

1. Each of the two pictures is a square with 100 points inside.

In both cases the coordinades of the points were calculated independently using a pseudorandom number generator. In one case an additional systematic modification of the coordinates was applied. The method of modification is unknown to us.

Which picture is the modified one? Formulate an argument which will support your guess.

**Solution:**

The first picture is the one with the method modification. If we draw a regular grid 10x10 over the picture, then we can check that each square of the grid contains exactly one generated point. This means that the points were not calculated for the whole area, instead of it, one point was calculated per each square of the grid.

2. It is easy to generate random numbers from the {1, 2, 3, 4, 5, 6} by throwing a dice. Suppose we have only one dice and we have to generate random integers in the interval [0, 10]. Describe the strategy of dice throwing which will generate each integer 0,1, ..., 10 with the same probability. (The dice is a classical 6-sided one).

**Solution:**

Throw the dice two times. There are 12 different outcomes, each of them occuring with the same probability. Take the first 11 outcomes to represent integers 0,1, ...,10. If the twelfth outcome occurs, repeat the experiment again (and do it until the twelfth outcome does not occur).

3. There is an array of sorted integer values. Describe a strategy which will rearrange the values into a random order using a psaudorandom number generator. The method should work in a time proportional to the length of the array.

By rearranging into a random order we mean that all possible permutations of the values are equally likely.

**Solution:**

A simple algorithm to generate a permutation of *n* items uniformly at, known as the *Knuth shuffle*, is go through the positions from 0 to *n* − 2 (we use a convention where the first element has index 0, and the last element has index *n* − 1), and for each position *i* swap the element currently there with a randomly chosen element from positions *i* through *n* − 1 (the end), inclusive. It is easy to verify that any permutation of *n* elements will be produced by this algorithm with probability exactly 1/*n*!, thus yielding a uniform distribution over all such permutations.

4. Find out whether the length of the period of the given linear congruential generator is maximum possible.

A) *xn*+1 = (91 *xn* + 49) mod 600 C) *xn*+1 = (37 *xn* + 55) mod 144

B) *xn*+1 = (8 *xn* + 80) mod 49 D) *xn*+1 = (99 *xn* + 81) mod 113

**Solution:**

The linear congruential generator has the form *xn*+1 = (A *xn* + C) mod M.

The length of period is maximum, i.e. equal to M, if and only if:

1. C and M are coprimes.

2. A – 1 is divisible by each prime factor of M.

3. If 4 divides M then also 4 divides A – 1.

We have to check validity of the conditions for formulas A), B), C) and D).

A) the third condition is not fulfilled, 4 divides 600, but 4 does not divide 90

B) all three conditions are fulfilled, the generator has period of maximum length

C) all three conditions are fulfilled, the generator has period of maximum length

D) the second condition is not fulfilled, M=113 is a prime, however, A–1=98 is not divisible by 113

5.Determine the period length in output of the Lehmer generator given by the relation

*xn*+1 = ((M−1)∙*xn*) mod M, (M is a prime).

**Solution:**

For some initial value *x0*, such that 0 <= *x0* < M, we derive

*x*1 = ((M−1)∙*x0*) mod M = (M∙*x0* − *x0*) mod M = −*x0* mod M

*x*2 = ((M−1)∙*x1*) mod M = −*x1* mod M = − (−*x0* mod M) mod M = (*x0* mod M) mod M = *x0*

This shows that the period length is 2.

6. Determine the upper and the lower bound of number of primes in the interval

A) [0, 109], B) [109, 2∙109], C) [2∙109, 3∙109].

**Solution:**

We apply formula n/ln n < π(n) < 1.25506 ∙ n/ln n. For example, 1.25506 ∙ (2∙109)/ln (2∙109) − (109)/ln (109) is an upper bound on the number of primes in interval B).

7. We say that an integer as a quasi-prime if it is an integer power of a prime. Write a pseudo-code of a modification of Eratosthenes' sieve which will generate exactly all quasi-primes.

