

Tushar's Blog

Home Basic Data Structure Number Theory Discuss Books Links Others About

মঙ্গলবার, ২৪ ফেব্রুয়ারী, ২০১৫

বিগ-ইন্টিজার (জাভা)

আমরা বিভিন্ন প্রোগ্রামিং ভাষা গুলো তে হিসাব-নিকাশ করার জন্য সাধারণত int (32-bit) বা long (64-bit) টাইপের ডাটা ব্যবহার করে থাকি। এদের মধ্যে int ডাটা দিয়ে খুব বড় নাম্বার গুলোর হিসাব-নিকাশ করা না গেলেও long ডাটা দিয়ে মোটামুটি 10^{18} এর কাছাকাছি সংখ্যাগুলো এক্সেস করা যায়। সমস্যা হয় তখন যখন আমাদেরকে এর চেয়েও অনেক অনেক গুণ বড় সংখ্যা গুলো এক্সেস করার দরকার হয়। তখন আমরা কোন সাধারান ডাটা দিয়ে সরাসরি এদের হিসাব-নিকাশ করতে পারি না, কারন এতে করে আমাদের ভেরিয়েবল গুলো তে ডাটার ওভারফ্লো হয়।

এধরণের সমস্যা দূর করার জন্য Java তে BigInteger নামের একটি বিল্ট-ইন ক্লাস তৈরি করা আছে যা দিয়ে খুব সহজেই অনেক বড় বড় সংখ্যার হিসাব-নিকাশ করে ফেলা যায় BigInteger এর সাইজ JVM এর পর্থাপ্ত মেমোরির উপর নির্ভর করে। তবে প্রয়োজন ছাড়া এটি ব্যবহার করা উচিত নয়, কারণ BigInteger সাধারান ডাটা-টাইপ এর তুলনায় অনেক ধীরে কাজ করে। এখন তাহলে দেখা যাক বিগ ইন্টিজার ব্যবহার করে আমরা কি কি কাজ এবং কিভাবে করতে পারি-

প্যাকেজ ইম্পোর্ট করাঃ

BigInteger ক্লাস টি জাভার math প্যাকেজের অন্তর্ভুক্ত। তাই BigInteger ব্যবহার করার জন্য আমাদেরকে প্রথমেই math প্যাকেজ টি ইম্পোর্ট করে নিতে হবে।

```
import java.math.BigInteger;
```

ভেরিয়েবল ডিক্লেয়ার করাঃ

বিগ ইন্টিজার ভেরিয়েবল ডিক্লেয়ার করা অন্যসব ডাটা টাইপের মতই। এটি মূলত BigInteger এর অবজেক্ট তৈরি করা হয়। যেমন, আমরা যদি a, b এবং c নামের তিনটি ভেরিয়েবল ডিক্লেয়ার করতে চাই তাহলে আমাদের এভাবে লিখতে হবে-

```
BigInteger a, b, c;
```

ইনপুট/ আউটপুটঃ

BigInteger এর ইনপুট/ আউটপুট ও অন্য সব ডাটা টাইপের মতই। যেমন, আমরা যদি একটি ভেরিয়েবলের মান ইনপুট নিয়ে তা আউটপুটে দেখাতে চাই তাহলে আমাদের প্রোগ্রাম টি এভাবে লিখতে হবে-

```
1. import java.util.*;
2. import java.math.*;
3. public class Main {
4.     public static void main(String[] args) {
5.         Scanner sc = new Scanner (System.in);
6.         BigInteger n;
7.
8.         n = sc.nextBigInteger();
9.         System.out.println(n);
10.    }
11. }
```

ড্যাুল অ্যাসাইন করাঃ

আমাদেরকে অনেক সময় এক ভেরিয়েবলের মান অন্য ভেরিয়েবলে অথবা সরাসরি কোন মান একটি ভেরিয়েবলে

Search Posts

 Search

Total Page views

 6,549

Follow by Email

 Submit

গুরুত্বপূর্ণ সংযোগ

- ▼ 2015 (4)
 - ▼ February (4)
 - [বিগ-ইন্টিজার \(জাভা\)](#)
 - [Download PDF Books of System Analysis](#)
 - [Download PDF Books of Assembly](#)
 - [Download PDF Books of Automata](#)

