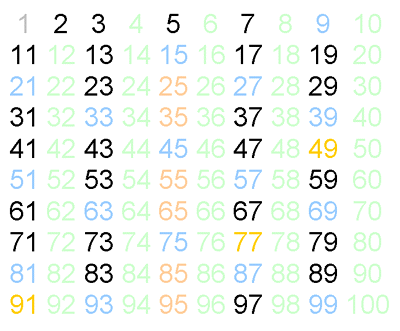
Prime Number Theory

***লেখক: শাফায়েত***

[**প্রাইম জেনারেটর (Sieve of Eratosthenes)**](http://www.shafaetsplanet.com/planetcoding/?p=624)

প্রাচীনকাল থেকেই গণিতবিদরা মাথা ঘামাচ্ছেন প্রাইম নাম্বার বা মৌলিক সংখ্যা নিয়ে। প্রাইম নাম্বারগুলো মধ্যে লুকিয়ে আছে বিষ্ময়কর কিছু সৌন্দর্য। যেকোনো কম্পোজিট বা যৌগিক সংখ্যাকে একাধিক প্রাইমের গুণফল হিসাবে মাত্র একভাবে লেখা যায়,ঠিক যেমন সব যৌগিক পদার্থ একাধিক মৌলিক পদার্থের সমন্বয়ে তৈরি। প্রাচীনকাল থেকেই মানুষ প্রাইম নিয়ে গবেষণা করছে,চলছে এখনো। গাউস,ফার্মা,ইউলারের মত কিংবদন্তি গণিতবিদরা কাজ করেছেন প্রাইম নিয়ে।



দ্রুত গতিতে প্রাইম সংখ্যা বের করার একটি পদ্ধতি আবিষ্কার করেন Eratosthenes,২০০ খ্রিস্টপূর্বের একজন গ্রীক গণিতবিদ,বিজ্ঞানি ও কবি। ২২০০ বছরেরও পুরানো সেই পদ্ধতি ব্যবহার করে আমরা আধুনিক কম্পিউটারে প্রাইম জেনারেট করি,খুব কম সময়ে বের করা যায় ১০কোটির নিচে সব প্রাইম সংখ্যা। এই অ্যালগোরিদমটি sieve of Eratosthenes নামে পরিচিত। sieve এর শাব্দিক অর্থ হলো ছাকনি যা অপ্রয়োজনীয় অংশ ছেটে ফেলে । Eratosthenes এর ছাকনি যৌগিক সংখ্যাগুলোকে ছেটে ফেলে দেয়।

Sift the Twos and sift the Threes,  
The Sieve of Eratosthenes.  
When the multiples sublime,  
The numbers that remain are Prime  
(Traditional,collected from wikipedia)

আমরা জানি প্রাইম সংখ্যা হলো সেসব সংখ্যা যাদের ১ এবং সেই সংখ্যটি ব্যতিত কোনো সংখ্যা দিয়ে ভাগ করা যায়না,যেমন ২,৩,৫,৭,২৯ ইত্যাদি। অন্যভাবে বলা যায় **সেসব সংখ্যাই প্রাইম যাদেরকে সংখ্যাটির বর্গমূলের সমান বা ছোটো কোনো প্রাইম দিয়ে ভাগ করা যায় না**। এই সংজ্ঞাটাই অ্যালগোরিদমের মূল অংশ,তাই আগে আমরা এটা বুঝতে চেষ্টা করব। ফর্মালভাবে প্রমাণ না করে ব্যপারটি বুঝানোর চেষ্টা করি। যেকোনো সংখ্যাকে আমরা কয়েকটি প্রাইমের গুণফল হিসাবে লিখতে পারি যাদের প্রাইম ফ্যাক্টর বলা হয়:

n=p1\*p2\*p3….\*pk

n যদি নিজেই প্রাইম হয় তাহলে n=p1(=n)। অন্যথায় অবশ্যই একাধিক প্রাইম ফ্যাক্টর থাকতে হবে। এবার চিন্তা করো কোনো সংখ্যা c কে দুটি সংখ্যার গুণফল c=a\*b হিসাবে লিখলে a আর b এর একটি অবশ্যই সংখ্যাটির বর্গমূলের থেকে ছোট,অন্যটি বড়। a,b দুটো সংখ্যাই c এর বর্গমূলের থেকে বড় হলে গুণফল c থেকে বড় হতো (ঠিক যেমন c=a+b হলে a বা b এর একটি c এর অর্ধেকের থেকে ছোট অন্যটি বড়)।

