শাফায়েতের ব্লগ

ভাগশেষের গণিত(মডুলার অ্যারিথমেটিক + বিগ মড)

২০১২-০১-২০ ১৪:০১:২২ শাফায়েত

-১৭ কে ৫ দিয়ে ভাগ করলে ভাগশেষ কত হয়? ২^১০০০ কে ১৭ দিয়ে ভাগ করলে ভাগশেষ কত হয় সেটা কি তুমি integer overflow এড়িয়ে নির্ণয় করতে পারবে? O(n) এ পারলে O(log n) কমপ্লেক্সিটিতে পারবে? যদি কোনো একটি উত্তর “না” হয় তাহলে এই পোস্ট তোমার জন্য। তবে তুমি যদি মডুলার ইনভার্স বা এডভান্সড কিছু শিখতে পোস্টটি খুলো তাহলে তোমাকে আপাতত হতাশ করতে হচ্ছে।

সি/জাভা সহ বেশিভাগ প্রোগ্রামিং ল্যাংগুয়েজে এ % কে ভাগশেষ অপারেটর ধরা হয়। x কে m দিয়ে ভাগকরে ভাগশেষ বের করার অর্থ x%m এর মান বের করা অথবা আমরা বলতে পারি x কে m দিয়ে mod করা। “determine answer modulo 1000″ এ কথাটির অর্থ হলো উত্তরকে ১০০০ দিয়ে mod করে তারপর আউটপুট দিতে হবে।

একটি সমস্যা দিয়ে শুরু করি। তোমার ১০০টি বই আছে,তুমি কয়ভাবে বইগুলো সাজাতে পারবে? খুব সহজ,১০০! (১০০ ফ্যাক্টরিয়াল) ভাবে সাজাতে পারবে। ১০০! ১৫৮ ডিজিটের বিশাল একটি সংখ্যা। তাই আমি তোমাকে প্রবলেমটা সহজ করে দিলাম,ধরো তুমি x উপায়ে বইগুলো সাজাতে পারবে,তাহলে তোমাকে x%97 কত সেটা বলতে হবে। অর্থাত ১০০! বের করে ৯৭ দিয়ে ভাগ করে ভাগশেষটা বের করাই তোমার সমস্যা। (Determine 100 factorial modulo 97)

এটা কিভাবে করবে? ১০০! এর মান তুমি বের করতে পারবেনা ৬৪বিট আনসাইনড ইন্টিজার দিয়েও, এরা ২^৬৪-১ পর্যন্ত সংখ্যা নিয়ে কাজ করতে পারে, তাই ওভারফ্লো হবে। কিন্তু আমরা জানি আমাদের উত্তর কখনোই 97 এর বড় হবেনা কারণ **কোনো সংখ্যাকে m দিয়ে mod করা হলে সংখ্যাটি m এর থেকে বড় হতে পারবেনা**।

আমরা এ ধরণের সমস্যা সমাধান করতে সাহায্য নিবো দুটি সুত্রের:

> **(a+b)%m=((a%m)+(b%m))%m**
> **(a*b)%m=((a%m)*(b%m))%m**

n সংখ্যক নম্বর a1,a2…an এর জন্য সুত্র দুটি ব্যবহার করতে পারবে।

উপরের সমস্যাটিতে ২য় সুত্রটি লাগবে। তোমার বের করা দরকার ১০০! %৯৭ অর্থাত:
(১০০*৯৯*৯৮*………..*১)%৯৭
তুমি যেটা করবে সেটা হলো গুণ করার সময় ২য় সুত্রের মত করে mod করতে থাকবে,তাহলে কোনো সময়ই overflow ঘটবেনা কারণ mod করলে প্রতি স্টেপে সংখ্যাটি ছোটো হয়ে যাচ্ছে। এটার কোড হতে পারে এরকম:

```
int fact=1;
 for(int i=1;i<=100;i++)
 {
  fact=((fact%97)*(i%97))%97;

 }
 printf("%d\n",fact);
```

এটার আউটপুট আসবে ০। অর্থাত ১০০! % ৯৭ =০। একটু খেয়াল করলেই বুঝবে এখানে আমরা ২য়

সুত্রটি প্রয়োগ করেছি ২টি করে সংখ্যা নিয়ে।

সুত্র দুটি কেনো কাজ করে সেটা জানা দরকার। আমি ১ম সুত্রটির প্রমাণ দেখাচ্ছি,২য়টিও একইভাবে করা যায়। প্রমানটি আমার নিজের মত করে করা।

ধরি (x+y)%৫ এর মান আমাদের বের করতে হবে। এখন যদি x%5=c1 আর y%5 = c2 হয়,তাহলে x কে আমরা লিখতে পারি 5n1+c1 এবং y কে লিখতে পারি 5n2+c2 যেখানে n1 আর n2 দুটি ইন্টিজার। এটা একদম বেসিক রুল,আশা করে বুঝতে সমস্যা হচ্ছেনা। এখন:

