

শাফায়েতের ব্লগ

প্রোগ্রামিং ও অ্যালগরিদম টিউটোরিয়াল

Home


অ্যালগরিদম নিয়ে যত লেখা!

আমার সম্পর্কে...

প্রাইম জেনারেটর (Sieve of Eratosthenes)

📅 জুলাই ১৭, ২০১১ by শাফায়েত



প্রাচীনকাল থেকেই গণিতবিদরা মাথা ঘামাচ্ছেন প্রাইম নাম্বার বা মৌলিক সংখ্যা নিয়ে। প্রাইম নাম্বারগুলো মধ্যে লুকিয়ে আছে বিস্ময়কর 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, প্রোগ্রামিং এর জগতে সুন্দরতম অ্যালগোরিদমগুলোর মধ্যে এটি একটি।

sieve এর শাব্দিক অর্থ হলো ছাকনি যা অপ্রয়োজনীয় অংশ ছেটে ফেলে (A sieve, or sifter, separates wanted elements from unwanted material using a woven screen such as a mesh or net)। Eratosthenes এর ছাকনি যৌগিক সংখ্যাগুলোকে ছেটে ফেলে দেয়।

“ Sift the Twos and sift the Threes,
The Sieve of Eratosthenes.


 top

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 সংখ্যাটি ব্যতীত কোনো সংখ্যা দিয়ে ভাগ করা যায়না, যেমন ২, ৩, ৫, ৭, ১১ 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:

$$n = p_1 * p_2 * p_3 * \dots * p_k$$

n যদি নিজেই প্রাইম হয় তাহলে $n = p_1 (=n)$ । 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 $n = p_1 * p_2 * p_3 * \dots * p_k$ তে ফিরে আসি। p_1, p_2, p_3 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, বাকিগুলো 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:

$$2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24$$

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:

$$3, 6, 9, 12, 15, 18, 21, 24$$

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

“ ৫, ১০, ১৫, ২০, ২৫

আমাদের আর কাটাকাটি প্রয়োজন নেই। ২৫ এর বর্গমূল ৫, তাই ২৫ এর 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 সংখ্যাগুলো অবশ্যই প্রাইম। ছাকনির উপর থেকে সেগুলো সংগ্রহ করে নেই:

“ ২, ৩, ৫, ৭, ১১, ১৩, ১৭, ১৯, ২৩

আমরা সিভের একটা কোড দেখি:

```
1 bool status[1100002];
2 void siv()
3 {
4     int N=1000000;
5     int sq=sqrt(N);
6     for(int i=4;i<=N;i+=2) status[i]=1;
7     for(int i=3;i<=sq;i+=2){
8         if(status[i]==0)
9             {
10                 for(int j=i*i;j<=N;j+=i) status[j]=1;
11             }
12     }
13     status[1]=1;
14
15 }
```

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 বাড়ানোর 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 বাড়ানো 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। 10^6 আকারের অ্যারে 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 যা প্রতিটিকে আমরা ফ্ল্যাগ হিসাবে ব্যবহার করতে পারি, সেটা নিয়ে বিস্তারিত আলোচনা পাবে [এখানে](#)।

ফেসবুকে মন্তব্য

2 comments

2 Comments

Sort by Oldest



Add a comment...

**MD Selim Mia**

Please add Miller Rabin prinality test...

Like · Reply · 2 · 1y

**Mohammed Jafar Sadik**

please put a correction on 10th line
 for(int j=i*i;j<=N;j+=i) status[j]=1;
 it will be:
 for(int j=2*i;j<=N;j+=i) status[j]=1;

Like · Reply · 3 · 51w

**Abu Hurayra**

I think the code is correct and optimized. All other numbers less then $i*i$ have already been checked. If $i=7$, we do not need to check $7*5=35$, because it was checked when $i=5$ was in action. So, we only need to calculate $i*i$ and above numbers. Thank you.

Like · Reply · 7 · 43w

Facebook Comments plugin

Powered by Facebook Comments



Posted in অ্যালগোরিদম/প্রবলেম সলভিং, প্রোগ্রামিং ? Tagged গণিত, নাম্বার থিওরী, প্রাইম, সিভ

39,739 বার পড়া হয়েছে

◀ ইউভিএ ১০৭০২(ট্রোভেলিং সেলসম্যান)

গ্রাফ থিওরিতে হাতেখড়ি ৫: মিনিমাম স্প্যানিং ট্রি(প্রিম অ্যালগোরিদম) ▶

9 thoughts on "প্রাইম জেনারেটর (Sieve of Eratosthenes)"

^
top

**Late_riser**

অক্টোবর ২৩, ২০১৩ at ১১:২৬ am

Nice writing.