এবার n=p1\*p2\*p3….\*pk তে ফিরে আসি। p1,p2,p3 ইত্যাদির মধ্যে যে কোনো ২টি যদি n এর বর্গমূল থেকে বড় হয় তাহলে তাদের গুণফল n কে ছাড়িয়ে যাবে,তাই নয় কি? সর্বোচ্চ একটি প্রাইম ফ্যাক্টর বর্গমূলের বাইরে যেতে পারে,বাকি গুলো কে অবশ্যই ভিতরে থাকতে হবে।

তাহলে আমরা নিশ্চিত **যে যৌগিক সংখ্যা কে তার বর্গমূলের থেকে ছোট কোনো প্রাইম দিয়ে ভাগ করা যাবে।** ২য় সংজ্ঞাটি এখন আমাদের কাছে পরিষ্কার: ”সেসব সংখ্যাই প্রাইম যাদেরকে সংখ্যাটির বর্গমূলের সমান বা ছোটো কোনো প্রাইম দিয়ে ভাগ করা **যায় না**”। বুঝতে না পারলে আরেকবার ভালো করে চিন্তা করে নিচের অংশ পড়ো।

এবার আমরা আমাদের ছাকনি চালু করি এবং প্রাইম বের করি। ২৫ এর নিচের সব প্রাইম আমরা বের করব। ২৫ এর বর্গমূল ৫, তাই ২৫ বা তার থেকে ছোট কোন সংখ্যাকে অবশ্যই ৫ বা তার থেকে ছোট কোনো প্রাইম দিয়ে ভাগ করা যাবে।  
২ একটি প্রাইম কারণ ২কে তার বর্গমূলের নিচে কোনো সংখ্যা দিয়ে ভাগ করা যায়না। তাহলে ২ এর মাল্টিপলগুলো কেও প্রাইম নয় কারণ তাদের ২ দিয়ে ভাগ করা যায়,সেগুলোকে আমরা কেটে দেই:

২,৪,৬,৮,১০,১২,১৪,১৬,১৮,২০,২২,২৪

২ এর পরের সংখ্যা ৩। ৩ যদি প্রাইম না হতো তাহলে ৩ এর বর্গমূলের নিচের কোনো প্রাইম ৩ কে বাদ দিয়ে দিত,যেহেতু ৩ বাদ পড়েনি তাই সংজ্ঞামতে ৩ প্রাইম। ৩ এর মাল্টিপল গুলো কে বাদ দেই:

৩,৬,৯,১২,১৫,১৮,২১,২৪

পরের সংখ্যা ৪। ৪ বাদ পড়ে গিয়েছে আগেই। তারপর আছে ৫। ৫ যদি প্রাইম না হতো তাহলে আগেই ছাকনিতে কাটা পড়ত, ৫এর মাল্টিপল গুলোকে কেটে দেই:

৫,১০,১৫,২০,২৫

আমাদের আর কাটাকাটি প্রয়োজন নেই। ২৫ এর বর্গমূল ৫, তাই ২৫এর নিচের সব সংখ্যার বর্গমূল ৫ থেকে ছোট। সুতরাং ২৫ এর নিচের সকল যৌগিক সংখ্যা ৫ বা তার নিচের কোনো প্রাইম দিয়ে বিভাজ্য। যেহেতু আমরা ২,৩,৫ এর সব মাল্টিপল কেটে দিয়েছি,বাকি সংখ্যগুলো অবশ্যই প্রাইম। ছাকনির উপর থেকে সেগুলো সংগ্রহ করে নেই:

২,৩,৫,৭,১১,১৩,১৭,১৯,২৩

আমরা সিভের একটা কোড দেখি:

bool status[1100002];

void siv()