> (x+y)%5
> = (5n1+c1+5n2+c2)%5
> =(5n1+5n2+c1+c2)%5 ——(১)

এখানে 5n1+5n2 অবশ্যই 5 এর মাল্টিপল,তাই আমরা লিখতে পারি
5n1+5n2=5N যেখানে N=n1+n2
এবং c1+c2=C
তাহলে (১) থেকে পাচ্ছি:
(5N+C)%5
এখন পরিস্কার বোঝা যাচ্ছে যে উত্তর হলো C%5। C কে আবার mod করতে হলো কারণ c1+c2 এর মান 5 এর থেকে বড় হতেই পারে। এখন

> ((x%5)+(y%5))%5———(২)
> =((5n1+c1)%5)+((5n2+c2)%5))%5
> (5n1+c1)%5=c1
> (5n2+c2)%5=c2

তাহলে ২ কে লিখতে পারি:
(c1+c2)%5 = C%5

তাহলে ১ম সুত্রটি প্রমাণিত হলো। তারমান যোগ করে mod করা আর আগে mod করে তারপর যোগ করে আবার mod করা একই কথা। সুবিধা হলে সংখ্যাটি কোনো স্টেপেই বেশি বড় হতে পারেনা। গুণের ক্ষেত্রেই একই সুত্র প্রযোজ্য।

নেগেটিভ সংখ্যার mod নিয়ে একটু আলাদা ভাবে কাজ করতে হয়। সি তে -17 % 5 এর মান দেখায় -২। কিন্তু সচরাচর আমরা ভাগশেষের যে সংজ্ঞা ব্যবহার করি তাতে x%m = p হলে গাণিতিকভাবে

> m এর সবথেকে বড় থেকে বড় মাল্টিপল যেটা x এর থেকে ছোট সেই সংখ্যাটিকে x থেকে বিয়োগ করলে যে সংখ্যাটি পাওয়া যায় সেটাই p।

যেমন 23 % 5 এর ক্ষেত্রে ৫*৪=২০ হলো ৫ এর সবথেকে বড় মাল্টিপল যেটা ২৩ এর থেকে ছোট,তাই 23 % 5=23-(5*4)=৩। -17 % 5 এর ক্ষেত্র খেয়াল করো -20 হলো ৫ এর সবথেকে বড় মাল্টিপল যেটে -১৭ থেকে ছোট,তাই উত্তর হবে ৩।
এই কেসটা handle করা একটি উপায় হলো নেগেটিভ সংখ্যাটিকে একটি 5 এর মাল্টিপল এর সাথে যোগ করা যেন সংখ্যাটি ০ থেকে বড় হয়ে যায়,তারপরে mod করা। যেমন:

> -17%5
> =(-17+100)%5
> =83%5
> =3

এটা উপরের সুত্রের প্রমাণের মত করেই কাজ করে,একটু গুতালেই প্রমাণ করতে পারবে। **নেগেটিভ সংখ্যার mod নিয়ে কনটেস্টে সবসময় সতর্ক থাকবে,এটা wrong answer খাওয়ার একটা বড় কারণ হতে পারে।**

এবার আসি সুপরিচিত big mod সমস্যায়। সমস্যাটি হলো তোমাকে (a^b)%m এর মান বের করতে হবে,a,b,m তোমাকে বলে দেয়া হবে,সবগুলোর range 2^31 পর্যন্ত হতে পারে। ১০০! % ৯৭ বের করার মত করে সহজেই তুমি overflow না খেয়ে মানটি বের করতে পারবে,সমস্যা হলো তুমি লুপ চালিয়ে একটি একটি গুণ করে (2^(2000000000))%101 বের করতে চাইলে উত্তর পেতে পেতে সম্ভবত নাস্তা শেষ করে আসতে পারবে। আমরা চাইলে O(lgn) এ এটা করতে পারি।
লক্ষ করো

