ӘОЖ 519.682.2

Д.Б.Әлібиев, М.А.Сексембаева

Е.А.Бөкетов атындағы Қарағанды мемлекеттік университеті
(E-mail: dalibiev@mail.ru)

Жай сандарды табу жолын C++ тілінде оңтайландыру

Мақалада уақыт немесе компьютер жадын аз қолданылуы жағынан тиімді болатын жай сандарды іздеу алгоритмін, мысалы, $N = 2*1010$ дейінгі жай сандарды табу, қарастыру жолдары келтірілген. Сонымен қатар Эратосфен алгоритмі, Миллер-Рабин тесті мен *BPSW* алгоритмі қарастырылып, әр алгоритмнің орындалу уақыты мен диапазонына қатысты кестелер мен графиктер салыстырмалы түрде бейнеленген.

Кілт сөздер: жай сандар, есептеуді оңтайландыру, алгоритм, Эратосфен алгоритмі, Миллер-Рабин тесті, BPSW алгоритмі, C++ тілі.

Жаңа жай сандарды зерттеу және табу көптеген математиктер буынының басты мәселесі болып келеді. Есептеу техникасының дамуымен бұл мәселе жаңа екпін алмай, сонымен қатар криптографияда практикалық қолданысқа ие болды. Жай сандар криптографияның, статистика мен басқа да есептеу салаларының бөлігі бола отырып, физика, химия, биология мен инженерлік істерде, фольклорда да қолданыс тапты [1–4].

Мақалада уақыт немесе компьютер жадын аз қолданылуы жағынан тиімді болатын жай сандарды табу алгоритмі қарастырылады, мысалы, $N = 2 \cdot 10^{10}$ дейінгі жай сандарды табу. Сонымен қатар қарастырылған әр алгоритмнің орындалу уақыты мен диапазонына қатысты кестелер мен графикалар бейнеленген. N үлкен мәнге ие болғанда жай санды табу қиындығын түсіну үшін біз қарапайым Решето Эратосфен алгоритмін шолудан бастайық [5]. Оны C++ бағдарламалау тілінде орындаймыз:

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <ctime>
typedef unsigned long long ULL;
 ULL get_next(const bool *s, const ULL N, const ULL current)
{  // бірінші элементті іздейміз == false, s + current_idx + 1 адресінен бастап массив соңына дейін.
  const bool* pointer = std::find(s + current + 1, s + N, false);
  // егер мұндай элементтер таппасақ, 0 қайтарамыз, немесе көрсеткіш пен массив арасындағы
//айырымды көрсетеміз
  return (pointer == s + N) ? 0 : pointer - s;
}
 int main()
{  const ULL max = 2000000;
  bool *checked = new bool [max];
  checked[0] = checked[1] = true;
```