{

int N=1000000;

int sq=sqrt(N);

for(int i=4; i<=N; i+=2)

status[i]=1;

for(int i=3; i<=sq;i+=2)

{

if(status[i]==0)

{

for(int j=i\*i; j<=N; j+=i)

status[j]=1;

}

}

status[1]=1;

}

status অ্যারেটা দিয়ে নির্দেশ করে একটি সংখ্যা প্রাইম নাকি কম্পোজিট। status[i]=0 হলে i একটি প্রাইম। শুরুতে সব ইনডেক্সে ০ আছে, আমরা উপরের অ্যালগোরিদম অনুযায়ী নন প্রাইম সংখ্যা গুলোকে কেটে দিবো, অর্থাৎ j যদি নন-প্রাইম হয় status[j]=1 করে দিবো। ৮ নম্বর লাইনে শুরুতেই ২ এর সব মাল্টিপল কেটে দিলাম। এরপরের পরের লুপটা ৩ থেকে শুরু করে ২ করে বাড়াবো কারণ জোড় সংখ্যা নিয়ে আর চিন্তা করা দরকার নেই। ১০ নম্বর লাইনে এসে যদি status[i]=0 পাই তাহলে অ্যালগোরিদম অনুযায়ী i অবশ্যই প্রাইম কারণ i এখনও কাটা পড়েনি, এবার i এর সবগুলো মাল্টিপল কেটে দিবো, এজন্য j এর লুপ শুরু করবো 2\*i থেকে এবং বাড়াবো i পরিমাণ। আমাদের কাজ শেষ, নন-প্রাইম সংখ্যাগুলো সব কেটে দিবে ভিতরের লুপটি, এখন status[i] এর মান দেখে আমরা i প্রাইম কিনা বের করতে পারবো।

সিভ দিয়ে প্রাইম জেনারেট করে খুব সহজে কোন সংখ্যার প্রাইম ফ্যাক্টর বা উৎপাদকে বিশ্লেষণ করা যায়। এই কাজটা তোমার হাতেই থাকলো ।

সিভে প্রতিটি সংখ্যা প্রাইম নাকি নন-প্রাইম সেটা আমরা একটি বুলিয়ান অ্যারে দিয়ে চেক করি। যত পর্যন্ত প্রাইম জেনারেট করব তত সাইজের অ্যারে লাগবে। ১০^৮ আকারের অ্যরে অনেক মেমোরি দখল করে। মেমোরি অপটিমাইজ করার জন্য অসাধারণ একটি পদ্ধতি হলো বিট ব্যবহার করা,একে bitwise সিভ বলা হয়। একটি ইন্টিজারে ৩২টি বিট থাকে যার প্রতিটিকে আমরা ফ্ল্যাগ হিসাবে ব্যবহার করতে পারি, সেটা নিয়ে বিস্তারিত আলোচনা পাবে [এখানে](http://www.shafaetsplanet.com/planetcoding/?p=855)।

# [বিটওয়াইজ্ সিভ(Bitwise sieve)](http://www.shafaetsplanet.com/planetcoding/?p=855)

বিটওয়াইজ সিভ প্রাইম সংখ্যা বের করার জন্য প্রচলিত অ্যালগোরিদম [Sieve of Eratosthene](http://www.shafaetsplanet.com/planetcoding/?p=624) এ মেমরির ব্যবহার অনেক কমিয়ে আনা যায়! সাধারণ সিভে N পর্যন্ত প্রাইম জেনারেট করলে N সাইজের একটি অ্যারে ডিক্লেয়ার করতে হয়। অ্যরের প্রতিটি এলিমেন্ট একটি করে ফ্ল্যাগ হিসাবে কাজ করে যেটা দেখে আমরা বুঝি একটি সংখ্যা প্রাইম নাকি কম্পোজিট। বিটওয়াইজ্ সিভে আমরা ফ্ল্যাগ হিসাবে ইন্টিজার বা বুলিয়ান এর বদলে সরাসরি বিট ব্যবহার করবো।

