

Lec 5

DSA LECTURE-5

PAGE NO.

DATE

Lecture 4 HW → There were 2 ques given as HW
 Done in Lec 4 HW.

1) Reverse a number 2) Rotate a number

Questions Done in Lecture 5 CW

- 3) GCD & LCM
- 4) Prime Factorization
- 5) The Curious Case of Benjamin Bulles
- 6) Pattern 1

Questions Assigned as Lecture 5 HW

- Pythagorean Triplet
- (2) Pattern 2

GCD & LCM



Largest Common Divisor Lowest Common Multiple

Greatest Common Divisor

- यदि 2 नं. दिये हों, तो उन 2 नं. का GCD क्या होता है।
- यदि समेत 2 नं. का Lowest Common Multiple क्या होता है।

GCD? → The greatest common divisor (gcd) of 2 or more integers, which are not all zero, is the largest positive integer that divides each of the integers.

Eg → GCD of 36 & 24 is 12.

→ Because 12 is highest number that divides both 36 & 24.

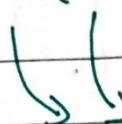
36 → Divisor of 36 → 2, 4, 6, 9, 12, 18 → Common Divisor = 2, 4, 6, 12 → Greatest = 12
 24 → Divisor of 24 → 2, 4, 6, 8, 12

LCM?

PAGE NO.

DATE

The least common multiple of 2 integers a and b is the smallest positive integer that is divisible by both a & b .



Here a & b both
should be greater
than 0.

Eg → LCM of 36 and 24 is 72.

Because the division
of integer by

$$36 \overline{)72} \quad 24 \overline{)72}$$

↑ ↑
 $\frac{72}{X}$ $\frac{72}{X}$

Zero is undefined.

(a) & (b) both are
divisible by 72.

अगर हम किसी भी no. को 0 से विभाग करते हैं तो (0) infinity होता है। इसका मतलब है कि वह कोई no. नहीं है। 0 का Multiple ही वह है। अब b 0 नहीं हो सकता।

∴ 72 is the multiple of a & b & it's smallest multiple of both. 72 is the LCM of a & b .

Divisor

$$\begin{array}{r}
 & 8 \\
 9 \sqrt{75} & \\
 -72 & \\
 \hline
 3 &
 \end{array}$$

Quotient Dividend
 ↓
 Remainder

PAGE NO. _____
DATE _____

Divisor,
Dividend,
Quotient,
Remainder
किसे कौलते हैं ?

finding GCD

Approach 1 → Brute Force.

Approach 2 → Division Method.

There are 2 approaches to find GCD.

1. Brute force to find GCD

हम 1 loop लगाते हैं → $\min(a, b)$ से 1 तक

अगर हम loop में ऐसा कोई
बी अंतिम फ्रियता है जिससे
a, b दोनों divide होते हैं
तो वो GCD होगा।

↳ Minimum of a & b

मतलब अगर a और b
में जो भी कोई दोहरा
integer होगा, उस
1 तक

आर loop में चलते - चलते
हम 1 तक पढ़ते जाएँ मतलब

ab का GCD 1 है और हम बोल सकते
कि a और b दोनों co-prime numbers हैं।

↳ Co-prime numbers
ही number होते हैं

↑ common
factor 1 ही।

FINDING GCD (Using Brute Force Approach)

PAGE NO.
DATE

```
→ import java.util.*;
→ import java.io.*;
→ public class Main {
    → public static void main (String args) {
        Scanner scan = new Scanner (System.in);
        int n1 = scan.nextInt();
        int n2 = scan.nextInt();
        int min = n1; → पहले मानिया कि n1 ने minimum
        if (n2 < n1)
            min = n2; → लेकिन n2 n1 से हमें हुआ
        } → तो minimum को n2 बनावें
    } → n1 को replace करें।
    int gcd = 1; → पहले मानिया कि gcd 1 है
    for (int i = min; i > 1; i--) → loop चलेगा reverse
        if ((n1 % i == 0) && (n2 % i == 0)) → order में min से
            gcd = i; → 1 तक तक सबसे पहले
        break; → सबसे first common
    } → divisor भी i.e. gcd में।
    } → अगर gcd मिलपाएँ तो loop
    } → break करें वही loop चलता जाएगा।
    } → और gcd की value द्योगी हो जाएगी।
    } → और loop चला greatest
    } → value मिलने के बाद भी तो
    } → gcd की value नहीं affect होती।
    System.out.println(gcd);
```

Time Complexity = $O(\min(a, b))$
→ Since we have just 1 loop from $\min(a, b)$ to 1.

