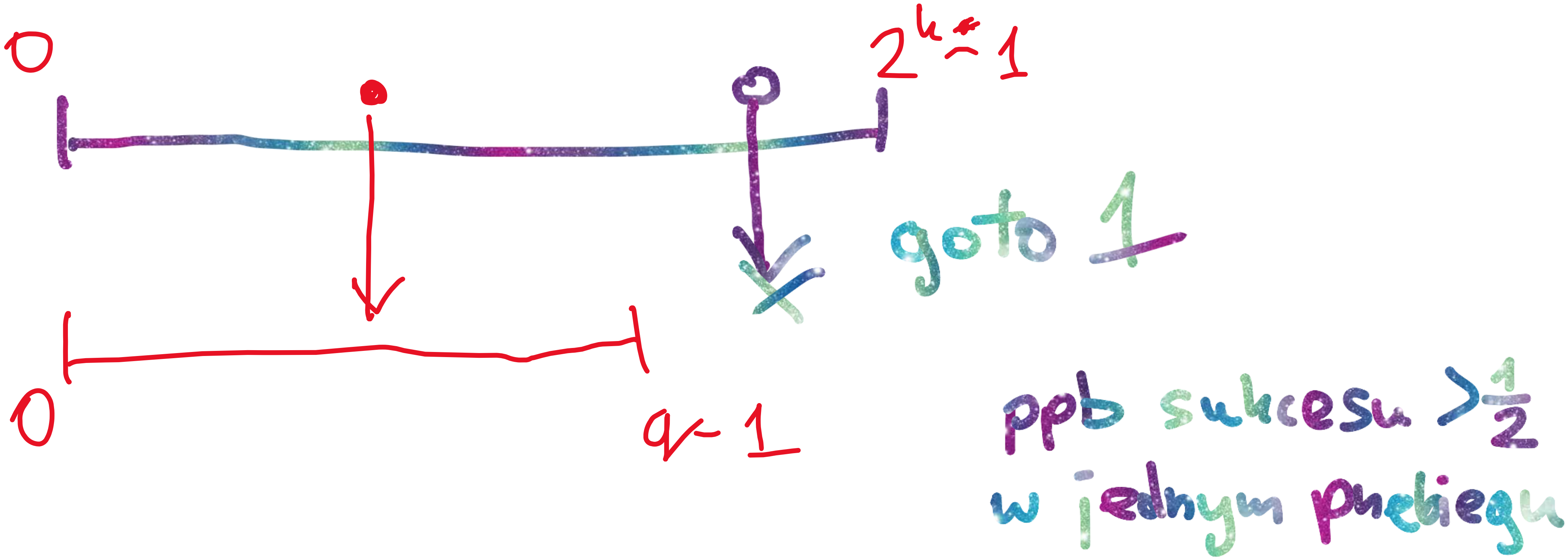


bardzo efektywne gdy mamy np.
hardware'owe wsparcie dla szyfrowania
np.: AES

Problem: domain

We have a good generator for uniform distribution over n bit numbers

How to get a uniform distribution over integers in the range [*0,q*)?



Problem: uniform versus non-uniform distribution

all good PRNG resources deliver the output that is uniform over some interval
e.g.: 32 bit nonnegative integers

Needed:
e.g. geometric distribution, Poisson, ...

single Bernoulli trial:

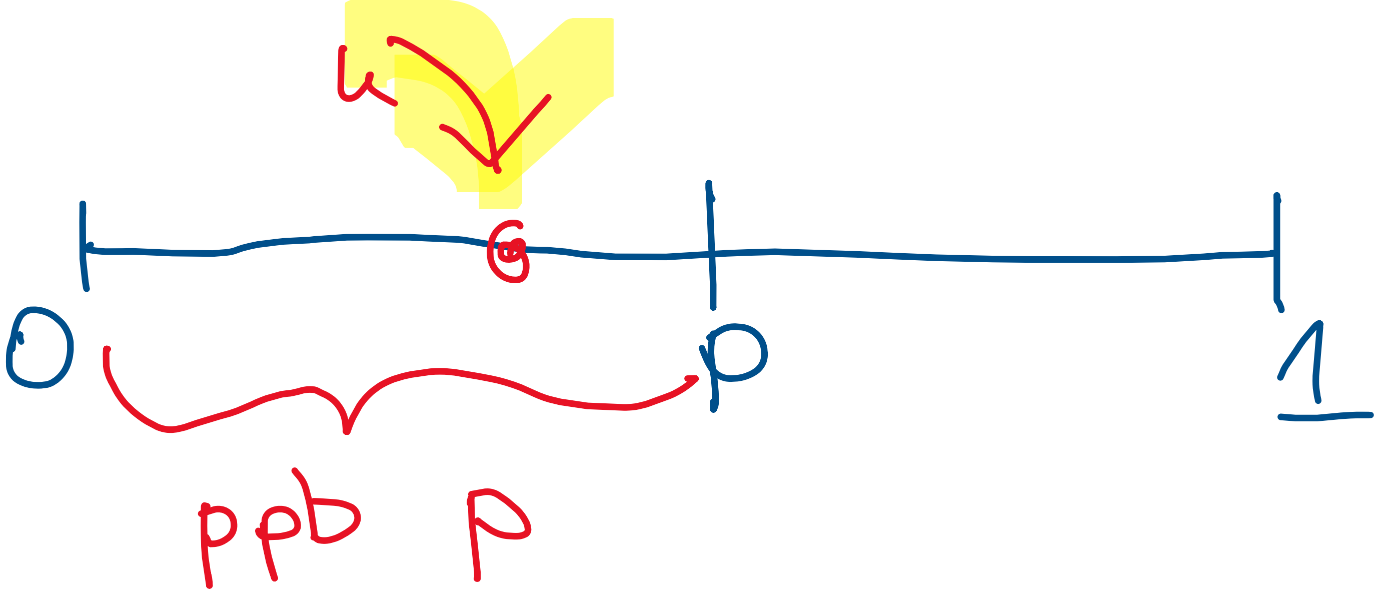
1 bit:

0 = ppb p

1 = ppb 1−p

procedure:

1. choose u uniformly at random in [0,1]
2. if $u<p$ then output 0 else output 1



n Bernoulli trials, number of successes:

n **times:**

- **0: with probability *p***
- **1: with probability 1-*p***

count the number of successes

stupid solution:

compute pbb according to formulas

...

n Bernoulli trials, number of successes:

n times:

0: with probability p

1: with probability 1-p

count the number of successes

procedure (in MATLAB):

```
n=20; p=0.34;
U=rand(n,1);
X=sum(U<p)
```

Matlab works on matrices. the function rand(n,1) generates a matrix n x 1 with random elements from [0,1)

↑ Summing over the whole matrix

geometric distribution:

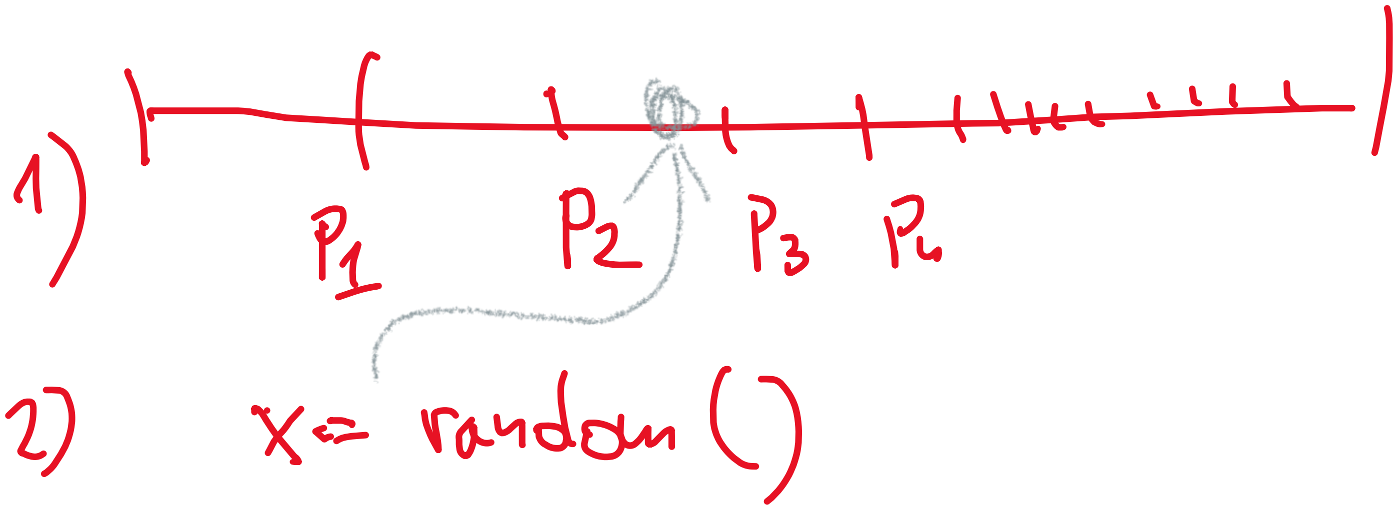
Bernoulli trials with pbb *p* of 0
output: the number of trials until 1 chosen

naive way: take mathematical formulas and then choose according to the probabilities

