THE METHOD OF STRUCTURIZATION OF A SET OF POSITIVE INTEGERS AND ITS APPLICATION TO THE PRIMALITY TESTING ALGORITHM

ALEXANDER FEDOROV

December 2008

Abstract

In this paper is offered and theoretically is based the algorithm permissive with the help of small number of arithmetic operations with arbitrary positive integer (N) to answer a question : is N composite or prime? The algorithm has a high operational speed which depends a little on value N ,and is based on The method of structurization of a set of positive integers (N_p) developed by the author. In limits of a framework of this method is defined a special set of the structured integers (N_s) in which it becomes possibility for testing of any structured integers (S_n) on a membership of a set of composite structured integers (N_sc). Between by N_p and N_s is established one-to-one correspondence : composite structured integers (S_nc) are corresponded to composite positive integers (N_c). Prime structured integers (S_np) are corresponded to prime positive integers (N_pr). Thus for testing arbitrary (N) it is necessary to map it into N_s. Then we test obtained S_n on a membership of N_sc. If S_n is a member of N_sc then the output follows that tested N is also composite.If S_n is not a member of N_sc then the output follows that tested N is also prime , since if S_n is not composite then it is prime ,tertiary is not given.

1 Introduction

The most of existent primality testing algorithms [1] require considerable quantity of arithmetic operations and operate with numbers considerably greater than tested N. With increase N the speed of algorithm diminishs. The Author has put before itself a problem of creation of algorithm, the speed of which depends a little on value of tested N. The method of structurization of a set of positive integers given below allows theoretically and practically to solve this problem.

2 General conception

remark 2.1 *The accepted abbreviations in this paper see on page 16.*

Definition 2.1 *The structurization of a set of positive integers* (N_p) *is its fragmentation on* K *ordered segments which follow one by one on the numerical axis without skips.The amount of numbers in each segment is identically and is equal* B *, which we shall term as a length of a segment.There are* $\frac{1}{2}B$ *of the even and* $\frac{1}{2}B$ *of the odd positive integers in each segment.Among of the odd integers can be as composite so prime.By elements of a set of the structured integers* (N_s) *are structured integers* (S_n) *which uniquely are determined by three integer variables* K, r, B*. And a position* S_n *on the numerical axis is congruent with a position of corresponding* N*.*

Example 2.1 *Let be* $N = 54$ *,* $B = 10$ *, then* $K = 5$ *,* $r = 4$ *;*
$N = 156$ *,* $B = 10$ *, then* $K = 15$ *,* $r = 6$ *;* $N = 7$ *,* $B = 10$ *, then* $K = 0$ $r = 7$ *;*

Theorem 2.1 *Between* N_p *and* N_s *there is a bijection.*

$$N = (BK + r) \tag{1}$$

Where: B *is a length of segment (see theorem 7.1) and* $K = N div B$ *is a number of segment;* $r = N mod B$ *is an offset of* N *inside of segment.*

Proof 2.1 *Let is given* $S_n(K, r, B)$ *then coordinate of a segment on a numerical axis is equal* BK*. And offset is equal* r *then a coordinate* S_n *is equal* $BK + r$*. Taking into account that positions* N *and* S_n *are congruent it follows that* $N = BK+r$ □

Definition 2.2 *A Composite structured integer* (S_nc) *is multiplication of two arbitrary* S_n*:* $(BK_1 + r_1); (BK_2 + r_2);$*, each of which can be as composite so prime.*

$$S_nc = (BK_1 + r_1)(BK_2 + r_2) \tag{2}$$

Where: $K_1, K_2 = 0, 1, 2, 3, ...; r_1, r_2 = 0, 1, 2, ..., B - 1;$ *(numbers of segments and offsets of multipliers).*

Theorem 2.2 *Between a set of composite positive integers (N_pc) and a set of composite structured integers (N_sc) there is a correspondence.*

$$N_c = (BK_1 + r_1)(BK_2 + r_2) \tag{3}$$

Proof 2.2 *Let be $N_c = ab$ where a,b $\in N_p$ then by theorem 2.1 we have $\varphi a = BK_1 + r_1$; $\varphi b = BK_2 + r_2$; $\varphi ab = (BK_1 + r_1)(BK_2 + r_2)$; whence follows that $N_c = (BK_1 + r_1)(BK_2 + r_2)$;* $\square$

