Extended Introduction to Computer Science
CS1001.py

Chapter A Randomness in Computing
Lecture 4b

Amir Rubinstein, Michal Kleinbort

School of Computer Science
Tel-Aviv University
Fall Semester 2023-24
http://tau-cs1001-py.wikidot.com

* Slides based on a course designed by Prof. Benny Chor

Lecture Plan

- Intro – what is randomness and basic notions
- `import random`
- Random walk
- PageRank

Randomness in Computing

- What is randomness? According to Wikipedia:

  Randomness is the lack of pattern or predictability in events.

  A random sequence of events, symbols or steps has no order and does not follow an intelligible pattern or combination.

  Individual random events are by definition unpredictable, but in many cases the frequency of different outcomes over a large number of events (or "trials") is predictable.

- The programs we have written so far were deterministic: given the same input, their execution path will be identical.

- In many cases it is useful to include randomness in computations. Such algorithms are termed randomized or probabilistic or coin flipping algorithms. Practically, their execution path cannot be reproduced.

Obtaining Random Sequences: a Piece of History

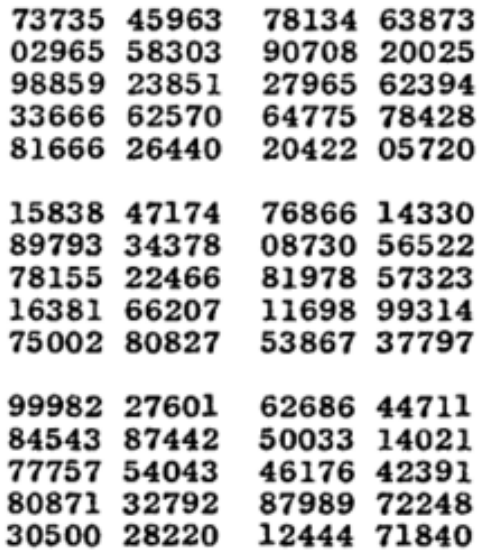
From Wikipedia:

A Million Random Digits with 100,000 Normal Deviates is a random number book by the RAND Corporation, originally published in 1955.

The book, consisting primarily of a random number table, was an important 20th century work in the field of statistics and random numbers.

It was produced starting in 1947 by an electronic simulation of a roulette wheel attached to a computer, the results of which were then carefully filtered and tested before being used to generate the table.

The RAND table was an important breakthrough in delivering random numbers, because such a large and carefully prepared table had never before been available. In addition to being available in book form, one could also order the digits on a series of punched cards.



73735	45963	78134	63873
02965	58303	90708	20025
98859	23851	27965	62394
33666	62570	64775	78428
81666	26440	20422	05720
15838	47174	76866	14330
89793	34378	08730	56522
78155	22466	81978	57323
16381	66207	11698	99314
75002	80827	53867	37797
99982	27601	62686	44711
84543	87442	50033	14021
77757	54043	46176	42391
80871	32792	87989	72248
30500	28220	12444	71840

Obtaining Random Sequences (cont.)

- True Random Number Generators (TRNG):
  A device that extract randomness from physical phenomena such as cosmic radiation, radioactive decay, etc.

  - A philosophical / meta-physical question for you to ponder about:
    Are there any events in nature that are truly random?

  - Recall Einstein's quote: "God does not play dice with the universe."
    referring to his views about quantum mechanics
    (which is often misinterpreted, see here)

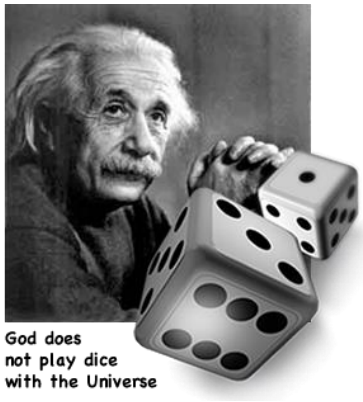


Image from:
http://www.smallplanet.us

- Pseudo-random Number Generators (PRNG):
  An algorithm that generates a long sequence that appears random ("random enough") w.r.t different tests. The sequence is generated deterministically, from an initial random value called "seed", which can be picked e.g. at startup by using the system's clock.

Example for PRNG – "Linear Congruential Generator"

$$x_{i+1} = (a \cdot x_i + c) \bmod m$$

$a = 1, c = 7, m = 12$

$x_0 = 0$ — The "seed"

$x_1 = 7$

$x_2 = 2$

$x_3 = 9$

…

```
>>> a, c, m = 1, 7, 12
>>> x = 0 # x0
>>> for i in range(20):
        print(x, end=" ") # end each print with a space
        x = (a*x + c) % m

0 7 2 9 4 11 6 1 8 3 10 5 0 7 2 9 4 11 6 1 8
```

- The numbers in the sequence must eventually enter a cycle. The length of the cycle is called the period of the PRNG.
- The choice of the parameters affects the period.
  Try, for example, $c = 8$ instead of 7.

