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,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.
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:

$n=p_1*p_2*p_3....*p_kn=p1*p2*p3....*pk$

nn যদি নিজেই প্রাইম হয় তাহলে $n=p_1(=n)n=p1(=n)$। অন্যথায় অবশ্যই একাধিক প্রাইম ফ্যাক্টর থাকতে হবে। এবার চিন্তা করো কোনো সংখ্যা cc কে দুটি সংখ্যার গুণফল $c=a*bc=a*b$ হিসাবে লিখলে aa আর bb এর একটি অবশ্যই সংখ্যাটির বর্গমূলের থেকে ছোট,অন্যটি

বড়। aa আর bb দুটো সংখ্যাই cc এর বর্গমূলের থেকে বড় হলে গুণফল cc থেকে বড় হতো (ঠিক যেমন $c=a+bc=a+b$ হলে aa বা bb এর একটি cc এর অর্ধেকের থেকে ছোট অন্যটি বড়)।

এবার $n=p_1*p_2*p_3....*p_kn=p1*p2*p3....*pk$ তে ফিরে আসি। $p_1,p_2,p_3p1,p2,p3$ ইত্যাদির মধ্যে যে কোনো ২টি যদি nn এর বর্গমূল থেকে বড় হয় তাহলে তাদের গুণফল nn 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[1000000 + 1];

void siv(int N) {
    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) {
```

```
12          for (int j = i * i; j <= N; j += i)
13              status[j] = 1;
14      }
15    }
16
17    status[1] = 1;
18 }
```

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।