But, I have a confusion.

In the line 12 (for(int j=i*i;j<=N;j+=i) status[j]=1;) instead of "i*i", won't it be "2*i" ?

Thanks 😊

Reply

**Rahat Noman**

আগস্ট ২৮, ২০১৬ at ৪:১০ pm

মনে করুন $i=5$;

$5*2$ বা $5*3$ বা $5*4$ এর স্টাটাস চেঞ্জ করার ত দরকার নেই কারণ এগুলোতে ত অলরেডি চেঞ্জ হয়ে গেছে কারণ এরা ত ২,৩,৪ এর মাল্টিপল ছিল।

This is just an optimizaion. $2*i$ দিলেও কোন প্রবলেম নেইই।

Reply

**Aseem Chakrabarthy**

জানুয়ারি ১৫, ২০১৫ at ১২:২৭ pm

Excellent!!!

Reply

Pingback: [প্রাইম নাম্বার জেনারেটর কোড রিপো \(Sieve of Eratosthenes Algorithm\) | রবিউলের রাফখাতা](#)**Yeasintamim_sust**

নভেম্বর ২৪, ২০১৫ at ১০:৪৫ am

"৮ নম্বর লাইনে শুরুতেই ২ এর সব মাল্টিপল কেটে দিলাম"

this line is not true because we do it in the 6 th line

[^](#)
top

there is also some problem in following blocks of line

""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।

""

///because we find status[i] in the 8 th line

///then we don't start j from 2*i in the code rather we start it from j=i*i....

Reply



Spa Green Creative Studio

জানুয়ারি ১১, ২০১৬ at ৯:৫৪ am

Great Topic.Thanks to share.

Reply

Pingback: [শাফায়েতের ব্লগ » Blog Archive](#)



Falahun

সেপ্টেম্বর ১৫, ২০১৬ at ৫:০২ pm

Thanks alot !

Reply

Pingback: [Number Theory \(সংখ্যাতত্ত্ব \) | Site Title](#)

Leave a Reply

Connect with:

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Save my name,
email, and
website in this
browser for the
next time I

☐ comment.

Post Comment

phonetic

probhat

english

English Blog

সাবস্কাইব

আমার সম্পর্কে

শাফায়েত, সফটওয়্যার ইঞ্জিনিয়ার @ **Traveloka** Singapore (বিস্তারিত...)



Shafaetsplanet
1,856 likes

Like Page

শাফায়েত আশরাফ

Share

প্রোগ্রামিং কনটেন্ট এবং অ্যালগোরিদম

অনুপ্রেরণা(৪):

কেন আমি প্রোগ্রামিং শিখবো?

কম্পিউটার বিজ্ঞান কেন পড়বো?

প্রোগ্রামিং কনটেন্ট এবং অনলাইন জাজে হাতেখড়ি

কনফিউজড প্রোগ্রামার

অ্যালগরিদম বেসিক(৬):

বিগ "O" নোটেশন

কমপ্লেক্সিটি ক্লাস(P-NP, টুরিং মেশিন ইত্যাদি)

হাল্টিং প্রবলেম

বাইনারি সার্চ - ১

বাইনারি সার্চ - ২(বাইসেকশন)

ফ্লয়েড সাইকেল ফাইন্ডিং অ্যালগোরিদম

ডাটা স্ট্রাকচার(১১):

^
top

লিংকড লিস্ট

স্ট্যাক

কিউ+সার্কুলার কিউ

স্লাইডিং রেঞ্জ মিনিমাম কুয়েরি (ডিকিউ)

ডিসজয়েন্ট সেট(ইউনিয়ন ফাইন্ড)

ট্রাই(প্রিফিক্স ট্রি/রেডিক্স ট্রি)

সেগমেন্ট ট্রি-১

সেগমেন্ট ট্রি-২(লেজি প্রপাগেশন)

অ্যারে কমপ্রেসন/ম্যাপিং

লোয়েস্ট কমন অ্যানসেস্টর

বাইনারি ইনডেক্সড ট্রি

গ্রাফ থিওরি(২০):

গ্রাফ থিওরিতে হাতেখড়ি

অ্যাডজেসেন্সি ম্যাট্রিক্স

অ্যাডজেসেন্সি লিস্ট

ব্রেথড ফাস্ট সার্চ (বিএফএস)