► 2014 (36)

► 2013 (3)

আমার সম্পর্কে



 Tushar Roy
আমার সম্পূর্ণ
প্রোফাইল দেখুন

অ্যাসাইন করার প্রয়োজন হতে পারে। একভেরিয়েবল থেকে অন্য ভেরিয়েবলে ভ্যালু অ্যাসাইন করতে আমরা সরাসরি ‘=’ ব্যবহার করতে পারি। যেমন, একটি ভেরিয়েবল **b** এর মান অপর একটি ভেরিয়েবল **a** তে অ্যাসাইন করতে চাইলে হলে লিখতে হবে-

```
BigInteger a = b;
```

কিন্তু সরাসরি যদি কোন মান একটি ভেরিয়েবলে অ্যাসাইন করতে চাই, তাহলে `BigInteger` এর কন্সট্রাক্টর ব্যবহার করে ভেরিয়েবলে (`BigInteger` অবজেক্ট) তার মান অ্যাসাইন করতে হবে। `BigInteger` এর কন্সট্রাক্টর টি প্যারামিটার হিসেবে একটি নাম্বারের স্ট্রিং নিয়ে থাকে যেমন-

```
BigInteger n = new BigInteger("1234567890123456789");
```

আবার আমরা `BigInteger.valueOf(int_value)` ব্যবহার করে যেকোনো ইন্টিজার রেঞ্জের সংখ্যা কে সরাসরি `BigInteger` হিসেবে ব্যবহার করতে পারি।

BigInteger অ্যারেঃ

`BigInteger` এর আবার অ্যারে! ? হ্যাঁ। অর্থাৎ হবার কিছু নেই @ বিগ-ইন্টিজারের অ্যারে ও আমরা ব্যবহার করতে পারি। এটিও অন্য সব ডাটা টাইপের অ্যারে এর মতই।

```
BigInteger[] array_name = new BigInteger[array_size];
```

এখন আমরা চাইলে এই অ্যারে টাকে সেটি ও করে ফেলতে পারি @ সম্পূর্ণ অ্যারে সেটি করতে চাইলে লিখতে হবে `Arrays.sort(array_name)`; আর কোন একটা নির্দিষ্ট রেঞ্জের ভ্যালু গুলো সেটি করতে চাইলে লিখতে হবে `Arrays.sort(array_name, fromIndex, toIndex+1)`; সেটিকরার ক্ষেত্রে খেয়াল রাখতে হবে যাতে অ্যারে টির কোন ইন্ডেক্সে নাল ভ্যালু না থাকে, তা নাহলে আমাদের প্রোগ্রাম টি এক্সেপশন থ্রো করবে।

BigInteger এর মেথডঃ

জাভার `BigInteger` ক্লাসে অনেক গুলো মেথড আছে, যা ব্যবহার করে আমরা বিগ-ইন্টিজার দিয়ে অনেক বড় বড় সমস্যা খুব সহজেই সমাধান করে ফেলতে পারি। আমি এখানে প্রয়োজনীয় কিছু মেথড নিয়ে আলচনা করছি।

যোগ/বিয়োগ/গুণ/ভাগ করাঃ

দুটি বিগ-ইন্টিজার যোগ করার জন্য আমাদেরকে `add()` মেথড টি ব্যবহার করতে হবে। যেমন ধরো আমরা **a** এবং **b** দুটি সংখ্যা যোগ করে **ans** নামের একটি ভেরিয়েবল রাখতে চাই। তাহলে আমাদের লিখতে হবে-

```
BigInteger ans = a.add(b);
```

একই ভাবে বিয়োগ করার জন্য `subtract()`, গুণ করার জন্য `multiply()`, এবং ভাগ করার জন্য আমাদেরকে `divide()` মেথড ব্যবহার করতে হবে।

ভাগশেষ বের করাঃ

একটি সংখ্যা কে অপর একটি সংখ্যা দিয়ে ভাগ করার পর অবশিষ্ট ভাগশেষ বা `mod` ভ্যালু বের করতে চাইলে আমাদেরকে `mod()` মেথড টি ব্যবহার করতে হবে। যেমন, আমরা যদি একটি সংখ্যা **a** কে অপর একটি সংখ্যা **m** দিয়ে `mod` করতে চাই তাহলে আমাদের লিখতে হবে-

```
BigInteger ans = a.mod(m);
```