```
  for (size_t i = 2; i<max; i++)
            checked[i] = false;
  ULL i;
  ULL current = 2;
  time_t now = clock();
    for (i = current; i < max && i; i = get_next(checked, max, current), current = i)
  {  for (ULL j = i * i; j < max; j += i)
 checked[j] = true;
 }
 std::cout << "Time: " << std::fixed << std::setprecision(3) << static_cast<double>(clock() - now) / CLOCKS_PER_SEC << std::endl;
  delete [] checked;
  std::cin.get();
  return 0;
}
```

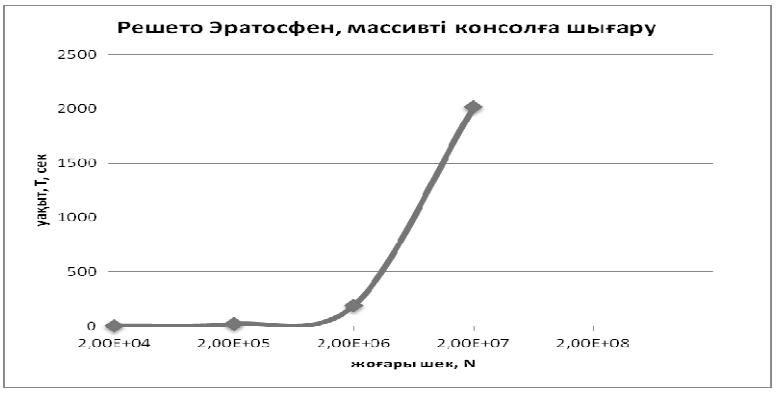
Бұл алгоритмнің орындалу жылдамдығын тестілейік. Жедел жадыда массивтер жиынын құрайтын болсақ, жоғары шек $N = 2*10^8$ тең. Сонымен қатар нәтижені консолға шығаратын жағдай мен консолда көрсетпеген жағдайда жылдамдық әр түрлі болады. Сәйкесінше көрсеткіштер 1-кестеде және график түрінде 1, 2-суреттерде бейнеленген.

1 - к е с т е

Решето Эратосфен алгоритмін қолданғандағы нәтиже

Жоғары шек, *N*	Орындалу уақыты, с	
	Нәтижені көрсетпеген кезде	Нәтижені көрсеткен кезде
$2 \cdot 10^4$	0,002	2,543
$2 \cdot 10^5$	0,028	20,246
$2 \cdot 10^6$	0,164	192,682
$2 \cdot 10^7$	2,117	2018,341
$2 \cdot 10^8$	21,079	Өлшенбеді

1-сурет. Решето Эратосфен алгоритмін қолданғандағы бағдарламаның орындалу уақыты



2-сурет. Решето Эратосфен алгоритмі көмегімен жай сандарды консолға шығарғанда өтетін уақыт

Көріп отырғанымыздай, қолданылған алгоритм $N <= 2*10^7$ болғанда тиімді және де осы диапазондағы көп деген есептер шешімінде қолдану үшін ұсынуға болады.

$N = 2 \cdot 10^{10}$ болғандағы нәтижелерді алу үшін бізге файл қажет, себебі жедел жады жеткіліксіз болып отыр. Сонымен қатар біз std::find қолдана алмаймыз.

N үлкен мәнде болғанда бұл әдістің кемшілігі — жай сандар қалатындай басқа сандарды алып тастау үшін файлға бірнеше рет жүгінуінде. $N = 2 \cdot 10^{10}$ болғанда бағдарламаның орындалу уақыты 10^4 секундтан асып кетеді. Егерде мақсатымыз — деректерді қамтитын файлды алу болса, онда бағдарламаның орындалу уақыты жүз есе баяулай түседі.

N үлкен мәнде болғанда жай сандар файлын бірден құратын алгоритмді қолдану қажет. Миллер-Рабин тестіне жүгінейік.

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <ctime>
 typedef unsigned long long ULL;
 ULL s,t,k;
ULL p[3]={2,7,61};
 ULL mulmod(ULL x,ULL y,ULL n)
{   ULL res=0;
    while(y)
    {   if(y&1) res=(res+x)%n;
        x=(x+x)%n;
        y>>=1;
    }
    return res;
}
 ULL powmod(ULL x,ULL y,ULL n)
{  ULL res=1;
    while(y)
    {
        if(y&1) res=mulmod(res,x,n);
        x=mulmod(x,x,n);
        y>>=1;
}
    return res;
}
bool miller(ULL s,ULL t,ULL a, ULL n)
{   ULL x=powmod(a,t,n);
if(x==1 || x==n-1)  return true;
for(ULL j=1;j<s;++j)
{   x=mulmod(x,x,n);
if(x==1)    return false;
```

```
if(x==n-1)  return true;
}
return false;
}
 bool del(ULL n)
{   ULL temp=(ULL) sqrt(1.*n);
for(ULL i=2;i<=temp;++i)
if(n%i==0)
return false;
return true;
}
 bool check(ULL n)
{   if(n<=1)    return false;
if(n==2)    return true;
if(n<=100)  return del(n);
t=n-1;
while(!(t&1))
{   ++s;
t>>=1;
}
bool ok=true;
for(ULL j=0;j<3 && ok;++j)
ok=miller(s,t,p[j], n);
return ok;
}
 int main()
{   time_t now = clock();
for (ULL n = 2; n<10000; n++)
if (check(n))
std::cout<<n<<"\n";
std::cout << "Time: " << std::fixed << std::setprecision(3) << static_cast<double>(clock() - now) /
CLOCKS_PER_SEC << std::endl;
std::cin.get();
 return 0;}
```

Бұл тест нәтижені бірден құруға мүмкіндік береді. Алдынғы шешіммен салыстырғандағы артықшылығы — біз жедел жады көлемімен шектелмеуімізде. Бірақ Миллер-Рабин тесті (3-сур.) бұл жерде Решето алгоритмінен баяу орындалады.