মিনিমাম স্প্যানিং ট্রি ১ (প্রিমস অ্যালগোরিদম)

মিনিমাম স্প্যানিং ট্রি ২ (ক্রুসকাল অ্যালগোরিদম)

টপোলজিকাল সর্ট

ডেপথ ফাস্ট সার্চ এবং আবারো টপোলজিকাল সর্ট

ডায়াক্রিট

ফ্লয়েড ওয়ার্শল

বেলম্যান ফোর্ড

আর্টিকুলেশন পয়েন্ট এবং ব্রিজ

স্ট্রংলি কানেক্টেড কম্পোনেন্ট

ম্যাক্সিমাম ফ্লো-১

ম্যাক্সিমাম ফ্লো-২

স্টেবল ম্যারেজ প্রবলেম

অয়লার ট্যুর(নতুন)

মিনিমাম ভারটেক্স কভার

ট্রি এর ডায়ামিটার নির্ণয়

লংগেস্ট পাথ প্রবলেম

অ্যালগোরিদম গেম থিওরি(৩):

গেম থিওরি-১

গেম থিওরি-২

গেম থিওরি-৩

ডাইনামিক প্রোগ্রামিং(৮):

শুরুর কথা

ডিপি 'স্টেট', NcR, ০-১ ন্যাপস্যাক

কয়েন চেঞ্জ, রক ক্লাইম্বিং

ডিপি সলিউশন প্রিন্ট করা এবং LIS

বিটমাস্ক ডিপি

মিনিমাম ভারটেক্স কভার(গ্রাফ+ডিপি)

লংগেস্ট কমন সাবসিকোয়েন্স(LCS)

ম্যাট্রিক্স চেইন মাল্টিপ্লিকেশন

ব্যাকট্র্যাকিং(১):

ব্যাকট্র্যাকিং বেসিক এবং পারমুটেশন জেনারেটর

নাম্বার থিওরি/গণিত(৫):

মডুলার অ্যারিথমেটিক

প্রাইম জেনারেটর (Sieve of Eratosthenes)

বিটওয়াইজ সিভ

ডিরেঞ্জমেন্ট

প্রোবাবিলিটি: এক্সপেক্টেড ভ্যালু

স্ট্রিং ম্যাচিং(২):

রবিন-কার্প

কেএমপি (KMP)(নতুন)

অন্যান্য(৩) :

ডিরেকশন অ্যারে

মিট ইন দ্যা মিডল

টেইল-কল রিকার্সন অপটিমাইজেশন(নতুন)

কোয়ান্টাম কম্পিউটার(২)

কোয়ান্টাম কম্পিউটার কী?

কোয়ান্টাম কম্পিউটারের শক্তি এবং সীমাবদ্ধতা



Shafaetsplanet
1,856 likes

Like Page

শাফায়েত আশরাফ

Share

Be the first of your friends to like this



Shafaetsplanet

about 2 months ago

মিট ইন দ্যা মিডল এলিগেন্ট একটা প্রবলেম সলভিং টেকনিক। এটার কাজ হলো প্রবলেমটাকে ঠিক দুইভাগে ভাগ করে ফেলে তারপর সেই দুইভাগকে কোনোভাবে মার্জ করে প্রবলেমটা সলভ করা। তবে ডিভাইড এন্ড কনকোয়ারের সাথে এটার পার্থক্য হলো ডিভাইড এন্ড কনকোয়ারে দুই ভাগে ভাগ করার পর ছোট ভাগগুলোকে বারবার ভাগ করা হয়, মিট ইন দ্যা মিডলে আমরা শুধু একবার ভাগ করবো। বিস্তারিত ব্লগে।



নতুন লেখা

অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং: ইন্টারফেস এবং

পলিমর্ফিজম

টেইল-কল রিকার্সন অপটিমাইজেশন

ট্রাভেলোকা এবং আমার সফটওয়্যার ইঞ্জিনিয়ারিং এ

হাতেখড়ি

স্ট্রিং ম্যাচিং: নুথ-মরিসন-প্র্যাট (কেএমপি) অ্যালগরিদম

অয়লার ট্যুর (ফ্লিরি এবং হেয়ারহজলার অ্যালগরিদম)



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0

top

International License.

Disclaimer: The advertisement shown in this site is automatically inserted by Google AdSense based on individual user's interest. It doesn't reflect the interest or ideology of the site owner.

AccessPress Staple | WordPress Theme: AccessPress Staple by AccessPress Themes