এছাড়াও `remainder()` মেথড টি দিয়ে একই কাজ করা যায়।

কম্পেয়ার করাঃ

অন্য সব ডাটা টাইপের মত বিগ ইন্টিজারে আমরা দুটি সংখ্যা তুলনা করার জন্য `==`, `>` বা `<` ব্যবহার করতে পারি না। দুটি বিগ-ইন্টিজার সমান কিনা তা দেখার জন্য আমাদের `equals()` মেথড টি ব্যবহার করতে হয়। সংখ্যা দুটি সমান হলে মেথড টি `true` রিটার্ন করে, ত নাহলে `false` রিটার্ন করে। দুটি বিগ-ইন্টিজার কম্পেয়ার করার জন্য জাভার `BigInteger` ক্লাসে `compareTo()` মেথড আছে। এই মেথড টি একটি ইন্টিজার সংখ্যা রিটার্ন করে। আমরা যখন দুটি সংখ্যা কম্পেয়ার করবো তখন প্রথম সংখ্যা টি যদি দ্বিতীয় সংখ্যার চেয়ে ছোটো হয় তাহলে মেথড টি `-1` রিটার্ন করবে, সংখ্যা দুটি যদি সমান হয় তাহলে `0` রিটার্ন করবে এবং প্রথম সংখ্যা টি যদি বড় হয় তাহলে `1` রিটার্ন করবে। নিচের কোড টি রান করে দেখো, তাহলে বিষয়টা আর ভাল ভাবে বুঝতে পারবে

```
1. import java.util.*;
2. import java.math.*;
```

```

3.
4. public class Main {
5.     public static void main(String[] args) {
6.         Scanner sc = new Scanner (System.in);
7.         BigInteger a, b;
8.
9.         a = sc.nextBigInteger();
10.        b = sc.nextBigInteger();
11.
12.        int cmp = a.compareTo(b);
13.
14.        if(cmp == -1)
15.            System.out.println("a is less than b");
16.        else if(cmp == 0)
17.            System.out.println("a is equal to b");
18.        else if(cmp == 1)
19.            System.out.println("a is greater than b");
20.    }
21. }

```

পাওয়ার বের করাঃ

আমরা চাইলে বিগ-ইন্টিজারের পাওয়ারও বের করতে পারি। এজন্য আমাদেরকে pow() মেথড টি ব্যবহার করতে হবে তবে এই মেথড টি প্যারামিটার হিসেবে একটি ইন্টিজার গ্রহন করে, তাই পাওয়ার এর মান টি অবশ্যই int ডাটা টাইপের রেঞ্জের মধ্যে হতে হবে। কোন একটি সংখ্যা n এর পাওয়ার p বের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = n.pow(p);
```

বিগ-মড ভ্যালু বেরকরাঃ

বিগ-মড বা $(n^p) \% m$ এক্সপ্রেশনের মান বের করার জন্য বিগ-ইন্টিজারের modpow() মেথড আছে। মেথড টি প্যারামিটার হিসেবে দুটি BigInteger নিয়ে থাকে। এখন আমরা যদি কোন একটি সংখ্যা n এর পাওয়ার p কে অপর একটি সংখ্যা m দিয়ে mod করতে চাই তাহলে আমাদের লিখতে হবে-

```
BigInteger ans = n.modPow(p,m);
```

মুডলার-ইনভার্সভ্যালু বের করাঃ

মুডলার-ইনভার্স বের করতে চাই এবং যে সংখ্যা টি দিয়ে mod করতে চাই, তাদেরকে অবশ্যই রিলেটিভলি-প্রাইম হতে হবে। এখন আমরা যদি কোন একটি সংখ্যা n এর মুডলার-ইনভার্স বের করতে চাই এবং যে সংখ্যা টি দিয়ে mod করবো তা যদি m হয় তাহলে আমাদের লিখতে হবে-

