অ্যালগরিদম - Algorithms সংখ্যাতত্ত্ব - Number theory

সংখ্যাতত্ত্ব: মডুলার অ্যারিথমেটিক **(Modular arithmetic) – Big mod**

Number theory: Modular arithmetic and modular inverse and their algorithms - Bangla tutorial [BigMod]

Sharif Hasan • December 17, 2020 সর্বশেষ আপডেট October 10, 2021 2 2,455 পড়তে 4 মিনিট লাগতে পারে

১০০! এর মধ্যে কয়টা ডিজিট আছে? হিসাব করলে দেখা যায় ১৫৮ টির মতো। বলা হলো আপনাকে ১০০! ফাক্টরিয়াল বের করে তার আউটপুট কে ৯৭ দিয়ে ভাগ করে তার ভাগশেষ কে প্রিন্ট করতে হবে। এখন কি আমরা কোনোভাবে অভারফ্লো (Overflow) এড়িয়ে গিয়ে সমধান করতে পারি? ১৫৮ ডিজিটের কোনও সংখ্যাতো ৬৪ বিট আনসাইনড এও ধরবে না। কিন্তু আমরা মডুলার অ্যারিথমেটিক **(Modular arithmetic)** এর সূত্র ব্যবহার করে এই ধরনের সমস্যা সমাধান করতে পারি।

আগের পোস্ট টির লিঙ্ক এখানে সংখ্যাতত্ত্ব: মৌলিক সংখ্যা (prime number) ও তার অ্যালগরিদম

> *আধুনিক মডুলার অ্যারিথমেটিক (Modular arithmetic) জনক হলেন জার্মান গণিতবিদ কার্ল ফ্রেডরিক গাউস।*

মডুলার অ্যারিথমেটিক **(Modular arithmetic)**

মডুলার অ্যারিথমেটিকে (Modular arithmetic) চলক একটা নির্দিষ্ট সংখ্যায় পোঁছানোর পর আবার ০ থেকে রিপিট হয়। উদাহরণে বুঝা যাক,

একটি কাটা ঘরির কথা ভাবি। যেখানে ঘড়িটি ১ টা থেকে ১২ টা পর্যন্ত সময় দেখাতে পারে। ধরি এখন ৭ টা বাজে। এর ৮ ঘণ্টা পরে ৩ টা বাজবে। আমরা যোগ করে পাই, ৭+৮=১৫, কিন্তু যেহেতু ঘড়িটি প্রত্যেক ১২ ঘণ্টা পরে পরে আবার আগের অবস্থানে আসে, তাই আমাদের ঘড়িতে (৭+৮)%১২ বা ৩ টা বেজেছে। আশা করি বুঝা গিয়েছে কি হচ্ছে। নিজ হাতে কাটা ঘড়ি থাকলে ঘুরিয়ে দেখা যেতে পারে। 🙂

বেশিরভাগ প্রোগ্রামিং ল্যাঙ্গুয়েজ (programming language) গুলোতে % অপারেটর দিয়ে ভাগশেষ বুঝানো হয়। x কে m দিয়ে ভাগ করার পর ভাগশেষ প্রোগ্রামিং এ $x\%m$ এর মান বের করা। একে **x mod m** পড়া হয়। যেহেতু ১০০! অনেক বড় সংখ্যা, তাই ধরে

নেই ১০০!=x। অর্থাৎ x বা ১০০! ফাক্টরিয়াল কে m বা ৯৭ দিয়ে ভাগ করে ভাগশেষ প্রিন্ট করাই আমাদের মুল সমস্যা।

উপরে যা বললাম, আমরা ১০০! বের করতে পারবো না। এটা অনেক বড় সংখ্যা, তবে আমরা ৯৭ দিয়ে ভাগ করে ভাগশেষ বের করতে পারি। এটা int ডাটা টাইপেই ধরে যাবে। যাইহোক, এ ধরনের সমস্যা সমাধানে আমরা নিচের দুটি সূত্রের সাহায্য নিবো। প্রথমে সূত্রগুলোতে চোখবুলাই একটু,

$(a+b)\%m = ((a\%m)+(b\%m))\%m \ldots\ldots$ (১)
$(a\times b)\%m = ((a\%m)\times(b\%m))\%m \ldots\ldots$ (২)

n সংখ্যক সংখ্যা $a_1, a_2, \ldots, a_n$ এর জন্য সুত্র দুটি ব্যবহার করতে পারবো। উপরের সমস্যা সমাধানে আমাদের ২য় সূত্রটি কাজে লাগবে। অর্থাৎ $(1\times 2\times 3\times \ldots 100)$ সমীকরণের বামপক্ষ ধরে এখন সমাধান করতে হবে। এভাবে করলে আমাদের ওভারফ্লো করবে না। কারণ প্রতিটি ধাপে গুণফলকে ৯৭ দ্বারা mod করা হবে।

নিচের C++ কোডটি দেখি,

C++

```
int big_factorial(int x,int m){
    int fact=1;
    for(int i=1;i<=x;++i){
        fact=((fact%m)*(i%m))%m;
    }
    return fact;
}
```

১০০! এর জন্য এর আউটপুট হবে ০। কারণ ১০০! কে ৯৭ নিঃশেষে ভাগ করে। এখানে দেখা যাচ্ছে আমরা লুপ এর ভিতরে কাজ করেছি দুটি করে সংখ্যা নিয়ে। একটু লক্ষ করলেই আশা করি বুঝা যাবে।

সূত্র দুটি কেন কাজ করে?