8. We say that an integer as a half-prime if it is a product of two primes. Write a pseudo-code of a modification of Eratosthenes' sieve which will generate exactly all half-primes.

9. A set {1000, 1001, …, 999999} was originally given. Then, all multiples of all primes less then 1000 (2, 3, 5, …, 991, 997) were excluded from S. Give an estimate of the cardinality of S and of the number of primes in S.

10. Determine the maximum number of primes in any of the intervals [30*k*, 30*k*+29], *k* = 1, 2, 3, 4, ... .

**Solution:**

30*k* is divisible by the primes *2*, *3* and *5*. We can eliminate numbers marked by the sieve of Eratosthenes for these primes (prime two eliminates *30k*, *30k+2*, *30k+4*, … etc.). The non-eliminated numbers might be the primes we search for.

11. Find the numerical value of

A) GCD(220, 284), B) GCD , C) GCD(2100, 100!)

**Solution:**

A) 220=2∙2∙5∙11, 284=2∙2∙71, hence GCD(220, 284)=2∙2=4

B) ,

This implies GCD

C) We need to calculate the exponent of 2 in the factorization of 100!.

There are 50 numbers divisible by 2 among numbers 1,2,3,…,100.

There are 25 numbers divisible by 4 among numbers 1,2,3,…,100.

There are 12 numbers divisible by 8 among numbers 1,2,3,…,100.

There are 6 numbers divisible by 16 among numbers 1,2,3,…,100.

There are 3 numbers divisible by 32 among numbers 1,2,3,…,100.

There are 1 numbers divisible by 64 among numbers 1,2,3,…,100.

The exponent of 2 in 100! is thus 50+25+12+6+3+1=97 and GCD(2100, 100!) = 297.

12. Example of modular exponentiation.

1889 mod 11 =

= 181011001B mod 11 =

= 181000000B + 10000B + 1000B + 1B  mod 11 =

= (181000000B ∙ 1810000B ∙181000B ∙181B ) mod 11 =

= ((181000000B mod 11) ∙ (1810000B mod 11) ∙ (181000B mod 11) ∙ (181B mod 11)) mod 11. (\*\*)

Intermediate calculations:

181B mod 11 = **7**

1810B mod 11 = ((181B mod 11) ∙ (181B mod 11)) mod 11 = (7∙7) mod 11 = 5

18100B mod 11 = ((1810B mod 11) ∙ (1810B mod 11)) mod 11 = (5∙5) mod 11 = 3

181000B mod 11 = ((18100B mod 11) ∙ (18100B mod 11)) mod 11 = (3∙3) mod 11 = 9

1810000B mod 11 = ((181000B mod 11) ∙ (181000B mod 11)) mod 11 = (9∙9) mod 11 = 4

18100000B mod 11 = ((1810000B mod 11) ∙ (1810000B mod 11)) mod 11 = (4∙4) mod 11 = 5

181000000B mod 11 = ((18100000B mod 11) ∙ (18100000B mod 11)) mod 11 = (5∙5) mod 11 = 3

Return to (\*\*):

((181000000B mod 11) ∙ (1810000B mod 11) ∙ (181000B mod 11) ∙ (181B mod 11)) mod 11 = (3∙4∙9∙7) mod 11 = 8.

Conclusion:

1889 mod 11 = 8.

13. Use the method presented in the previous example to compute the given values. In some cases, you may apply some reasoning to compute the result even faster:

A) 18189 mod 11 B) 2100 mod 20 C) 850 mod 7 D) 123456 mod 1000

14. The given code calculates integer power *xn*. Modify the code in such way that it will calculate *xn* mod *m*, for positive integer *m*.

BinPower(int *x*, int *n*) {

int *r* = 1, *y* = *x;*

while (*n* > 1) {

if (n % 2 == 1) *r* \*= *y;*

*y* \*= *y;*

*n* /= 2;

}

return *r\*y;*

}