3 The course-of-value functions for a set of composite structured integers

Propose 3.1 *The course-of-value function (CVF)for N_sc is expression of the following view: $K = BK_1K_2 + K_1r_2 + K_2r_1 + (r_1r_2)divB$ where : $K_1, K_2 = 0, 1, 2, 3, ...$; $r_1, r_2 = 0, 1, 2, ..., B - 1$(numbers of segments and offsets of multipliers); B (see theorem 7.1); and $K = NdivB$ -number of segment in which is located an arbitrary S_nc.*

Proof 3.1 *Unclosing a parentheses in (3) we get :*

$$N_c = B^2K_1K_2 + BK_1r_2 + BK_2r_1 + r_1r_2 \tag{4}$$

By theorem 2.1 we have: $K = (B^2K_1K_2 + BK_1r_2 + BK_2r_1 + r_1r_2)divB$ whence we compute number of a segment K in which arbitrary S_n is composite:

$$K = BK_1K_2 + K_1r_2 + K_2r_1 + (r_1r_2)divB \tag{5}$$

Next by theorem 2.1 we compute offset r : $r = (B^2K_1K_2 + BK_1r_2 + BK_2r_1 + r_1r_2)modB$ whence we get the condition of membership of arbitrary S_n to one of the classes of CVFs:

$$r = (r_1r_2)modB \tag{6}$$

Fix in (5) r_1, r_2 and we shall change of K_1, K_2 from 0 up to ∞ . Thus we shall cover the some subset of N_sc. We shall term (5) as a Course-of-Value Function(CVF) of N_sc. Each CVF differs from another by pair of values (r_1, r_2) (B is constant for all CVFs of the class which belong to arbitrary S_n). Then the number of CVFs covering N_sc will be equally to (F)- number of all possible pairs formed from all positive integers less than B i.e. the number of all possible CVFs. $F = B \times B = B^2$. $\square$

remark 3.1 *For the some values K_1, K_2, r_1, r_2 in (5) CVF is undefined $(K = 0)$ or mixed type (prime and composite).*

Example 3.1 *If in (5) put $K_1 = 0, r_1 = 0$ then $K = 0$ i.e. CVF is undefined. If in (5) put $K_2 = 0, r_2 = 0$ then $K = 0$ i.e. CVF is undefined. If in (5) put $K_1 = 0, r_1 = 1$ then $K = BK_2 + r_2$ i.e. CVF is mixed type. If in (5) put $K_2 = 0, r_2 = 1$ then $K = BK_1 + r_1$ i.e. CVF is mixed type.*

For exclusion of these shortcomings it is necessary to test in algorithm the conditions $r_1 > 1$; $r_2 > 1$. If these conditions are not fulfil then $(K_1$ or K_2 or K_1 and $K_2) = 1, 2, 3, ...$;

4 The classification of the course-of-value functions

Propose 4.1 *There are following classes of CVFs :*
$BK + 0, BK + 2, BK + 4, ..., BK + 2m$ for $S_n ce$
and $BK + 1, BK + 3, BK + 5, ..., BK + (2m + 1)$ for $S_n co$
where: $m = 0, 1, 2, 3, ...$. The mean number of CVFs in each class is equal to $F_m = B$;

Proof 4.1 *For a beginning we shall separate CVFs for even and odd S_n. For this is necessary that $B = 10(m + 1)$. where $m = 0, 1, 2, 3, ...$. Then $S_n mod 10 = (BK + r) mod 10 = r mod 10$, whence follows that if S_n is odd structured integers then r is also odd . If S_n is even structured integers then r is also even. This allows neatly to separate CVFs on two subsets: for even and odd S_n. By condition (6) if r is odd then permissible values r_1, r_2 are also odd. If r is even then permissible values r_1, r_2 are both of even or one of them is odd. Thus for structured composite odd integers $(S_n co)$ permissible values r_1, r_2 are all odd N less than B. And permissible values r_1, r_2 for structured composite even integers $(S_n ce)$ all N less than B. Since r gets values from 0 up to (B-1) then the number of classes is equal to B. Including $\frac{1}{2}B$ classes of $S_n ce$ and $\frac{1}{2}B$ classes of $S_n co$. By (1) each class has a following view :$BK + r$ then for $S_n ce$ we get following classes: $BK+0, BK+2, BK+4, ..., BK+2m$ where m=0,1,2,3,... and for $S_n co$ we get following classes: $BK+1, BK+3, BK+5, ..., BK+(2m+1)$. The mean number of CVF in each class is equal to $F_m = \frac{F}{B}$;*
By proof 3.1 $F = B^2$ then $F_m = \frac{B^2}{B} = B$; □