3-сурет. Миллер-Рабин тесті

Осы алгоритмді қиындатайық. BPSW алгоритміне көшейік, мұнда Миллер-Рабин тесті негізгі бөлім болып табылады.

BPSW — бұл үш тестінің комбинациясы: аз көлемді тривиалды бөлгіштікке тексеру, Миллер-Рабин тесті, Лукас-Селфридждің қуатты тесті.

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <ctime>
typedef long long LL;
const int trivial_limit = 50;
LL p[1000];
LL gcd (LL a, LL b)
{return a ? gcd (b%a, a) : b;
}
LL powmod (LL a, LL b, LL m)
{LL res = 1;
while (b)
if (b & 1)
res = (res * 1ll * a) % m,  --b;
else
a = (a * 1ll * a) % m,  b >>= 1;
return res;
}
bool miller_rabin (LL n)
{LL b = 2;
for (LL g; (g = gcd (n, b)) != 1; ++b)
if (n > g)
return false;
LL p=0, q=n-1;
while ((q & 1) == 0)
++p,  q >>= 1;
LL rem = powmod (b, q, n);
if (rem == 1 || rem == n-1)
return true;
for (LL i=1; i<p; ++i) {
rem = (rem * 1ll * rem) % n;
if (rem == n-1)  return true;
}
return false;
}
LL jacobi (LL a, LL b)
{if (a == 0)  return 0;
if (a == 1)  return 1;
if (a < 0)
if ((b & 2) == 0)
return jacobi (-a, b);
else
return - jacobi (-a, b);
LL a1=a,  e=0;
while ((a1 & 1) == 0)
a1 >>= 1,  ++e;
LL s;
if ((e & 1) == 0 || (b & 7) == 1 || (b & 7) == 7)
s = 1;
else
s = -1;
if ((b & 3) == 3 && (a1 & 3) == 3)
s = -s;
if (a1 == 1)
return s;
return s * jacobi (b % a1, a1);
}
bool bpsw (LL n)
{if ((LL)sqrt(n+0.0) * (LL)sqrt(n+0.0) == n)  return false;
```

```
LL dd=5;
for (;;)
{LL g = gcd (n, abs(dd));
if (1<g && g<n)  return false;
if (jacobi (dd, n) == -1)  break;
dd = dd<0 ? -dd+2 : -dd-2;
}
LL p=1,  q=(p*p-dd)/4;
LL d=n+1,  s=0;
while ((d & 1) == 0)
++s,  d>>=1;
LL u=1, v=p, u2m=1, v2m=p, qm=q, qm2=q*2, qkd=q;
for (LL mask=2; mask<=d; mask<<=1)
{u2m = (u2m * v2m) % n;
v2m = (v2m * v2m) % n;
while (v2m < qm2)   v2m += n;
v2m -= qm2;
qm = (qm * qm) % n;
qm2 = qm * 2;
if (d & mask) {
LL t1 = (u2m * v) % n,  t2 = (v2m * u) % n,
t3 = (v2m * v) % n,  t4 = (((u2m * u) % n) * dd) % n;
u = t1 + t2;
if (u & 1)  u += n;
u = (u >> 1) % n;
v = t3 + t4;
if (v & 1)  v += n;
v = (v >> 1) % n;
qkd = (qkd * qm) % n;
}
}
if (u==0 || v==0)  return true;
LL qkd2 = qkd*2;
for (int r=1; r<s; ++r)
{v = (v * v) % n - qkd2;
if (v < 0)  v += n;
if (v < 0)  v += n;
if (v >= n)  v -= n;
if (v >= n)  v -= n;
if (v == 0)  return true;
if (r < s-1) {
qkd = (qkd * 1ll * qkd) % n;
qkd2 = qkd * 2;
}
}
return false;
}
bool prime (LL n)
{for (LL i=0; i<trivial_limit && p[i]<n; ++i)
if (n % p[i] == 0)
return false;
if (p[trivial_limit-1]*p[trivial_limit-1] >= n)
return true;
if (!miller_rabin (n))
return false;
return bpsw (n);
}
```

