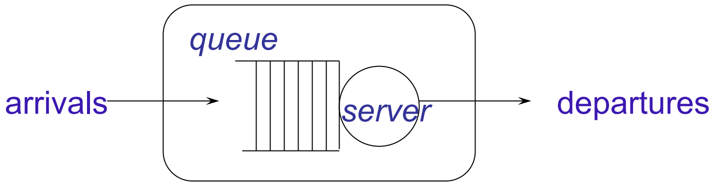
caso studio:
machine shop model

Discrete-event simulation
Trace-driven simulation

Single server queue



Consider *n*=10 job and given arrival and service times:

Arrival times:

15 47 71 111 123 152 166 226 310 320

Service times:

43 36 34 30 38 40 31 29 36 30

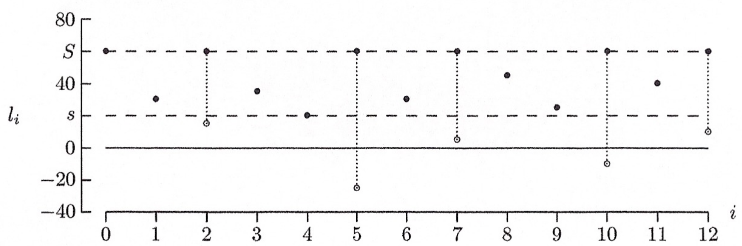
1

case study 2

Discrete-event simulation
Trace-driven simulation

A simple inventory system

i	:	1	2	3	4	5	6	7	8	9	10	11	12
d_i	:	30	15	25	15	45	30	25	15	20	35	20	30



2

- ssq1 and sis1 require input data from an outside source
- The usefulness of these programs is limited by amount of available data:
  - What if more data needed? Se ci servono più dati?
  - What if the model changed? Se modello cambia?
  - What if the input data set is small or unavailable? Se ho pochi dati?

↓

Random number generator

- It produces real values between 0.0 and 1.0 (Uniforme)
- The output can be converted to random variate via mathematical transformations (convertito in distr. di probabilità)

Performance Modeling of Computer Systems and Networks

Prof. Vittoria de Nitto Personè

Random Number Generators

Università degli studi di Roma Tor Vergata
Department of Civil Engineering and Computer Science Engineering

Historically there are three types of generators

- table look-up generators (1950)
- hardware generators
- algorithmic (software) generators

generatori testabili e controllabili, ma non tutti sono validi! ("C" no!)

Algorithmic generators are widely accepted because they meet all of the following criteria:

- *randomness* - output passes all reasonable statistical tests of randomness
- *controllability* - able to reproduce output, if desired
- *portability* - able to produce the same output on a wide variety of computer systems
- *efficiency* - fast, minimal computer resource requirements
- *documentation* - theoretically analyzed and extensively tested

Algorithmic Generators

- An *ideal* random number generator produces output such that *each* value in the interval $0.0 < u < 1.0$ is *equally likely* to occur
  (numeri $R \in [0,1]$ sono ∞)
- A *good* random number generator produces output that is (almost) statistically indistinguishable from an ideal generator

Conceptual Model

m numeri interi in un'urna

- Choose a *large* positive integer m>0. This defines the set

  $\chi_m = \{1, 2, \ldots m-1\}$
- Fill a (conceptual) urn with the elements of χ_m
- Each time a random number u is needed, draw an integer x "at random" from the urn and let $u = x/m \in [0,1]$

- Each draw *simulates* a sample of an independent identically distributed sequence of *Uniform*(0, 1)
- The possible values are 1/*m*, 2/*m*, … (*m*-1)/*m*
- It is important that *m* be large so that the possible values are $\underbrace{\text{densely distribute}}_{\text{insieme denso se `m' grande}}$d between 0.0 and 1.0

Conceptual Model

- 0.0 and 1.0 are impossible

  This is important for some random variates
- the same probability for each draw→ replacement of the drawn element
- for practical reasons, we will draw without replacement

  If *m* is large and the number of draws is small relative to *m*, then the distinction is largely irrelevant

Se estraggo '5', nella nuova estrazione lo rimetto. A livello software non posso "rimetterlo dentro"

Lehmer Generator

- is defined in terms of two fixed parameters:
  - *modulus m*, a fixed large prime integer
  - *multiplier a*, a fixed integer in χ_m
- the possible values are 1/*m*, 2/*m*, … (*m*-1)/*m*

The integer sequence x_0, x_1, … is defined by the iterative equation

$$x_{i+1} = g(x_i)$$ genera numero a partire dal precedente!