Example 4.1 *Let is given arbitrary structured integer $(K; r; B;)$. Then we generate all possible pairs $(r_1; r_2;)$ of integers less than B . For each* *pair $(r_1;r_2)$ we test the condition $r = (r_1r_2) mod B$ .If the condition is true then value of pair $(r_1;r_2)$ are saved to array of CVFs of the class. When all pairs are sorted out in the array are saved all the CVFs of the class to which belongs S_n .*

5 Testing of arbitrary structured integer on a membership of a set of composite structured integers

Theorem 5.1 *$S_n(K, r, B)$ is composite if for anyone K, K_2, r_1, r_2, B is fulfilled one of the following conditions :*

$(K-(r_1r_2) div B) mod r_1 = 0$; $(K-K_2r_1-(r_1r_2) div B) mod (BK_2+r_2) = 0$;

Proof 5.1 *From (5) we compute to K_1*

$$K_1 = \frac{K - K_2r_1 - (r_1r_2) div B}{BK_2 + r_2} \tag{7}$$

Since K_1 is integer variable then a numerator in (7) should is divided on a denominator without a residual. The condition for this purpose as follows:

$$(K - K_2r_1 - (r_1r_2) div B) mod (BK_2 + r_2) = 0 \tag{8}$$

This condition cannot be used in the case when $K_1 = 0$ since condition $(K - K_2r_1 - (r_1r_2) div B) < (BK_2 + r_2)$ K_1 does inaccessible it . In this case it needs the numerator in (7) is equated to a zero:

$$(K - K_2r_1 - (r_1r_2) div B) = 0$$

Whence follows that:

$$K_2 = \frac{K - (r_1r_2) div B}{r_1} \tag{9}$$

Since K_2 is integer variable then a numerator in (9) should is divided on a denominator without a residual. The condition for this purpose as follows:

$$(K - (r_1r_2) div B) mod r_1 = 0 \tag{10}$$

At first we test condition (10). If it is true then the output follows that tested S_n is composite. And if it is false then we test condition (8) for each value of the CVF, varying K_2 in the range $0, 1, 2, ...$ If for anyone value of

K_2 the condition (8) is true then the output follows that tested S_n is composite. If by the time of when $(K - K_2r_1 - (r_1r_2)divB) < (BK_2 + r_2)$ and the condition (8)still is false then make a jumper to the next CVF of the class and process is iterated as above. After testing of all CVFs of the class and if the condition (8) still is false then the output follows that tested S_n is not composite (is prime). □

6 The dead zone of algorithm

Definition 6.1 *The fixed database (FDB) represents a subset of the set of positive integers composed of the all primes lees than N_h (Maximal limit of the dead zone of algorithm).*

Lemma 6.1 *The dead zone of algorithm is located in the range $N_l \leq N \leq N_h$ where :$N_l = 2; N_h = 189;$*

Proof 6.1 *By definition 2.2 the negative values of K_1, K_2 in (8),(10) are not permissible. However for the definite values of K, K_2, r_1, r_2, B the numerators of (8),(10) get a negative values. Since the case of $K_1 = 0$ is considered by separately then minimal value for $K_1 = 1$. It follows that $K - K_2r_1 - (r_1r_2)divB = BK_2 + r_2$ then for K_m (the value of K corresponding to the maximal limit of the dead zone of algorithm) we shall get $K_m = BK_2 + r_2 + K_2r_1 + (r_1r_2)divB$.Further we compute the value of K_m for the following minimal values of $B = 10; K_2 = 0; r_1 = 1;$ and maximal value $r_2 = 9$ (in order to get maximal value of $(r_1r_2)divB$). We get for $K_m = 9 + 9 = 18$. By the theorem 2.1 we have for $N = BK + r = BK + (r_1r_2)divB$ for $K = K_m$ we get for maximal limit of dead zone $N_h = 10 \cdot 18 + 9 = 189$. The minimal limit of the dead zone of algorithm (N_l) is determined by the least integer more than "1". Hence N_l=2.* □

Theorem 6.1 *For $N \leq 189$ if $N \in FDB$ then N is prime,if $N \notin FDB$ then N is composite.*

Proof 6.2 *By definition 6.1 and lemma 6.1 if $N \leq 189$ is prime then it coincides with one of elements FDB and the output follows that tested N is prime. If $N \leq 189$ is composite then it is not a one of elements FDB and the output follows that tested N is composite.* □

7 The optimal value of B

Definition 7.1 *We shall make an estimate of speed of algorithm by a number of cycles of testing (V). By the number of cycles of testing* (V_0) *is understood a testing* S_n *on a membership of* $N_s c$ *for one CVF of the class. By the number of cycles of testing* (V_c) *is understood a testing* S_n *on a membership of* $N_s c$ *for all CVFs of the class.*

Further we shall examine a possibility of a diminution v and speeding up of algorithm.