```
void prime_init()
{for (LL i=2, j=0; j<trivial_limit; ++i)
{bool pr = true;
for (LL k=2; k*k<=i; ++k)
if (i % k == 0)
pr = false;
if (pr)
p[j++] = i;
}
}
int main()
{   prime_init();
LL top;
std::cin >> top; //жоғары шекті көрсету
std::cin.sync();
time_t now = clock();
for (LL n = 2; n<top; n++)
 if (prime(n))
 std::cout<<n<<"\n";
std::cout << "Time: " << std::fixed << std::setprecision(3) << static_cast<double>(clock() - now) / CLOCKS_PER_SEC << std::endl;
std::cin.get();
  return 0;}
```

Сәйкесінше, деректер 2-кестеде жиналған және 4, 5-суреттерде график түрінде бейнеленген.

2 - к е с т е

BPSW алгоритмін қолданғандағы нәтиже

Жоғары шек, N	Орындалу уақыты, с	
	Нәтижені көрсетпеген кезде	Нәтижені көрсеткен кезде
$2 \cdot 10^4$	0,006	2,286
$2 \cdot 10^5$	0,235	18,126
$2 \cdot 10^6$	2,571	197,5
$2 \cdot 10^7$	26,319	2013,221
$2 \cdot 10^8$	265,436	Өлшенбеді

4-сурет. BPSW алгоритмі көмегімен жай сандарды іздеу

5-сурет. BPSW алгоритмін қолданып жай сандарды консолға шығару

Егер алынған нәтижелерді 1-кестенің берілгенімен салыстырсақ, онда нәтижені алу уақыты бірдей деуге болады. Сонымен қатар *BPSW* алгоритмінде кемшіліктер жоқ, себебі көп жадыны қажет етпейді. Осылайша, BPSW алгоритмін қолдана отырып, біз $N = 2 \cdot 10^{10}$ дейінгі жай сандар файлын аламыз.

Сонымен, $N = 2 \cdot 1010$ дейінгі жай сандарды анықтап, оның файлын алу үшін BPSW алгоритмі қолданылды. Барлық түсініктемелер мен зерттеу нәтижелері кесте және графика түрінде айқындалып көрсетілген.

Орындалған есептеулер мен бағдарламалаулар келесідей сипаттамадағы компьютерде жүргізілді: Операциялық жүйе: Windows 7 кәсіби; Процессор: Intel(R) Core(TM) i7 3,3 GHz; жедел жады: 8 Гб.

Әдебиеттер тізімі

1 *Крэндалл Р., Померанс К.* Простые числа. Криптографические и вычислительные аспекты. — М., 2011. — 666 с.
2 *Жельников В.* Криптография от папируса до компьютера. — М., 1997. — 336 с.
3 *Василенко О.Н.* Теоретико-числовые алгоритмы в криптографии. — М.: МЦНМО, 2003. — 326 с.
4 [ЭР]. Режим доступа: http://e-maxx.ru/algo/bpsw — тест BPSW на простоту чисел.
5 [ЭР]. Режим доступа: http://e-maxx.ru/algo/prime_sieve_linear — Решето Эратосфена с линейным временем работы.

Д.Б.Алибиев, М.А.Сексембаева

Оптимизация поиска простых чисел на С++

В статье рассмотрены пути оптимизации поиска простых чисел с помощью алгоритма Решето Эратосфена, теста Миллера-Рабина и с помощью алгоритма *BPSW*. Приведены результаты программ при поиске простых чисел, например, максимум при $N = 2*1010$. Также данные программы на С++ оптимизированы по времени или по использованию памяти компьютера. Результаты нашли отображение в таблицах и в виде графика для каждого рассмотренного алгоритма.

D.B.Alibiyev, M.A.Seksembayeva

Optimization finding prime numbers in C++

In this article discusses ways to optimize for finding prime numbers using the Sieve of Eratosthenes algorithm, Miller-Rabin test and using an algorithm *BPSW*. In the work shows the results of the programs, where searching for prime numbers, such as a maximum for $N = 2*1010$. Also in the work shows programs in C ++ optimized by time or by use memory. The results were reflected in the tables and as a graph for each of the considered algorithm.

References

1 Crandall R., Pomerance K. *Primes. Cryptographic and computational aspects*, Moscow, 2011, 666 p.
