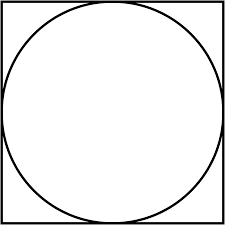
Algoritmi probabilisti:

* Las Vegas: oferau o solutie exacta in o complexitate de timp care variaza in functie de niste elemente generate aleator.
  + Ex: Quicksort - cu pivot selectat aleator. Complexitate medie O(n log n), worst case O(n2).
  + caz particular: paranoid Quicksort: se repeta alegerea aleatoare pt pivot pana cand se gaseste un pivot “bun”. In medie, un asemenea pivot se gaseste din 2 incercari. Complexitate totala O(n log n).
* Monte Carlo: o solutie “probabil” buna, intr-un numar determinat de pasio.
  + Frievald’s Algorithm pentru verificarea daca produsul a doua matrici este egal cu o a treia. Exista o sansa <½ ca algoritmul, la o iteratie, sa ofere un rezultat false positive. Ca atare, se repeta algoritmul de mai multe ori. Daca se repeta de 10 ori, de fiecare data cu un rezultat pozitiv (DA), atunci exista o sansa <(½)10 ca raspunsul corect sa fie unul negativ.



* Un algoritm simplu pentru aproximarea valorii lui pi:

Fie un patrat cu colturile (0,0) (0,1) (1,1) (1,0)

Si un cerc inscris in patrat:

centrul: (½,½ )

raza: ½

Generam *n* puncte in interiorul patratului. Dintre aceste unele vor fi si in cerc, iar altele vor fi in exteriorul cercului. Numaram cate puncte sunt in cerc si ne rezulta *p*.

Pentru valori foarte mari ale lui *n*, la cat va tinde raportul *p/n*?

Tinde la raportul dintre aria cercului si aria patratului.

Tinde la pi/4.

pentru algoritm avem nevoie:

* o structura de tip pereche sa tinem minte un punct
* o functie care genereaza aleator si uniform distribuit puncte in suprafata delimitata de patrat
* o functie care calculeaza distanta dintre doua puncte\*
* folosim functia pentru a verifica daca un punct generat este sau nu in cerc.
* generez in mod repetat cate un punct. verific daca acesta este in cerc si incrementez un contor. Fac raportul dintre contor si numarul total de puncte generat. Inmultesc raportul cu 4. astfel obtin o aproximare pentru pi.