Lemma 7.1

$$V_0 = \frac{4N - 3B^2}{6B^2}$$

Proof 7.1 *From (8) follows that* $V_0 = K_2$ *for the condition:*

$$K - K_2 r_1 - (r_1 r_2) divB = BK_2 + r_2$$

Whence we get

$$V_0 = \frac{K - r_2 - (r_1 r_2) divB}{B + r_1}$$

We substitute r_1, r_2 *its mean values:* $r_1, r_2 = \frac{1}{2}B$. *Whence we get:*

$$V_0 = \frac{K - \frac{1}{2}B - \frac{1}{4}B}{B + \frac{1}{2}B} = \frac{4K - 3B}{6B}$$

But as $K = \frac{N}{B}$ *then*

$$V_0 = \frac{4N - 3B^2}{6B^2} \qquad \square$$

Lemma 7.2

$$If\ B_1 = BD;\ then\ V_0 1 \approx \frac{V_0}{D^2}$$

Proof 7.2 *Let* $B_1 = DB$ *then*

$$V_0 1 = \frac{4N - 3B_1^2}{6B_1^2} = \frac{4N - 3D^2B^2}{6D^2B^2}$$

Finally

$$V_0 1 \approx \frac{V_0}{D^2} \qquad \square$$

Lemma 7.3

$$V_c = \frac{4N - 3B^2}{6B}$$

Proof 7.3 *Since by Propose 4.1 the number of CVFs equals B then* $V_c = V_0 B$ *.Whence by lemma 7.1 we get:*

$$V_c = V_0 B = \frac{4N - 3B^2}{6B} \qquad \square$$

Lemma 7.4

$$If\ B_1 = BD; then V_c 1 \approx \frac{V_c}{D}$$

Proof 7.4 *Let* $B_1 = DB$ *then*

$$V_c 1 = \frac{4N - 3B_1^2}{6B_1} = \frac{4N - 3D^2B^2}{6DB}$$

Finally

$$V_c 1 \approx \frac{V_c}{D} \qquad \square$$

Definition 7.2 *The optimal value of B is such value for the which* V_c *depends a little from N and all necessary conditions of testing (8),(10) also are fulfilled.*

Theorem 7.1 *For speeding up of algorithm and realization of conditions of testing it is necessary to increase B with growth N as follows*
$B = 10;\ for\ N\ < 10000$
$B = 100;\ for\ N \geq 10000$
$B = 1000;\ for\ N \geq 1000000$
....
*B = B*10; N=N*100;*

Proof 7.5 *By lemma 7.3 the number of cycles of testing* V_c *increases with growth of N , that reduces to a diminution of speed of algorithm. By lemma 7.4 for a diminution of the number of cycles of testing and increase of speed of algorithm it is necessary to increase of* B*. In an ideal it would be possible with the help of increase of B coupled with increase of* V_c $(B = (N div 10^5 + 1)10)$ *but then for some N the conditions of testing (8), (10) will not be fulfilled* $(B \geq K)$*. In accordance with the stated above*

by one of the best values of B depending on N will be as follows:
$B = 10;\ for\ N\ < 10000$
$B = 100;\ for\ N \geq 10000$
$B = 1000;\ for\ N \geq 1000000$
...
*B = B*10; N=N*100;* □

8 The Fedorov's algorithm for the testing of arbitrary N with the answer a question: is it composite or prime?

Offered algorithm is based on the following definitions, lemmas, theorems and proposes which belong to the " THE METHOD OF STRUCTURIZATION OF A SET OF POSITIVE INTEGERS" stated above. For the step 8.1:definition 2.1 . For the step 8.2:theorem 2.1,theorem 7.1 . For the step 8.3:propose 4.1 . For the step 8.4:theorem 6.1 For the step 8.5:proof 3.1,proof 4.1 . For the step 8.6:theorem 5.1,proof 5.1 . For the step 8.7:theorem 5.1,proof 5.1 .