এ টিউটোরিয়াল পড়ার আগে দুটি বিষয় তোমাকে জেনে আসতে হবে

১. Sieve of Eratosthene এর সাধারণ ভার্সণ,তুমি এটা আমার [এই পোস্টটি](http://www.shafaetsplanet.com/planetcoding/?p=624) পড়ে শিখতে পারবে সহজেই।

২. সি/সি++ এ বিটওয়াইজ্ অপারেটরের ব্যবহার। এটাও খুব সহজে শিখতে পারবে [এখান থেকে](http://zobayer.blogspot.com/2009/12/bitwise-operations-in-cc-part-1.html)। টিউটোরিয়ালটির ১ম ২টি অংশ খুব ভালো করে পড়ে ফেলো,বিটমাস্ক ডিপি,বিএফএস যখন শিখবে তখন অনেক কাজে লাগবে।

আশা করি এখন তুমি বিটওয়াইজ্ অপারেটর সম্পর্কে অনেক কিছু জানো,সাধারণ সিভ লিখতে কোনো সমস্যা হয়না তোমার। এবার আমরা শিখবো বিটওয়াইজ সিভ।

সাধারণ সিভে status বা flag অ্যারেটার কাজ কি? এই অ্যারের ইনডেক্সের মান দেখে আমরা বলতে পারি একটি সংখ্যা প্রাইম কিনা। ধরলাম তোমার status অ্যারেটা ইন্টিজার টাইপের। প্রতিটি ইন্টিজারের মধ্য আছে ৩২টি বিট। আমরা কেনো এতগুলো বিট ব্যবহার করবো খালি ০ বা ১ নির্দেশ করতে? আমরা অ্যারের যেকোনো ইনডেক্সের প্রতিটি বিট দিয়ে একটি সংখ্যা নির্দেশ করতে পারি।

তুমি যখন ইন্টিজার অ্যারেতে ১-৭ পর্যন্ত প্রাইম জেনারেট কর, তোমার অ্যারের অবস্থা বাইনারিতে থাকে এরকম:

status[১]=০০০........০০(৩২টি শূন্য)

status[২]=০০০........০১(৩১টি শূন্য,১টি ১)

status[৩]=০০০........০১(৩১টি শূন্য,১টি ১)

status[৪]=০০০........০০(৩২টি শূন্য)

status[৫]=০০০........০১(৩১টি শূন্য,১টি ১)

status[৬]=০০০........০০(৩২টি শূন্য)

status[৭]=০০০........০১(৩১টি শূন্য,১টি ১)

প্রতিটি ইনডেক্সে ৩১টি বিট কোনো কাজে লাগছেনা অথচ এই বিশাল সংখ্যক অব্যবহৃত বিট আমরা সহজেই কাজে লাগাতে পারি। আমরা ধরে নিবো:

status[০] এর

>>> শূন্যতম বিট ০ এর প্রাইমালিটি নির্দেশ করে (সবথেকে ডানের বিট==০ তম বিট)

>>> ১ম বিট ১ এর প্রাইমালিটি নির্দেশ করে (সবথেকে ডানের বিট==০ তম বিট)

>>> ২য় বিট ২ এর প্রাইমালিটি নির্দেশ করে

>>> ৩য় বিট ৩ এর প্রাইমালিটি নির্দেশ করে

.........................

>>> ৩১তম বিট ৩১ এর প্রাইমালিটি নির্দেশ করে

status[১] এর

>>> শূন্যতম বিট ৩২ এর প্রাইমালিটি নির্দেশ করে

>>> ১ম বিট ৩৩ এর প্রাইমালিটি নির্দেশ করে

...............

status[২] এর

>>> শূন্যতম বিট ৬৪ এর প্রাইমালিটি নির্দেশ করে

>>> ১ম বিট ৬৫ এর প্রাইমালিটি নির্দেশ করে

তাহলে ১-৭ পর্যন্ত প্রাইম জেনারেট করলে তোমার অ্যারের অবস্থা দাড়াবে:

status[১]= ০০০০…….১০১০১১০০(মোট ৩২টি বিট) (সবথেকে ডানের বিট==০ তম বিট)

