Java tricks for competitive programming (for Java 8)

Although practice is the only way that ensures increased performance in programming contests but having some tricks up your sleeve ensures an upper edge and fast debugging.

1) Checking if the number is even or odd without using the % operator:(Also For C++)

Although this trick is not much better than using % operator but is sometimes efficient (with large numbers). Use & operator:

System.out.println((a & 1) == 0 ? “EVEN” : “ODD”);

Example:

num = 5

Binary: “101 & 1” will be 001, so true

num = 4

Binary: “100 & 1” will be 000, so false.

2) Fast Multiplication or Division by 2 (Also For C++)

Multiplying by 2 means shifting all the bits to left and dividing by 2 means shifting to the right.

Example : 2 (Binary 10): shifting left 4 (Binary 100) and right 1 (Binary 1)

n = n << 1; // Multiply n with 2

n = n >> 1; // Divide n by 2

3) Swapping of 2 numbers using XOR: (Also For C++)

This method is fast and doesn’t require the use of 3rd variable.

// A quick way to swap a and b

a ^= b;

b ^= a;

a ^= b;

4) Faster I/O:

http://www.geeksforgeeks.org/fast-io-in-java-in-competitive-programming/

5) For String manipulations:

Use StringBuffer for string manipulations, as String in java is immutable.

You can refer here :

http://www.geeksforgeeks.org/g-fact-27-string-vs-stringbuilder-vs-stringbuffer/

6) Calculating the most significant digit: To calculate the most significant digit of any number log can be directly used to calculate it.

Suppose the number is N then

Let double K = Math.log10(N);

now K = K - Math.floor(K);

int X = (int)Math.pow(10, K);

X will be the most significant digit.

7) Calculating the number of digits directly: To calculate number of digits in a number, instead of looping we can efficiently use log :

Number of digits in N = Math.floor(log10(N)) + 1;

8) Inbuilt GCD Method: Java has inbuilt GCD method in BigInteger class. It returns a BigInteger whose value is the greatest common divisor of abs(this) and abs(val). Returns 0 if this0 && val0.

Syntax :

public BigInteger gcd(BigInteger val)

Parameters :

val - value with which the GCD is to be computed.

Returns :

GCD(abs(this), abs(val))

```
// Java program to demonstrate how
// to use gcd method of BigInteger class

import java.math.BigInteger;

class Test
{
public static int gcd(int a, int b)
{
BigInteger b1 = BigInteger.valueOf(a);
BigInteger b2 = BigInteger.valueOf(b);
BigInteger gcd = b1.gcd(b2);
```

return gcd.intValue();

}

```
public static long gcd(long a, long b)
{
    BigInteger b1 = BigInteger.valueOf(a);
    BigInteger b2 = BigInteger.valueOf(b);
    BigInteger gcd = b1.gcd(b2);
        return gcd.longValue();
}


// Driver method
public static void main(String[] args)
{
    System.out.println(gcd(3, 5));
    System.out.println(gcd(10000000000L, 600000000L));
}
```

}

Output:

1

200000000

9) checking for a prime number: Java has inbuilt isProbablePrime() method in BigInteger class. It returns true if this BigInteger is

probably prime(with some certainty), false if it’s definitely composite.

BigInteger.valueOf(1235).isProbablePrime(1)

10) Efficient trick to know if a number is a power of 2 The normal technique of division the complexity comes out to be O(logN), but it can be solved using O(v) where v are the number of digits of number in binary form.

```
/* Method to check if x is power of 2*/
static boolean isPowerOfTwo (int x)
{
/* First x in the below expression is
for the case when x is 0 */
return x!=0 && ((x&(x-1)) == 0);
}
```

11) Sorting Algorithm:

Arrays.sort() used to sort elements of a array.

Collections.sort() used to sort elements of a collection.

http://www.geeksforgeeks.org/arrays-sort-in-java-with-examples/

http://www.geeksforgeeks.org/collections-sort-java-examples/

For primitives, Arrays.sort() uses dual pivot quicksort algorithms.

12) Searching Algorithm:

Arrays.binarySearch()(SET 1 | SET2) used to apply binary search on an

sorted array.
Collections.binarySearch() used to apply binary search on a collection based on comparators.

http://www.geeksforgeeks.org/arrays-binarysearch-java-examples-set-1/

http://www.geeksforgeeks.org/arrays-binarysearch-in-java-with-examples-set-2-search-in-subarray/

http://www.geeksforgeeks.org/collections-binarysearch-java-examples/

13) Copy Algorithm:
Arrays.copyOf() and copyOfRange() copy the specified array.
Collections.copy() copies specified collection.

https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#copy(%20java.util.List)

http://www.geeksforgeeks.org/java-util-arrays-copyof-java-examples/

14) Rotation and Frequency :We can use Collections.rotate() to rotate a collection or an array by a specified distance. You can also use Collections.frequency() method to get frequency of specified element in a collection or an array.

http://www.geeksforgeeks.org/java-util-collections-rotate-method-java-examples/

http://www.geeksforgeeks.org/java-util-collections-frequency-java-examples/

15) Most data structures are already implemented in the Collections Framework. For example : Stack, LinkedList, HashSet, HashMaps, Heaps etc.

http://www.geeksforgeeks.org/stack-class-in-java/

http://www.geeksforgeeks.org/linked-list-in-java/

http://www.geeksforgeeks.org/hashset-in-java/

http://www.geeksforgeeks.org/priority-queue-class-in-java-2/

http://www.geeksforgeeks.org/java-util-hashmap-in-java/