Step 8.1 *Enter tested positive integer N then goto step 8.2.*

Step 8.2 *We map N into N_s , computing $K = NdivB$; $r = NmodB$; B by theorem* 7.1; *then goto step 8.3.*

Step 8.3 *We separate obviously composite numbers checking conditions $rmod2 = 0$; $rmod5 = 0$; . If these conditions are true (separately) then the output follows that " N is composite ". If the both conditions is false then goto step 8.4.*

Step 8.4 *If $N \leq 189$ (hit in an dead zone of algorithm) then we test N on membership of FDB: if N is identical with anyone element of FDB the output follows that "N is prime" . If N is not identical not with one of element of FDB then the output follows that "N is composite" . If $N > 189$ then goto step 8.5.*

Step 8.5 *We make an array of CVFs of the class to which belongs N.For this we form all possible pairs ($r_1; r_2$) of the odd integers less than B. Then for each pair($r_1; r_2$) we test the condition $r = (r_1 r_2) modB$ (it is the condition of membership of the class). If the condition is true then values of the pair($r_1; r_2$)are saved into array of CVFs of the class. If the* *condition is false then values of the pair($r_1; r_2$)are not saved into array of CVFs of the class. If all pairs of the odd positive integers less than B are sorted out then goto step 8.6.*

Step 8.6 *Test of arbitrary N on membership of $N_s co$ (special criterion $K_1 = 0$). For this we test the first condition $(K - (r_1 r_2) divB) modr_1 = 0$ for pair $(r_1; r_2)$ of array of the class. If the first condition is true for pair $(r_1; r_2)$ then the output follows that "N is composite" . If the first condition is false for pair $(r_1; r_2)$ then goto step 8.7.*

Step 8.7 *We test of arbitrary N on membership of $N_s co$ (general criterion). For this we read from array of CVFs of the class values of the first pair $(r_1; r_2)$ and substitute its in the second condition : $(K - K_2 r_1 - (r_1 r_2) divB) mod(BK_2 + r_2) = 0$; besides substitute computed on the step 8.2 values of K, B and also initial value $K_2 = 0$; or $K_2 = 1; if\ (r_1$ or r_2 or r_1 and $r_2) = 1$. Then we test the second condition. If the second condition is true then the output follows that "N is composite". If the second condition is false then we increment $K_2 = K_2 + 1$ and a testing is continuing as defined above . This process is repeating until then while $(K - K_2 r_1 - (r_1 r_2) divB) < (BK_2 + r_2)$. If for this time the second condition is false for all values of K_2 then we read from array of CVFs of the class values of the next pair $(r_1; r_2)$. And process defined above is beginning with testing of the first condition for each CVFs of the class and is repeating until then while the first or second conditions will become true for any CVF of the class or false for all CVFs of the class. In the last case the output follows that "N is prime" .*

9 The detailing of algorithm on TPascal

```
Step1:
(The enter of tested number)
begin
var
N:longint;
clrscr;
writeln ('Type tested number N then press Enter');
readln (N);
goto step2;
end;

Step2:
```

```
(Computation of B , K , r )
label
m1,m2;
var
N,K,r,B,U : longint;
(Computation of optimal value of B)
begin
B := 10; U :=10000;
m1: $if\ N > U\ then$
begin
B := B*10; U := U*100;
goto m1;
end; end else
(Computation of number of segment)
K := N div B;
(Computation of an offset)
r := N mod B;
goto step 3;
end;
```

```
Step 3:
(Separating of obviously composite numbers)
label
m1,m2;
var
N,K,r :longint;
begin
if r mod 2 = 0 then
goto m1;
else
if r mod 5 = 0 then
$ifN > 5\ then$
m1: begin
clrscr;
write ('N-composite');
writeln;
goto m2; end
else
else
```

```
goto step4;
m2: end;

Step4 :
(Testing of N in dead zone of algorithm )
label
m1;
const
FDB: array [1..44] of integer =
=(3,5,7,11,13,17,...,157,163,167,173,179,181,0,0,0);
var
N,t: longint;
if N ≤ 189 then begin
m1: if FDB [t] = N then begin
clrscr;
write('N-prime');
writeln;
end
else
if FDB[t] > 0 then begin
t :=t+1; goto m1;
end
else
begin
clrscr;
write ('N-composite');
writeln;
end ;
end
else
goto step5;

Step5 :
(Making of an array of CVFs of the class)
label
m1,m2,m3;
const
c=8000;
var
a : array[1..c] of longint;
d: array[1..c] of longint;
```

```
var
r,r1,r2,i,j,B : longint;
begin
r1 :=1; r2 :=1; i :=1; j :=1;
(Nulling of an array of CVFs of the class)
m1: a[i] :=0 ; d[j] :=0 ;
i :=i+1; j :=j+1;
if i < c then
goto m1
else
(Generation of pairs (r1;r2) for CVFs of the class)
i :=0; j :=0;
m2: if (r1*r2)mod B = r then begin
i :=i+1; j :=j+1;
a[i] := r1; d[j] := r2;
goto m3;
end
else
m3: r2 :=r2+2;
if r2 < B then
goto m2;
else
r1 ;=r1+2; r2 :=1;
if r1 < B then
goto m2;
else
goto step6 ;
end;

Step6:
(Testing of N (special criterion for K1=0 )
label
m1,m2;
var
r1,r2,i,j,k,B : longint;
begin
i :=1; j :=1;
m1: r1 := a[i]; r2 := d[j];
if r1 > 1 then
if (K-(r1*r2)div B )mod r1 = 0 (the first condition)
then begin
```

```
clrscr;
write ('N-composite');
writeln;
goto m2;
end
else else
if a[i]  > 0 then
begin
i :=i+1; j :=j+1;
goto m1;
end
else
goto step7;
m2: end;
```