৭টি সংখ্যার কাজ একটি ইনডেক্সেই শেষ!!। শুধু ৭টি নয়,আসলে ৩১টি সংখ্যার কাজ শেষ হবে ১টি ইনডেক্স(কারণ প্রতি ইনডেক্সের ৩১টি বিট ব্যবহার করছি আমরা)। এখন প্রশ্ন হলো কোনো সংখ্যার প্রাইমালিটি কত নম্বর ইনডেক্সের কত নম্বর বিট দিয়ে নির্দেশ করা হবে? খুব সহজ, সংখ্যাটি i হলে i/৩২ নম্বর ইনডেক্সের i%৩২ নম্বর বিট আমাদের চেক করতে হবে। তাহলে i=১ হলে চেক করবো ০ নম্বর ইনডেক্সের ১ নম্বর বিট, i=৩৩ হলে চেক করবো ১ নম্বর ইনডেক্সের ১ নম্বর বিট ইত্যাদি। (শুন্য বেসড ইনডেক্সিং)

কোডিং অংশ একদম সহজ। তুমি যেহেতু বিটওয়াজ অপারেটরের ব্যবহার জানো,কোনো সংখ্যার pos তম বিটে ১ বা ০ আছে নাকি সহজেই চেক করতে পারবে। pos তম বিটে নিজের ইচ্ছামত ১ বা ০ বসাতেই পারবে। আমাদের এখানে ০ বসানো দরকার নেই,১ বসাতে পারলেই চলবে। দুটি ফাংশন লিখে ফেলি:

bool Check(int N,int pos){return (bool)(N & (1<<pos));}

int Set(int N,int pos){ return N=N | (1<<pos);}

এবার সিভ লিখে ফেলি:

int N =100,prime[100];

int status[100/32];

void sieve()

{

int i, j, sqrtN;

sqrtN = int( sqrt( N ) );

for( i = 3; i <= sqrtN; i += 2 )

{

if( check(status[i/32],i%32)==0)

{

for( j = i\*i; j <= N; j += 2\*i )

{

status[j/32]=Set(status[j/32],j % 32) ;

}

}

}

puts("2");

for(i=3;i<=N;i+=2)

if( check(status[i/32],i%32)==0)

printf("%d\n",i);

}

লক্ষ্য করো,আমরা সাধারণ সিভের মত করেই সব লিখেছি,তবে status[i] এর মান চেক করার বদলে status[i/32] এর i%32 নম্বর বিটের মান চেক করেছি।

বিটওয়াইজ সিভ ব্যবহার করে ১০^৮ পর্যন্ত প্রাইম তুমি জেনারেট করে ফেলতে পারবে। সাধারণ সিভের থেকে সময় + মেমরি কম লাগবে। সাধারণ গুণ,ভাগ অপারেশনের থেকে বিটের অপারেশনগুলো দ্রুত কাজ করে। আমরা আরো কিছু অপটিমাইজেশন করতে পারি। যেমন তুমি উপরে দেয়া [টিউটোরিয়াল](http://zobayer.blogspot.com/2009/12/bitwise-operations-in-cc-part-1.html) পড়ে থাকলে জানো যে কাওকে ২ দিয়ে গুণ করা আর সংখ্যাটির বাইনারিকে ১ ঘর বামে শিফট করা একই কথা। আবার ২ দিয়ে ভাগ করা আর ১ ঘর ডানে শিফট করা একই কথা,৩২ দিয়ে mod করা আর ৩১ দিয়ে AND করা একই কথা। তাহলে আমরা নিচের মত করে কোডটি লিখতে পারি:

int status[(mx/32)+2];

void sieve()

{

int i, j, sqrtN;

sqrtN = int( sqrt( N ) );

for( i = 3; i <= sqrtN; i += 2 )

{

if( Check(status[i>>5],i&31)==0)

{

for( j = i\*i; j <= N; j += (i<<1) )

{

status[j>>5]=Set(status[j>>5],j & 31) ;