```
BigInteger ans = n.modInverse(m);
```

GCD বের করাঃ

দুটি সংখ্যার Greatest Common Divisor (GCD) বের করার জন্য বিগ-ইন্টিজারে gcd() মেথড টি আছে। দুটি সংখ্যা a এবং b এর GCD বের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.gcd(b);
```

ম্যাক্সিমাম/মিনিমাম বের করাঃ

জাভার বিগ-ইন্টিজারে দুটি সংখ্যার মধ্যে বৃহত্তর সংখ্যা বের করার জন্য max() এবং ক্ষুদ্রতর সংখ্যা বের করার জন্য min() মেথড রয়েছে। দুটি সংখ্যার মধ্যে ম্যাক্সিমাম বা মিনিমাম বের করার জন্য আমাদের লিখতে হবে-

```

BigInteger largest = a.max(b);
BigInteger smallest = a.min(b);

```

Absolute মান বের করাঃ

কোন সংখ্যার পরম মান বা Absolute value বের করার জন্য বিগ-ইন্টিজারে `abs()` একটি মেথড আছে। কোন একটি সংখ্যা `n` এর Absolute মান বের করতে চাইলে লিখতে হবে-

```
BigInteger ans = a.abs();
```

সম্ভাব্য-প্রাইম নাম্বারঃ

আমাদের কে অনেক সময় প্রাইম নাম্বার বের করার প্রয়োজন হতে পারে। বিশেষ করে ক্রিপ্টোগ্রাফীর জন্য প্রাইম নাম্বার জেনারেট করা অনেক জরুরি। একটি বিগ-ইন্টিজার সম্ভাব্য প্রাইম নাম্বার কি না তাও আমরা বের করতে পারি। এজন্য BigInteger ক্লাসে `isProbablePrime()` মেথড আছে। মেথড টি তে probabilistic primality টেস্টের জন্য Miller-Rabin অ্যালগরিদম ব্যবহার করা হয়েছে। মেথড টি প্যারামিটারে certainty হিসেবে একটি ইন্টিজার ভ্যালু নিয়ে থাকে এবং একটি বুলিয়ান ভাটি রিটার্ন করে। certainty দিয়ে অসম্ভাব্যতার হার বের করা হয়। তারপর তা থেকে সম্ভাব্যতার হার বের করা হয়। সম্ভাব্যতার হার বের করা হয় $(1 - 0.5^{\text{certainty}})$ এভাবে। কোন একটি সংখ্যা প্রাইম নাম্বার হবার সম্ভাব্যতা যদি এই মান কে অতিক্রম করে তাহলে মেথড টি true রিটার্ন করে, অন্যথায় false রিটার্ন করে। অর্থাৎ certainty এর মান যত বড় হবে, সম্ভাব্যতার হার তত বাড়বে। আবার অন্য দিকে সম্ভাব্যতার হার যত বাড়বে, প্রোগ্রামের এক্সিকিউশন টাইম ততো বেশি লাগবে। তাই এটা ব্যবহারকারীর উপর নির্ভর করে যে সে কতটুকু সময়ের মধ্যে কত সম্ভাব্যতার প্রাইম নাম্বার চায়।

কোন একটি সংখ্যা `n`, certainty `c` এর সাপেক্ষে প্রাইম কি না বের করতে হলে আমাদের লিখতে হবে-

```
boolean flag = n.isProbablePrime(c);
```

আবার কোন একটি সংখ্যার পরবর্তী কোন সংখ্যা টি একটি সম্ভাব্য প্রাইম তা বের করার জন্য বিগ ইন্টিজারে আছে `nextProbablePrime()` মেথড। মেথড টি একটি সম্ভাব্য প্রাইম নাম্বার রিটার্ন করে এবং রিটার্ন করা নাম্বারটি একটি কম্পোজিট নাম্বার হবার সম্ভাব্যতা থাকে 2^{-100} এর কম। কোন একটি সংখ্যা `n` এর পরবর্তী সম্ভাব্য প্রাইম সংখ্যা টি বের করতে হলে আমাদের লিখতে হবে-

```
BigInteger ans = n.nextProbablePrime();
```