Step7:
Testing of N (general criterion)

```
label
m1,m2,m3,m4;
var
r1,r2,i,j,K,B,K2 : longint;
begin
i :=1; j :=1;
m1: r1 := a[i]; r2 := d[j];
if r1 = 0 then goto m4 else
if r2 > 1 then
K2 := 0;
else
if r2 =1 then K2 := 1;
m2: if  (K-r1*K2-(r1*r2)div B)mod(B*K2+r2)=0  (the second condition)
then  begin
clrscr;
write('N-composite');
writeln;
goto m3;
end
else
if (K - r1 * K2 - (r1 * r2)divB) > (B * K2 + r2) then
begin
K2 :=K2+1;
goto m2;
```

```
end
else
$if\ a[i] > 0\ then$
begin
i :=i+1; j :=j+1; goto m1;
end
else
m4: begin
clrscr;
write('N-prime');
writeln;
goto m3;
end;
m3:  end;
```

10 The approbation of the offered algorithm

For approbation of the offered algorithm the author developed the program " Testpro ". With the help of this program and the tables of prime numbers the author has tested positive integers in a range from 3 up to 10000000. Coincidence of results 100 percents.

11 Differences of the offered algorithm from others

The algorithm uses for testing numbers which considerably less than N $(K = \frac{N}{B})$. The operation of division in algorithm on numbers of the form (BK_2+r_2) is realized with the stride parameter is equal to B , that allows much faster to obtain the result. From all a set of course-of-value functions is used only the small part relating to that class CVFs to which belongs tested number N. There is a possibility of achievement of a maximum speed of algorithm by union in groups tested numbers with identical B and r thus the array of $CVFs$ of the class forms once on the whole group and it allows to save more than 50 percents of time of testing. All, stated above and also the increase of B with growth of N allows considerably to speed up algorithm.

APPENDIX

THE ABBREVIATIONS

N_p -a set of positive integers .
N_s -a set of structured integers .
N_sc -a set of composite structured integers .
N_pc -a set of composite positive integers .
N_pr -a set of prime positive integers .
S_n - the structured integers .
S_nc -the composite structured integers .
S_nco -the odd composite structured integers .
S_nce -the even composite structured integers .
S_np -the prime structured integers .
N - the positive integers .
N_c -the composite positive integers .
K - a number of segment in which is located a tested number N.
K_1, K_2 - the numbers of segments of multipliers in composite structured integers.
r - an offset which determines a position of tested number N inside of segment.
$r1, r2$ - the offsets of multipliers in composite structured integers.
B - a length of segment which determines a quantity of integers in segment.
CVF - a course-of-value function for N_sc .
$CVFs$ - the course-of-value functions for N_sc .
FDB - the fixed data base composed of all primes less than 189 .
F - the number of all possible CVFs.
F_m - the mean number of CVFs in each class.
V_0 - the number of cycles of testing for one CVF of the class.
V_c - the number of cycles of testing for all CVFs of the class.
N_h - a maximal limit of the dead zone of algorithm.
N_l - a minimal limit of the dead zone of algorithm.

References

[1] O.N. Vasilenko: Number-Theoretic Algorithms in Cryptography,Translations of Mathematical Monographs, volume 232 , American Mathematical Society, (2006)