}

}

}

puts("2");

for(i=3;i<=N;i+=2)

if( Check(status[i>>5],i&31)==0)

printf("%d\n",i);

}

ফাংশন ব্যবহার না করে ম্যাক্রো ব্যবহার করলে আরো কম সময় লাগবে। [প্রোগ্রামিং কনটেস্টে](http://www.shafaetsplanet.com/planetcoding/?cat=35) খুব কমই বিটওয়াইজ সিভ ব্যবহার করা দরকার হয়,সাধারণ সিভেই কাজ চলে। তারপরেও এটা শিখলে বিটের কনসেপ্ট গুলো কিছুটা পরিস্কার হবে,অন্য কোনো প্রবলেমে হয়তো মেমরি কমিয়ে ফেলতে পারবে। ডাইনামিক প্রোগ্রামিং এ আমরা প্রায়ই বিটমাস্কের ব্যবহার করি মূলত মেমরি কমানোর জন্য।

তোমার ইম্প্লিমেন্টেশন সঠিক নাকি চেক করতে নিচের সমস্যাটি সমাধান করে ফেলো:  
<http://www.spoj.pl/problems/TDPRIMES/>

(নোট: অনেকের ভূল ধারণা আছে যে bool টাইপের ভ্যারিয়েবলের আকার ১বিট। আসলে bool এর আকার ৮বিট বা এক বাইট,char এর সমান। এর কারণ হলো কম্পিউটার ১ বাইটের ছোটো মেমরি সেগমেন্টকে অ্যাড্রেস করতে পারেনা, তাই ভ্যারিয়েবলের নূন্যতম আকার ১ বাইট)

UVA 10311

### মন্তব্য

হাসান  
UVA 10311 (Goldbach and Euler) এবার করে ফেলা যাবে!

শাফায়েত

ধন্যবাদ। 10311 এ অবশ্য bitwise sieve দরকার হয়নি আমার।

অনিন্দ্য

ভাই, ১৫ তম লাইনে j%31 এর পরিবর্তে j%32 হবে না? j%32 এবং j&31 একই হবার কথা। যদি P=2^x হয়, তাহলে n%p = n&(p-1) হওয়ার কথা। এটা নিয়ে একটু confusion হচ্ছে।

শাফায়েত

হ্যা j%32 হবে, আমি ঠিক করে দিচ্ছি। যদিও j%31 লিখলেও কোডটা কাজ করে এজন্য হয়তো খেয়াল করিনি ঠিকমত তাই কনফিউশন তৈরি হয়েছে।

[কাইদুল](http://www.kaidul.web44.net)

int(signed integer) এর left Most bit যেহেতু sign bit, তাই আমরা যখন একে left shifting করি তখন left most বিটে 1 পুশ করলে সংখ্যাটা নেগেটিভ হওয়ার সম্ভাবনা থাকে। তাই i<<1 ব্যবহার করা কি ঠিক হবে?   
আমি ভুল হতে পারি, for( j = i\*i; j <= N; j += 2\*i ) লাইনের মাধ্যমে কি সব মাল্টিপল বের হবে? for( j = 2\*i; j <= N; j += i ) দিলে হবার কথা !  
i%31 গুলোকে i%32 দিয়ে রিপ্লেস করলে ভাল হত। কিছুটা কনফিউজিং লাগছে

শাফায়েত

টেকনিক্যাল কিছু সমস্যার জন্য কোড এডিট করতে পারছিনা, ঠিক হলে রিপ্লেস করে দিবো।  
i\*i দিলেও কাজ হবে, <http://www.lightoj.com/article_show.php?article=100>1 এই লিংকটা দেখেন।

[আলিম](http://onlyprogramming.wordpress.com)

ভাইয়া, এই প্রবলেম এর ক্ষেত্রে কিভাবে প্রাইম Generate করব?  
<http://www.spoj.com/problems/KPRIMES2/>

শাফায়েত

এটায় মনে হচ্ছে আরো অ্যাডভান্সড কিছু লাগবে, এই থ্রেডটা দেখো : <http://apps.topcoder.com/forums/?module=Thread&threadID=698955&start=0&mc=5#1335192>