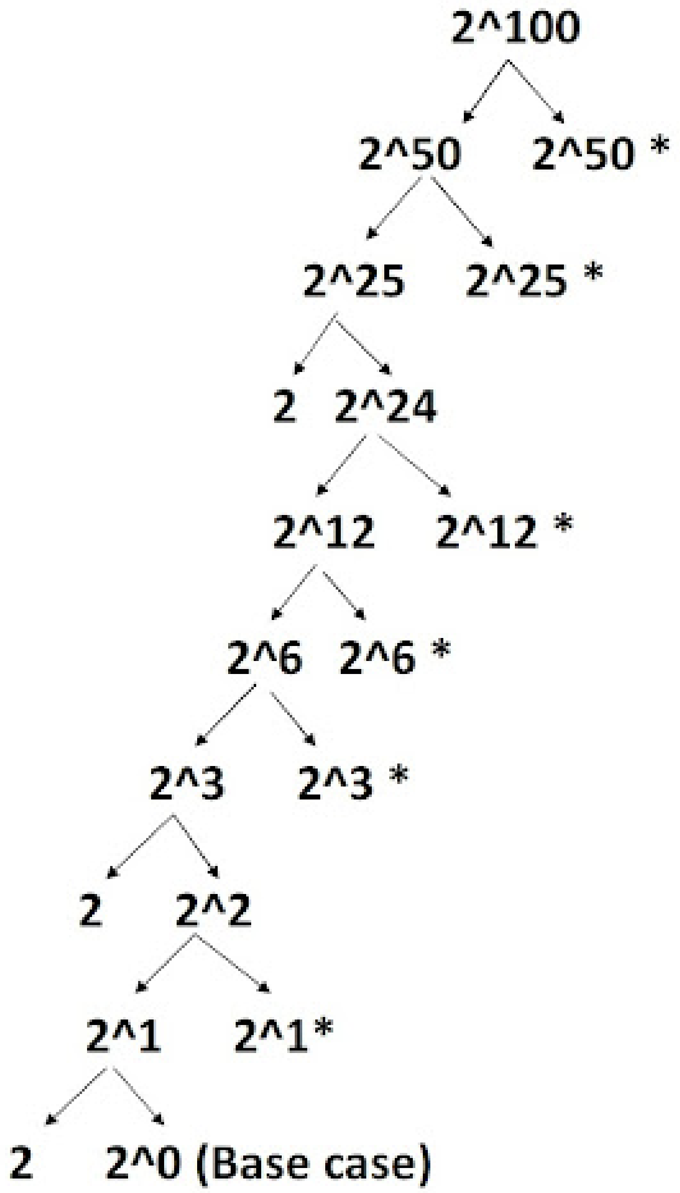
2^100
=(2^50)^2
এবং
(2^50)
=(2^25)^2

এখন বলো 2^50 বের করতে কি 2^26,2^27 ইত্যাদি বের করার দরকার আছে নাকি 2^25 পর্যন্ত বের করে square করে দিলেই হচ্ছে? আবার 2^25 পর্যন্ত আসতে (2^12)^2 পর্যন্ত বের করে square করে সাথে ২ গুণ করে দিলেই যথেষ্ট,অতিরিক্ত ২ গুণ করছি সংখ্যাটি বিজোড় সে কারণে। প্রতি স্টেপে গুণ করার সময় mod করতে থাকবে যাতে overflow না হয়। recursion ব্যবহার করে কোডটি লেখা জলের মত সোজা:

```
#define i64 long long
i64 M;
i64 F(i64 N,i64 P)
{
 if(P==0) return 1;
 if(P%2==0)
 {
  i64 ret=F(N,P/2);
  return ((ret%M)*(ret%M))%M;
 }
 else return ((N%M)*(F(N,P-1)%M))%M;

}
```

মন্তব্য অংশে “হাসান” একটি বিগ মডের সুন্দর রিকার্শন-ট্রি এর ছবির লিংক দিয়েছে, ছবিটা এরকম:



মডুলার অ্যারিথমেটিক ব্যবহার করে বিশাল আকারের ফলাফল কে আমরা ছোট করে আনতে পারি ফলাফলে বিভিন্ন characteristic নষ্ট না করে,তাই এটা গণিতে খুব গুরুত্বপূর্ণ। প্রোগ্রামিং কনটেস্টে প্রায়ই বিভিন্ন প্রবলেমে মডুলার অ্যারিথমেটিক প্রয়োজন পড়বে,বিশেষ করে counting আর combinatorics এ যেখানে ফলাফল অনেক বড় হতে পারে,ফ্যাক্টরিয়াল নিয়ে কাজ করতে হতে পারে।

ভাগ করার সময় গুণ,আর যোগের মত সুত্র দুটি কাজ করেনা,এটার জন্য তোমাকে extended euclid আর modular inverse জানতে হবে।

সিপিউর জন্য mod খুব costly একটা অপারেশন। **যোগ,গুণের থেকে mod করতে অনেক বেশি সময় লাগে ।** অপ্রয়োজনে mod ব্যবহার করলে কোড time limit exceed করতে পারে,তাই overflow হবার আশংকা না থাকলে সব জায়গায় mod করা দরকার নেই। আমার একটি কোড ৩সেকেন্ডে time limit exceed হবার পর খালি কিছু mod সরিয়ে ১.৩ সেকেন্ড নামিয়ে এনেছি।

এখন চিন্তা করার জন্য একটি প্রবলেম। ধরো তোমাকে একটি অনেক বড় সংখ্যা(bigint) দিয়ে সেটাকে ২^৩১ এর ছোট একটি সংখ্যা দিয়ে mod করতে বলা হলো। O(length_of_bigint) কমপ্লেক্সিটিতে কিভাবে করবে?
সাহায্য:

২৩=(০*১০+২)*১০ + ৩
১২৩৯=(((০*১০+১)*১০ + ২)*১০ +৩)*১০+৯

প্র্যাকটিসের জন্য প্রবলেম:
http://uva.onlinejudge.org/external/3/374.html
http://uva.onlinejudge.org/external/101/10127.html

ফেসবুকে মন্তব্য