সূত্র দুটি কেন কাজ করে টা আমাদের জানা দরকার। এর জন্য আমরা প্রমাণ করার চেষ্টা করতে পারি।

এখন ধরি q_1 এবং q_2 দুটি সংখ্যা যা a এবং b কে m দিয়ে ভাগ করার পরে আমাদের ভাগফল। অর্থাৎ $q_1 = \lfloor\frac{a}{m}\rfloor, q_2 = \lfloor\frac{b}{m}\rfloor$ এবং c_1, c_2 হচ্ছে আরও দুটি সংখ্যা যা যথাক্রমে a এবং b কে m দিয়ে ভাগ করার পরে ভাগশেষ হিসেবে পাওয়া যায়। অর্থাৎ $a\%m = c_1, b\%m = c_2$।

তাহলে আমরা বলতে পারি,

$a = q_1 \times m + c_1$
$b = q_2 \times m + c_2$

a, b এর মান বসিয়ে পাই,
$(a + b) \ \% \ m = (q_1 \times m + c_1 + q_2 \times m + c_2) \ \% \ m$

তাই (১) সমীকরণের বামপক্ষ থেকে লিখা যায়,

$(a + b) \ \% \ m$
$(q_1 \times m + c_1 + q_2 \times m + c_2) \ \% \ m$
$= (m(q_1 + q_2) + c_1 + c_2) \ \% \ m$

ধরি, $(q_1 + q_2) = Q$ এবং $c_1 + c_2 = C$, তাহলে

$= (m.Q + C) \ \% \ m$
$= C \ \% \ m$

এখানে, $m.Q$ স্পষ্ট ভাবেই m এর গুণিতক, সুতরাং আমাদের উত্তর $C\%m$, C কে আবার mod করলাম কারণ $c_1 + c_2 >= m$ হতেই পারে।

আবার (১) নং সমীকরণের ডানপক্ষ থেকে পাই,

$(a \ \% \ m + b \ \% \ m)\% \ m$
$= ((q_1 \times m + c_1) \ \% \ m + (q_2 \times m + c_2) \ \% \ m)\% \ m$

এখন, $(q_1 \times m + c_1) \ \% \ m = c_1$ এবং $(q_2 \times m + c_2) \ \% \ m = c_2$ তাই,

$= (c_1 + c_2)\%m$
$= C \ \% \ m$

সুতরাং $L.H.S. = R.H.S.$ প্রমাণ করা হলো। একই ভাবে ২ নং সূত্রটিও প্রমাণ করা যাবে।

ঋণাত্মক সংখ্যার **mod (Mod of negative numbers)**

Negative বা ঋণাত্মক সংখ্যার mod বের করতে হলে আমরা সরাসরি % অপারেটর ব্যবহার করতে পারি না। যেমন -১৭ কে ৫ দ্বারা mod করলে সি তে উত্তর আশে -২। ভাগশেষের সংজ্ঞানুযায়ী,

> *m এর সবথেকে বড় থেকে বড় মাল্টিপল যেটা x এর থেকে ছোট সেই সংখ্যাটিকে x থেকে বিয়োগ করলে যে সংখ্যাটি পাওয়া যায় সেটাই ভাগশেষ c.*

এখানের উদাহরণে ৫ এর সবচেয়ে বড়ো গুণিতক বা মাল্টিপল যেটা -১৭ থেকে ছোট টা হলো -২০। তাই সংজ্ঞানুযায়ী আমাদের উত্তর আসার কথা -১৭-(-২০)=৩। তাই প্রোগ্রামিং কন্টেস্ট গুলোতে ঋণাত্মক সংখ্যা নিয়ে সতর্ক না থাকলে অল্পেতেই Wrong answer (WA) খেতে হতে পারে। তাই আমরা এটা সমাধানের জন্য যা করবো তা হলো, **x** এর সাথে **m** এর এমন একটি মাল্টিপল যোগ করবো, যেন যোগফল ধনাত্মক হয়। যেমন, x=১৭ এবং m=৫ এর জন্য

> ***-১৭%৫=(-১৭+১০০)%৫= ৮৩%৫=৩***

মডুলার অ্যারিথমেটিক **(Modular arithmetic): Big mod** সমস্যা

ধরা যাক আমাদের ৩ টি সংখ্যা দেয়া আছে, a,b,m । এখন আমাদের $(a^b)\%m$ বের করতে হবে। আমরা ভাবতেই পারি উপরের ২ নং সূত্র দিয়ে কাজটি করা যাবে ফাক্টরিয়াল এর মতো করে। হ্যাঁ করা যাবে। তবে সমস্যা হল যখন b এর মান অনেক বড় হবে। $(2^{2000000000})\%10$ ওই ভাবে বের করতে প্রচুর সময় লাগবে। তবে আমরা এই সমস্যাটিও সহজে(!) $O(log_2 n)$ এ করতে পারি।