Randomness in Python

- Luckily, Python employs a more sophisticated pseudo random number generator, called a Mersenne Twister, with a period of size $2^{19937} - 1$ (will not be further discussed).

```
>>> import random

>>> random.random() #generate a random number ∈ [0,1)
0.9724062711684623
>>> random.random()
0.9793789492766168
>>> random.random()
0.2880152915931866
```

Randomness in Python – Some Useful Functions

```
>>> random.uniform(3.2, 12.01) # random number ∈[3.2, 12.01)
9.311113665186017


>>> random.randrange(0,1000,10) # random int from range(start, stop, step)
920


>>> random.choice([1,5,6,-44,9]) # random element from the input collection
-44
>>> lst = [random.choice("a"*5 + "bcdef") for i in range(1000)]
>>> lst[:10]
['c', 'a', 'a', 'a', 'a', 'e', 'b', 'a', 'a', 'a']
>>> lst.count("a")/len(lst)
0.497
>>> lst.count("b")/len(lst)
0.101


>>> L = [1,2,3,4,5]
>>> random.shuffle(L) #in place permutation
>>> L
[2, 5, 1, 4, 3]
```

Does Randomness Have any Uses in Computing?

Use	Example from this course
Simulation	Random walk
Sampling	Estimating π*
Cryptography	Diffie-Hellman secret sharing**
Performance (efficiency)	Fermat's primality testing Random QuickSort

Next example

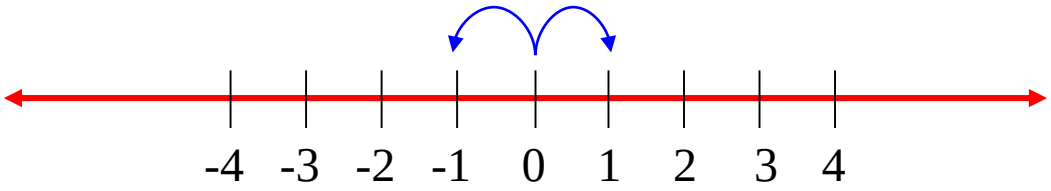
Will see these later in the course

* also program testing, polls,…

** and almost any cryptographic application: encryption, authentication,…

Random Walk

- A path that consists of a succession of random steps.
- Simple unbiased 1-Dimensional random walk
  - a marker is placed at zero on the number line, and at each step moves +1 or −1 with equal probability.



- There are numerous other versions, for example: higher dimensions, biased walks, graph walks, etc.

Random walk in two dimensions.
https://upload.wikimedia.org/wikipedia/commons/f/f3/Random_walk_2500_animated.svg

Properties of Random Walks

- What is the average distance from origin after *n* steps?
- What is the probability that a walk will ever return to the origin?
- What is the probability that 2 simultaneous random walks will meet?
- …
- And what about >1D?

- We will merely scratch the surface, by exploring the "back to the origin" scenario.

Back to Origin in a 1D Simple Unbiased Random Walk

```
def back2origin():
    pos = 0
    num_steps = 0
    while True:
        step = random.choice([+1,-1])
        pos += step
        num_steps += 1
        print("step", num_steps, ": position", pos)
        if pos == 0:
            break
    return True
```

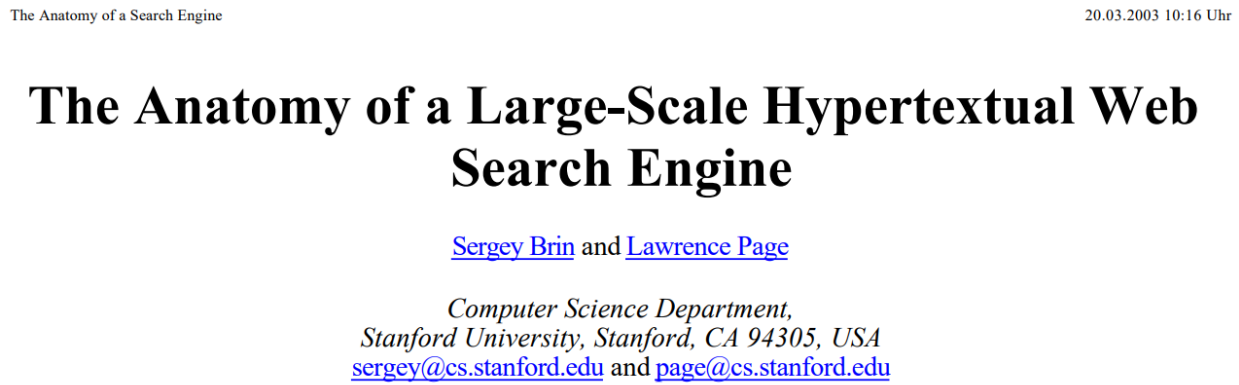
- It turns out (no proof here) that for dimension ≤ 2, a simple unbiased random walk will eventually return to the origin with probability 1.
- But for higher dimensions the probability decreases as the dimension grows.

Random Walk in Action

- Random walks are used to simulate various types of phenomena from a diverse range of fields, such as:
  - Economics: shares prices
  - Genetics: genetic drift (frequency of gene variants (alleles) in a population)
  - Physics: Brownian motion and diffusion
  - Ecology: population dynamics
  - Networks: Twitter's WTF ("Who to Follow")
  - Internet: Google's Pagerank algorithm to rank internet pages (next slide)
  - …

Random Walk at Google's PageRank

- Interestingly, random walks are used for ranking webpages



The Anatomy of a Search Engine 20.03.2003 10:16 Uhr

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

- Watch: https://www.youtube.com/watch?v=meonLcN7LD4
- Ranks of pages can be approximated empirically using random walks, as described in the above video, as well as computed analytically using equation systems and some algebra (in particular Eigenvectors (וקטור עצמי))

Lecture Summary

- What is randomness and basic notions
- `import random`
- Examples: random walk (and in particular PageRank)