আমাদের অনেক সময় র্যান্ডম প্রাইম নাম্বারের দরকার হতে পারে। বিগ ইন্টিজারে র্যান্ডম ভাবেও সম্ভাব্য প্রাইম নাম্বার জেনারেট করা যায়। এজন্য বিগ ইন্টিজারে `probablePrime()` মেথড আছে। মেথড টি প্যারামিটারে `bitLength` হিসেবে একটি ইন্টিজার নিয়ে থাকে। মেথড টি রিটার্ন হিসেবে একটি ইন্টিজার এবং করা নাম্বার টি এই `bitLength` সংখ্যক বিটের হবে। মেথড টি কল করার সময় এর মান সব সময় 2 বা তার বেশি হতে হবে। আর `rnd` হচ্ছে একটি র্যান্ডম বিট সোর্স, একটি সম্ভাব্য প্রাইম নাম্বার রিটার্ন করার জন্য যার primality টেস্ট করা হয়। মেথড টি একটি সম্ভাব্য প্রাইম নাম্বার রিটার্ন করে এবং রিটার্ন করা নাম্বার টি একটি কম্পোজিট নাম্বার হবার সম্ভাব্যতা থাকে 2^{-100} এর কম। এখন আমাদের যদি `n` সংখ্যক বিটের একটি র্যান্ডম সম্ভাব্য প্রাইম নাম্বার বের করতে হয়, তাহলে আমাদের লিখতে হবে-

```
Random rnd = new Random();
```

```
BigInteger ans = BigInteger.probablePrime(n, rnd);
```

Bitwise অপারেশনঃ

বিগ ইন্টিজারেও আমরা বিট-ওয়াইজ অপারেশন করতে পারি। এজন্য BigInteger ক্লাসে কিছু বিট-ইন মেথড রয়েছে।

AND (`a&b`) : বিগ ইন্টিজারের বিট-ওয়াইজ AND ভ্যালু বের করার জন্য রয়েছে `and()` মেথড। দুটি বিগ ইন্টিজার `a` এবং `b` এর AND ভ্যালু বের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.and(b);
```

OR (`a|b`) : বিগ ইন্টিজারের বিট-ওয়াইজ OR ভ্যালু বের করার জন্য রয়েছে `or()` মেথড। দুটি বিগ ইন্টিজার `a` এবং `b` এর OR ভ্যালু বের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.or(b);
```

XOR (`a^b`) : বিগ ইন্টিজারে `xor()` মেথড টি দিয়ে বিট-ওয়াইজ XOR বা Exclusive OR ভ্যালু বের করা হয়।

দুটি বিগ ইন্টিজার a এবং b এর XOR ভ্যুলবের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.xor(b);
```

NOT (~a): বিগ ইন্টিজারের বিট-ওয়াইজ NOT ভ্যুলবের করার জন্য রয়েছে not() মেথড। কোন একটি বিগ ইন্টিজার a এর NOT ভ্যুলবের করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.not();
```

Shift-Left (a<<n): আমরা চাইলে বিগ ইন্টিজারের বিট গুলো কে শিফটও করতে পারি। বিট গুলো কে বাম দিকে শিফট করার জন্য আমাদেরকে shiftLeft() মেথড টি ব্যবহার করতে হবে। মেথড টি প্যারামিটারে একটি ইন্টিজার সংখ্যা নিয়ে থাকে। একটি নাম্বারের বিট গুলো কে আমরা বাম দিকে কত বিট শিফট করবো এটা হচ্ছে তার মান। কোন একটি সংখ্যা a কে আমরা যদি n বিট বামে শিফট করতে চাই, তাহলে লিখতে হবে-

```
BigInteger ans = a.shiftLeft(n);
```

Shift-Right (a>>n): বিগ ইন্টিজারের বিট গুলো কে ডান দিকে শিফট করার জন্য আমাদেরকে shiftRight() মেথড টি ব্যবহার করতে হবে। এটি shiftLeft() মেথড টির মতই। কোন একটি সংখ্যা a কে আমরা যদি n বিট ডানে শিফট করতে চাই, তাহলে লিখতে হবে-

```
BigInteger ans = a.shiftRight(n);
```