with

$$g(x) = \underbrace{a}_{\text{moltiplicatore}}\underbrace{x}_{\text{precedente estrazione}} \bmod m$$

$x_0 \in \chi_m$ is called *initial seed*

- Because of the mod operator, $0 \le g(x) < m$
- 0 must not occur
  - since *m* is prime, $g(x) \neq 0$ if $x \in \chi_m$
  - if $x_0 \in \chi_m$, then $x_i \in \chi_m$ for all $i \geq 0$
- IF the multiplier and prime modulus are chosen properly, a Lehmer generator is statistically indistinguishable from drawing from χ_m with replacement
- NOTE, there is nothing random about a Lehmer generator

→ pseudo-random generator

esempio: $m = 7$, urna $\mathcal{X}_7 = \{1,2,3,4,5,\overset{n-1}{6}\}$. Scelgo $a = 3$, $x_0 = 1$ seme. Genero, partendo da $g(x) = ax \bmod m$, i seguenti valori:
$g(x_0) = 1 \cdot 3 \bmod 7 = 3 = x_1$; $g(x_1) = 3 \cdot 3 \bmod 7 = 2 = x_2$; $g(x_2) = 3 \cdot 2 \bmod 7 = 6 = x_3$; $g(x_3) = 3 \cdot 6 \bmod 7 = 4 = x_4$ …. cioè: 1 3 2 6 4 5 1, generati tutti

- Se $x_0 = 3$? (DEVE appartenere a $\mathcal{X}$) genero: 3 2 6 4 5 1 3, cambia l'inizio, e basta! Quindi x_0 è irrilevante
- Se $a = 2$, $x_0 = 1$? genero 1 2 4 1, non sono tutti! Rilevante! $a = 3$ è moltiplicatore FULL Period (ottimale)
- Se $a = 2$, $x_0 = 3$? genero 3 6 5 3, sottosequenza! 'a' ed 'm' SONO IMPORTANTI

Parameter Considerations

- the choice of *m* is dictated, in part, by system considerations
  - on a system with 32-bit 2's complement integer arithmetic, $\underbrace{2^{31}-1}_{=m}$ is a natural choice (it is prime!)
  - with 16-bit or 64-bit integer representation, the choice is not obvious (the maxes are not prime)
  - in general, we want to choose *m* to be the largest representable prime integer
- Given *m*, the choice of *a* must be made with great care
  ↳ influisce sul periodo pieno

- For a chosen (a, *m*) pair, does the function g(·) generate a **full-period** sequence?
- If a full period sequence is generated, how random does the sequence appear to be?
- Can *ax* mod *m* be evaluated efficiently and correctly?
  - Integer overflow can occur when computing *ax*

Full Period Multipliers

da dove parto

- If we pick any initial seed $x_0 \in \chi_m$ and generate the sequence x_0, x_1, x_2, … then x_0 will occur again (x_0 ricompare dopo un po' di generazioni)

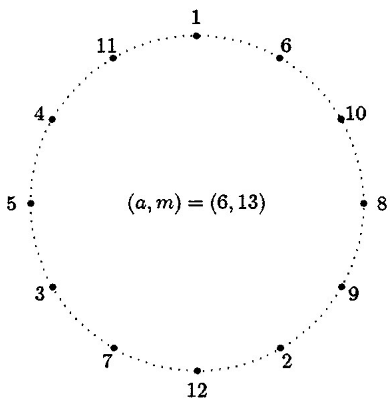
- Further x_0 will reappear at index p that is either $m - 1$ or a divisor of $m - 1$

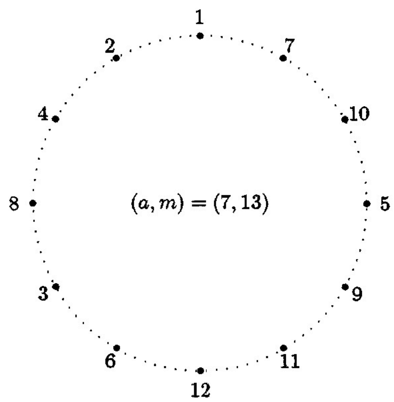
$$p = \begin{cases} m-1 & \text{full period} \leftarrow \text{NOSTRO INTERESSE} \\ \text{divisore di } m-1 \end{cases}$$

We are interested in choosing full-period (FP) multipliers where p = m-1

13

Full-period multipliers generate a virtual circular list with m-1 distinct elements.