Naiwnie :

procedure (in MATLAB):

```
X=1;
while rand<p
X=X+1;
end;
X
```



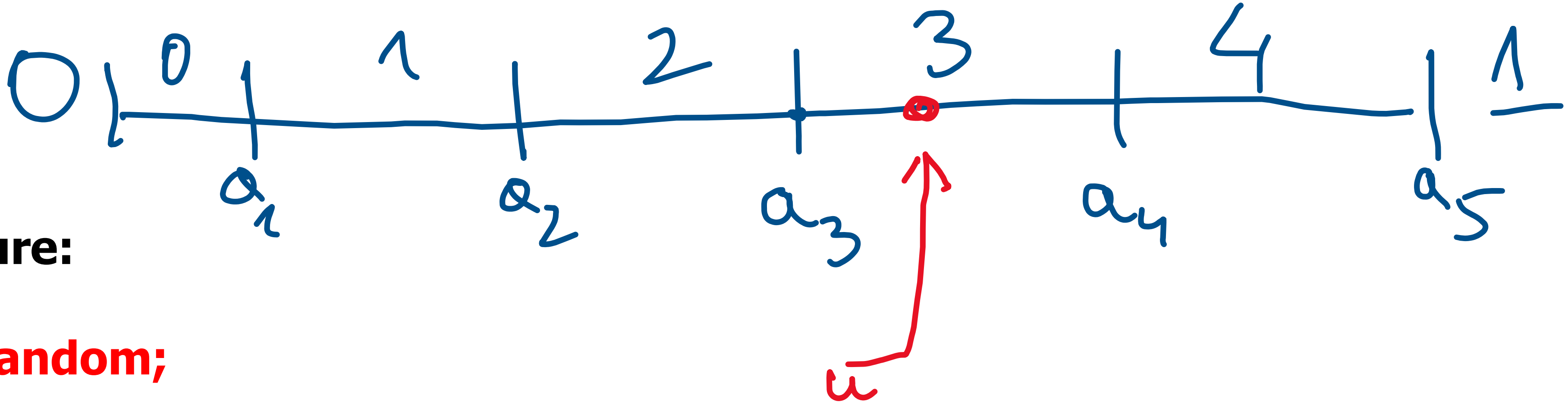
arbitrary discrete distribution:

assume: ***n*** **possible values,** ***p(i)*** **-probability of the ith value**

approach: for each $i \leq n$ **compute**

$$a_i = \sum_{j<i} p_j$$

the results saved in a data structure ***D***



procedure:

1. **u=random;**
2. **with D find i such that** $a_i \leq u < a_{i+1}$

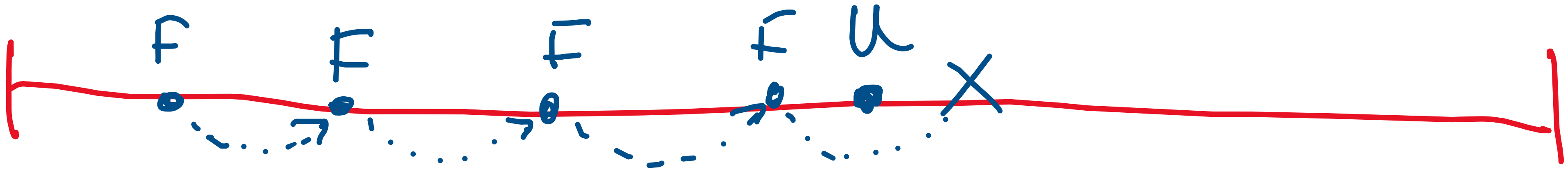
a_3 = ppb(0) + ppb(1) + ppb(2)

Poisson distribution:

$$Pr(X=i)=e^{-\lambda}\cdot\frac{\lambda^i}{i!}$$

Matlab program from Byron:

```
lambda  =  5;                          % Parameter
U       =  rand;                       % Generated Uniform variable
i       =  0;                          % Initial value
F       =  exp(-lambda);               % Initial value, F(0)
while (U >= F);                        % The loop ends when U < F(i)
    F = F + exp(-lambda) * lambda^i/gamma(i+1);
    i = i + 1;
end;
X=i
```



tylko tyle iteracji ile trzeba

the case of invertible CDF

Theorem 2 *Let X be a continuous random variable with cdf $F_X(x)$. Define a random variable $U = F_X(X)$. The distribution of U is Uniform(0,1).*

Proof: First, we notice that $0 \leq F(x) \leq 1$ for all x, therefore, values of U lie in $[0,1]$. Second, for any $u \in [0,1]$, find the cdf of U,

$$\begin{aligned}
F_U(u) &= \boldsymbol{P}\{U \leq u\} \\
&= \boldsymbol{P}\{F_X(X) \leq u\} \\
&= \boldsymbol{P}\{X \leq F_X^{-1}(u)\} && \text{(solve the inequality for } X\text{)} \\
&= F_X(F_X^{-1}(u)) && \text{(by definition of cdf)} \\
&= u && (F_X \text{ and } F_X^{-1} \text{ cancel})
\end{aligned}$$