Test-Bit (a & (1<<n)): একটি বিগ ইন্টিজারের n তম বিট টি 0 নাকি 1 তা বের করার জন্য BigInteger ক্লাসে আছে testBit() মেথড। মেথড টি প্যারামিটারে n এর মান নেয় এবং n তম বিট টি 1 হলে true রিটার্ন করে আর 0 হলে false রিটার্ন করে। এখন আমরা যদি কোন একটি সংখ্যা a এর n তম বিট চেক করতে চাই, তাহলে লিখতে হবে-

```
boolean flag = a.testBit(n);
```

Clear-Bit (a & ~(1<<n)): একটি বিগ ইন্টিজারের কোন একটি বিট কে 0 করতে হলে আমাদের clearBit() মেথড টি ব্যবহার করতে হবে। কোন একটি সংখ্যা a এর n তম বিট টি কে 0 করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.clearBit(n);
```

Set-Bit (a | (1<<n)): একটি বিগ ইন্টিজারের কোন একটি বিট কে 1 করতে হলে আমাদের setBit() মেথড টি ব্যবহার করতে হবে। কোন একটি সংখ্যা a এর n তম বিট টি কে 1 করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.setBit(n);
```

Flip-Bit (a ^ (1<<n)): একটি বিগ ইন্টিজারের কোন একটি বিট কে ফ্লিপ করতে হলে অর্থাৎ 0 থাকলে 1 করতে অথবা 1 থাকলে 0 করতে চাইলে আমাদের flipBit() মেথড টি ব্যবহার করতে হবে। কোন একটি সংখ্যা a এর n তম বিট টি কে ফ্লিপ করতে চাইলে আমাদের লিখতে হবে-

```
BigInteger ans = a.flipBit(n);
```

AND-NOT (a & ~b): দুটি বিগ ইন্টিজারের বিট -ওয়াইজ AND-NOT ভ্যুলবের করতে চাইলে আমাদেরকে andNot() মেথড টি ব্যবহার করতে হবে। দুটি বিগ ইন্টিজার a এবং b এর AND-NOT ভ্যুলবের করতে চাইলে

আমাদের লিখতে হবে-

```
BigInteger ans = a.andNot(b);
```

Lowest Set Bit ($\log_2(a \& -a)$): কোন সংখ্যার সবচেয়ে ছোট কোন বিটে 1 আছে তা বের করার জন্য বিগ ইন্টিজারে `getLowestSetBit()` মেথড আছে। কোন একটি বিগ ইন্টিজার `a` এর সবচেয়ে ছোট কোন বিট টি 1 তা বের করতে চাইলে আমাদের লিখতে হবে-

```
int ans = a.getLowestSetBit();
```

আরো কিছু প্রয়োজনীয় মেথডঃ

toString(): বিগ ইন্টিজার কেষ্টিং এ কনভার্ট করতে ব্যবহার করা হয়।

toByteArray(): বিগ ইন্টিজার কেবাইট-অ্যারে তে কনভার্ট করতে ব্যবহার করা হয়।

negate(): বিগ ইন্টিজারের সাইন পরিবর্তন করতে ব্যবহার করা হয়।

bitCount(): কোন সংখ্যার 2's complement এ সাইন বিটের বিপরীত কয়টি বিট আছে তা বের করার জন্য ব্যবহার করা হয়।

bitLength(): একটি সংখ্যা কে বাইনারি বৈজ্ঞেয় রিপ্রেজেন্ট করতে (সাইন বিট ছাড়া) কয়টি বিটের প্রয়োজন তা জানার জন্য ব্যবহার করা হয়।

signum(): কোন সংখ্যা ধনাত্মক, ঋণাত্মক অথবা শূন্য কি না তা জানার জন্য ব্যবহার করা হয়। ধনাত্মক হলে 1, ঋণাত্মক হলে -1 এবং শূন্য হলে মেথড টি 0 রিটার্ন করে।

intValue(): BigInteger থেকে int ডাটায়কনভার্ট করতে ব্যবহার করা হয়। তবে এ ক্ষেত্রে ডাটা ওভারফ্লো হলে মেথড টি কোন এক্সেপশন থ্রো করে না। এক্ষেত্রে `intValueExact()` ব্যবহার করলে এই মেথড টি এক্সেপশন থ্রো করে।