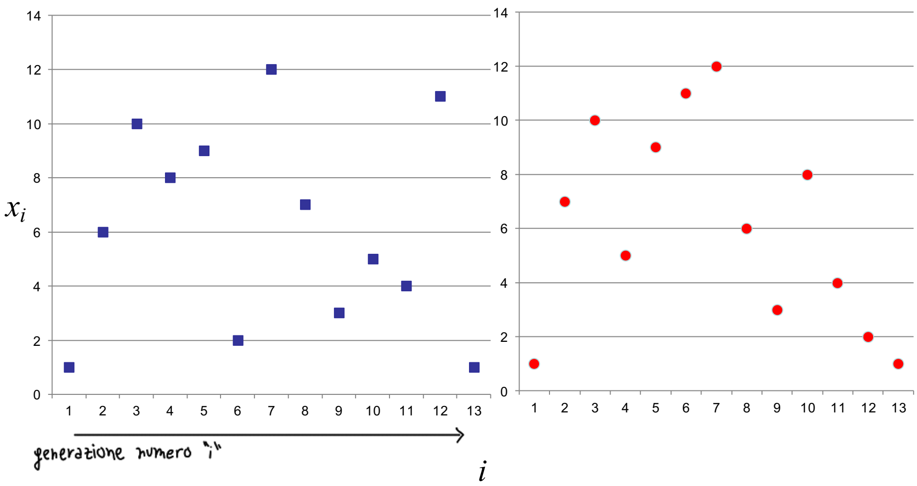




14

m=13

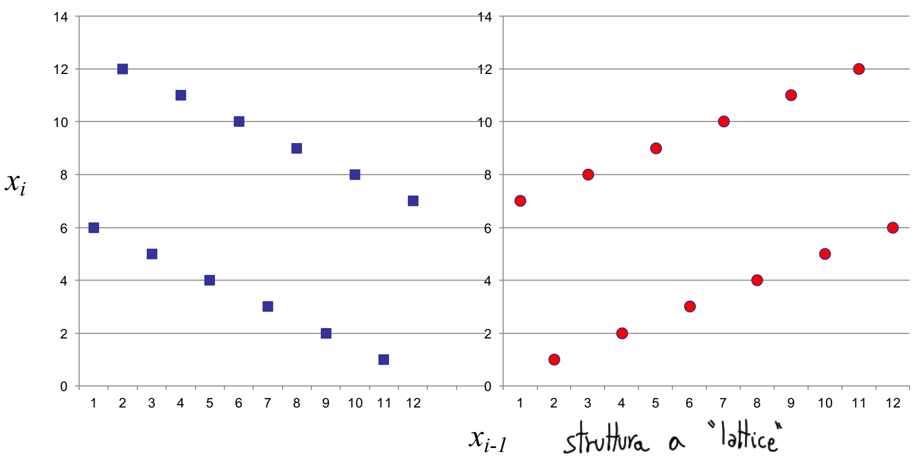
a=6, x_0=1 sembrano casuali a=7, x_0=1



15

m=13

a=6, x_0=1 qui non sembra random a=7, x_0=1



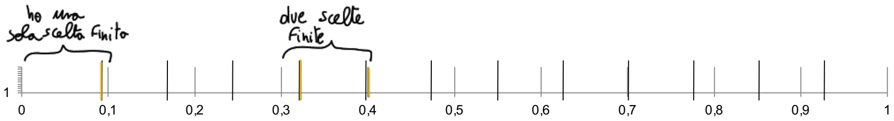
16

il grafico si legge cosi:
se m = 13 ho set {1,2,3,….,12}, a = 6, Xo = 1
computo: g(Xi-1) = a * Xi-1 = Xi
g(X0) = 6 *1 mod 13 = 6 = X1 (infatti X1 creato a partire da X0, che è seed = 1)
g(X1) = 6*6 mod 13 = 10 = X2 (X2 creato a partire da X1 = 6)
g(X2) = 6*10 mod 13 = 8 = X3

Ora, per ogni g(x) creo una coppia (Xi-1,Xi), qui nell esempio ho: (1,6), (6,10), (10,8). Li metto nel grafico ed ottengo tale struttura a lattice.

15/03/22

da "0" a "1" (ho diviso per 13) sono tutte a Lattice



a=6	0,076923077	0,461538462	0,769230769	0,615384615	0,692307692	0,153846154
a=7	0,923076923	0,538461538	0,230769231	0,384615385	0,307692308	0,846153846

0,076923077	0,538461538	0,769230769	0,384615385	0,692307692	0,846153846
0,923076923	0,461538462	0,230769231	0,615384615	0,307692308	0,153846154

12 **6** **3** | **8** **4** **2**

dimezzano → dimezzano →