2 Zhelnikov V. *Cryptography from papyrus to the compute*, Moscow, 1997, 336 p.
3 Vasilenko O.N. *Number-theoretic algorithms in cryptography*, Moscow: MCCME, 2003, 326 p.
4 http://e-maxx.ru/algo/bpsw - test BPSW the simplicity of numbers
5 http://e-maxx.ru/algo/prime_sieve_linear -Sieve of Eratosthenes with linear time work

ӘОЖ 517.518

Д.Б.Әлібиев, А.Б.Сейтімбетова

Е.А.Бөкетов атындағы Қарағанды мемлекеттік университеті
(E-mail dalibiev@mail.ru)

Интеллектуалды білім беру ресурстарын өңдеу үшін Web 2.0 технологиясын қолдану мүмкіндіктерін зерттеу

Мақалада постиндустриялық экономикаға мамандарды даярлау кезінде туындайтын дәстүрлі білім беру бағдарламаларының мәселелері сарапталды. Аталған мәселенің көбі Web 2.0. қолдану арқылы шешімін табатыны көрсетілді. Инновациялық білім беру бағдарламаларын дайындау үшін Web 2.0 құралдарын қолданудың отандық және шетелдік тәжірибесі зерттелді, білім беруде Web 2.0 құралдары, сервистері мен технологиясы қарастырылды. Интеллектуалды білім беру ресурстарын өңдеуге арналған Web 2.0 технологиясының мүмкіндіктері бағаланды.

Кілт сөздер: Web 2.0, білім жүйесі, жаңа технологиялар, білім беру бағдарламалары.

Нақты қоғамдағы білім беру мазмұнының қалыптасуы, білім беру технологиялары мен білім беруді ұйымдастырудың негізгі принциптері оның әлеуметтік-экономикалық құрылысына тән ерекшеліктерімен анықталады.

Мамандарды даярлау қоғамдық игілік іс ретінде қарастырылып, өндірістік жүйеден тыс, VII ғасырда негізі қаланған Ян Амос Коменскийдің авторитарлық сыныптық-сағаттық жүйесіндегі технократиялық тәсілдеме негізінде іске асады. Ол жаппай білім беруге және номенклатурасы баяу өзгеретін бұйымдардың жаппай өндірісінің қажеттіліктеріне бағытталған. Өз заманында бұл білім берудегі революциялық үлкен қадам болды [1].

Постиндустриялық қоғамда тауарларды жаппай өндіру мен баяу өзгеретін номенклатурадан ерекшелігі – өндірістің басқа түрі басымдылық танытады, атап айтқанда, тұтынушылардың жеке тапсырыстары бойынша қызмет көрсету және тауарларды шығару. Бұл жаңа антропоцентрлік тәсілдеме негізіндегі «нарықпен басқарылатын» индустрияның пайда болуына әкелді.

Жаңа индустрияның негізгі ұйымдастырушы формасы зауыт немесе фабрика емес (шоғырландырылған өндіріс), керісінше, шашыраңқы орналасқан өндіріс: корпорациялар, өнеркәсіп кластерлері, трансұлттық холдингтер; экономиканың жаһандануы жүреді. Жаңа әлеуметтік-экономикалық құрылыста жоғары «өмір сапасы» қамтамасыздандырылады, өндірістің біріншілік факторы — интеллектуалды капитал болады.

Өткен ғасырдың 60-жылдарының ортасында білім беруге деген жаңа талаптарды дәстүрлі сыныптық-сағатттық жүйе негізінде жүзеге асыру мүмкін болмады. Білім берудің сыныптық-сағаттық жүйесі қатты сынға ұшырады. Оған дәлел ретінде Мичиган университетінің (АҚШ) профессоры Д. Сангердің сөзінен үзіндіні келтіруге болады: «Біз жаппай білім берудегі ұлы эксперименттің аяқталуына жақындап қалдық. Ян Амос Коменскийдің сыныптық-сағаттық жүйесі сәтсіз болып шықты, ол сауатсыздықтың жоғары деңгейін көрсететін, оқымаған жұмыскерлер буынын жасап шығарды, жүйе әрі қарай оқуға деген талпыныстыны жойып жібереді. Аталған жүйе саны аз, таңдаулы топты (элитаны) даярлауға бағытталып, мүмкіндігі шектеулі енжар бейбақтарды қалыптастырады».