longValue(): BigInteger থেকে long ডাটায়কনভার্ট করতে ব্যবহার করা হয়। এই মেথড টি ও ডাটা ওভারফ্লো হলে এক্সেপশন থ্রো করেনা। এক্ষেত্রে `longValueExact()` ব্যবহার করলে এই মেথড টি এক্সেপশন থ্রো করে।

floatValue(): BigInteger থেকে float ডাটায়কনভার্ট করতে ব্যবহার করা হয়।

doubleValue(): BigInteger থেকে double ডাটায়কনভার্ট করতে ব্যবহার করা হয়।

hashCode(): হ্যাশ-কোড বের করার জন্য ব্যবহার করা হয়।

divideAndRemainder(): মেথড টি দুটি সংখ্যার ভাগফল এবং ভাগশেষ এক সাথে একটি BigInteger অ্যারে আকারে রিটার্ন করে। রিটার্ন করা অ্যারে টির [0] ইন্ডেক্স থাকে ভাগফলের মান এবং [1] ইন্ডেক্স থাকে ভাগশেষ এর মান। নিচের কোড টি রান করে দেখো তাহলে বিষয় টি আরো ভাল ভাবে বুঝতে পারবে।

```
1. import java.util.*;
2. import java.math.*;
3.
4. public class Main {
5.     public static void main(String[] args) {
6.         Scanner sc = new Scanner (System.in);
7.
8.         BigInteger a=sc.nextBigInteger();
9.         BigInteger b=sc.nextBigInteger();
10.
11.         BigInteger arr[] = a.divideAndRemainder(b);
12.
13.         System.out.println("Quotient = "+arr[0]);
14.         System.out.println("Remainder = "+arr[1]);
15.     }
16. }
```

এই ছিলো আমাদের BigInteger নিয়ে আলোচনা। Java তে BigDecimal নামে আরো একটি ক্লাস আছে যা দিয়ে Decimal সংখ্যার হিসাব নিকাশ করা যায়। এটির ব্যবহার BigInteger এর মতই। প্রায় সব গুলো মেথড একই রকম। BigInteger এর মেথড গুলোর সেসি কোড দেখতে চাইলে [এখানে](#) ঘুরে আসতে পারো।

এখন আসি প্রবলেম সলিভিং এর কথায়। এক্ষেত্রে আমি বলবো জাভার BigInteger ব্যবহার না করে নিজে কোড ইমপ্লিমেন্ট করে প্রবলেম সলভ করতে। এতে করে তোমার প্রোগ্রামিং এর দক্ষতা বাড়বে আর তার সাথে তোমার প্রোগ্রামটিও অনেক ইফিশিয়েন্ট হবে তবে জাভা দিয়ে কোন প্রোজেক্ট বা অ্যাপ্লিকেশন তৈরি করতে অনায়াসে BigInteger ব্যবহার করতে পারো। যাই হোক এখন যেহুত বিগ ইন্টিজার শিখে ফেলেছ, তাই নিচের প্রবলেম গুলো ঝটপট সমাধান করে ফেলো। হ্যাপি কোডিং ☺

- Codeforces 66A - Petya and Java
- Light OJ 1214 - Large Division
- UVA 424 - Integer Inquiry
- UVA 465 - Overflow
- UVA 495 - Fibonacci Freeze
- UVA 619 - Numerically Speaking
- UVA 623 - 500!
- UVA 713 - Adding Reversed Numbers
- UVA 748 - Exponentiation
- UVA 1226 - Numerical surprises
- UVA 10013 - Super long sums
- UVA 10083 - Division
- UVA 10106 - Product
- UVA 10464 - Big Big Real Numbers
- UVA 10494 - If We Were a Child Again
- UVA 10523 - Very Easy !!!
- UVA 10551 - Basic Remains
- UVA 10579 - Fibonacci Numbers
- UVA 10814 - Simplifying Fractions
- UVA 10925 - Krakovia
- UVA 10929 - You can say 11
- UVA 11448 - Who said crisis?
- UVA 11879 - Multiple of 17

Download PDF

Posted by Tushar Roy at ১২:২৫ am



Google-এ

Labels: Bangla, Basic, BigInteger, Java, Programming, বাংলা

কোন মন্তব্য নেই:

একটি মন্তব্য পোস্ট করুন