এই সমস্যা সমাধানের জন্য আমরা Recursion এর সাহায্য নিবো। আগে আমরা আমাদের কোডটি দেখে নিই।

```cpp
1 int big_mod(int a,int b,int m){
2     if(b==0) return 1%m;
3     int x=big_mod(a,b/2,m);
4     x=(x*x)%m;
5     if(b%2==1) x=(x*a)%m;
6     return x;
7 }
```

ধরি আমাদেরকে $(2^{100})\%10$ বের করতে বলা হয়েছে। এটা সমাধান করতে আমাদেরকে ১০০টি ২ কে গুন করতে হবে না। আমরা আমাদের সূচক ১০০ কে ভেঙ্গে $2^{50}\%10$ এ রূপান্তর করবো। ধরি এটি $(a^b)\%m$ = x, যেখানে শুরুতে a=2,b=100,m=10

$(2^{100})\%10$

$=(2^{50}\times2^{50})\%10$ …….. প্রোগ্রাম এর ৩ নং লাইন।

$=(2^{50}\%10\times2^{50}\%10)\%10$ ……. ২ নং সূত্র থেকে।

=(x.x)%m ………. প্রোগ্রামের ৪ নং লাইন।

$(2^{50})\%10$
$=(2^{25}\times2^{25})\%10$
$=(2^{25}\%10\times2^{25}\%10)\%10$
$=(x.x)\%m$

এখন সমস্যা হলো যখন আমাদের সূচক বিজোড় হবে। যেমন এর পরে যখন x কে আবার ভাঙবো তখন, 2^{25} পাবো। তখন আমরা তো সূচককে সমান দুইভাগে ভাগ করতে পারবো না। তাতে আমাদের কি? আমরা একে নিচের মতো প্রসেস করবো।

$(2^{25})\%10$
$=((2^{12}\times2^{12})\%10 \times 2)\%10$
$=((2^{12}\%10\times2^{12}\%10)\%10 \times 2)\%10$
$=((x.x)\%10 \times a)\%m$

এখানে একটা টেকনিক করলাম। $2^{12}\times2^{12}=2^{24}$ এর সাথে 2 গুন করার পরে আমরা আবার 2^{25} ফিরে পাই। যা আমরা প্রোগ্রামের ৫ নং লাইনে করেছি।

`বিগ মড (Big Mod) এলগরিদমের রানটাইম হলো $O(log_2 n)$। কারণ প্রতিবার আমাদের n এর মান দুইভাগ হচ্ছে। তার মানে $n = 2^k$ হলে আমাদেরকে k সংখ্যক বার রিকার্সিভ কল করতে হবে। বা, $log_2 2^k = k$। এই ধরেনের সমস্যা সমাধান পদ্ধতিকে devide and conquer বলা হয়। নিচের ছবিটা কিছুটা হেল্প করতে পারে আরেকটু বুঝতে।

উপরের ছবিতে আমরা প্রথম 2^{100} কে ভেঙে ভেঙে উপর থেকে নিচের দিকে গিয়েছি, বেস কেস b=0 তে যাওয়ার পরে নিচ থেকে উপরে উঠেছি। পথের মধ্যে গুনের কাজগুলো শেষ করেছি। লক্ষণীয় যে b বিজোড় হলে আমরা নিচে গিয়েছি $b = \lfloor \frac{b}{2} \rfloor$ হিসেবে। বাকি একটা a কে আমরা পরে আলাদা করে গুন করে দিয়েছি। এতে করে আমাদের মান সমান থেকেছে।

নাম্বার থিউরি নিয়ে আরও জানতে পড়ুন: সংখ্যাতত্ত্ব: মৌলিক সংখ্যা ও তার অ্যালগরিদম [Algorithms] [C++] সংখ্যাতত্ত্ব (১): সংখ্যাতত্ত্বের প্রাথমিক আলোচনা ও বিভাজ্যতার নীতি

Modular multiplicative inverse নিয়ে লিখার ইচ্ছা হয়েছিল। কিন্তু কিছুতেই তেল খুঁজে পেলাম না সমীকরণ লিখতে গিয়ে। অন্য কোন সময় লিখব। #Happy_coding

লেখাটি কেমন লেগেছে আপনার?

রেটিং দিতে হার্টের উপর ক্লিক করুন।

গড় রেটিং 4.4 / 5. মোট ভোট: 48

#অ্যালগরিদম #মডুলার অ্যারিথমেটিক #সংখ্যাতত্ত্ব

2 টি মন্তব্য

Mahmudul Hasan

July 9, 2022 at 1:58 AM

Effective writing ❤ . I tried to learn from many other source but this is the best ❤ . Now I can do it. Thank you vaiya ❤ .

Reply

Sharif Hasan

July 9, 2022 at 2:08 AM

ধন্যবাদ ভাইয়া ❤️

Reply