Space Complexity = $O(1)$
→ Because no auxiliary space was needed

2. Division Method to find GCD.

PAGE NO. _____
DATE _____

Quotient

Divisor Dividend

Remainder

इस method में हमें 2 ग्रेटर numbers में से एक को dividend

$n_1 = 36$ बनाता है और दूसरे को divisor ताकि

$n_2 = 24$ वह division करते हैं जिसको dividend और divisor

24 $\sqrt{36}$ को dividend ताकि divisor वाले 1.

20

अब जो remainder मिलेगा उसकी अब divisor
बनातो और जो पहले divisor था वो अब
dividend बनायेगा।

2.

12

24

24

0

अब ये task करता जायगा

जबकि हमें एक ऐसा Divisor of

मिलजाएं जो dividend को पूरी

तरीके से divide करता है और

remainder 0 होता है।

जैसा divisor dividend को पूरा पूरा

divide करदे मतलब remainder

0 हो दे, तो उक्त जाओ; तब loop

को छोड़ो, तो divisor ही remainder

होगा।

1

24 $\sqrt{36}$

24

12

2

24

24

0

GCD

→ remainder 0 मिलते

divisor of this division

is the greatest common divisor.

Now, there can be a question in your mind

PAGE NO.

DATE

→ कौनसे number को dividend choose करना है और कौनसे number को divisor choose करना है?

Important Point

→ ऐसा जरूरी नहीं है कि हम greater number को dividend बनाना है और smaller को divisor.

We can just put any of these 2 i.e either dividend or divisor and the algorithm will further take care of it in the very first step (in case the smaller number is the dividend first)

एवं किसी भी no. of a और b में से divisor बना सकते हैं और dividend बना सकते हैं।

→ अगर हमने smaller number को divisor बना भी दिया तो हमारी Algorithm अपने आप बादमें उस smaller को divisor बनाकरी जोरे larger को dividend बना देगी।

Even if we choose the smaller number as the dividend first, it will be automatically adjusted in the next division operation due to the property of the algorithm

remainder becomes divisor and divisor becomes dividend

इस divisor से जब dividend को divide किया तो remainder 0 हो जाएगा तो GCD है।

$$\begin{array}{r} 36 \sqrt{24} \\ -24 \\ \hline 0 \end{array}$$
$$\begin{array}{r} 0 \\ 36 \sqrt{24} \\ -24 \\ \hline 0 \end{array}$$
$$\begin{array}{r} 36 \sqrt{24} \\ -24 \\ \hline 12 \end{array}$$
$$\begin{array}{r} 12 \sqrt{24} \\ -24 \\ \hline 0 \end{array}$$

तो अब हमें तो समझ आगया कि GCD Division Method से कैसे निकलते हैं।

PAGE NO.		
DATE		

Now, how to find LCM?

There is a relation between LCM & GCD.

Lowest
Common
Multiple

Greatest
Common
Divisor

$$\text{GCD} * \text{LCM} = n_1 * n_2$$

GCD और LCM की multiplication की value is equal to number 1 और number 2 की multiplication की value.

इसीकी modify करके हम ऐसे भी लिख सकते हैं, which gives us the value of LCM.

$$\text{LCM} = (n_1 * n_2) / \text{GCD}$$

GREATEST COMMON DIVISOR

USING DIVISION METHOD

```

→ import java.util.*;
→ import java.io.*;
→ public class Main {
→   public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    int n1 = scan.nextInt();
    int n2 = scan.nextInt();
    int dividend = n1;
    int divisor = n2;
  }
}
  
```

प्रैगलग से
n1, n2
की value

Same करना

variable
में पर्याप्त
कीटी

Original

Value R
करना चाही

करोगे bcz

we need

original

values to
find LCM.

int gcd = 1;

while (dividend % divisor != 0)

{

int remainder = dividend % divisor;

dividend = divisor;

divisor = remainder;

}

gcd = divisor;

System.out.println(gcd);

int lcm = (n1 * n2) / gcd;

System.out.println(lcm);

}

}

}

] n1 की dividend होता है।
] n2 की divisor

] Initially gcd को 1 माना

] Loop तक

चलेगा पहले क

remainder 0 तक

पर्याप्त

दृष्टि
remainder

0 हुआ

संतुष्ट

अब

divisor की

ई दृष्टि

gcd की

} lcm की
value बढ़ती
formula के करके।

Ques. 2

PAGE NO.

DAT

PRIME FACTORIZATION OF A NUMBER

- 1) We are required to print/display the prime factorization of a number -
- 2) Take as input a number n.
- 3) Print all its prime factors from smallest to largest.

जब तक	2	1440
इस नो. से	2	720
divide करते	2	360
दूर remainder	2	180
आता इसे	2	90
वर्तक इससे	3	45.
divide करते	3	15
यहाँ तक	5	5
इससे		1
next		
no. से divide		
करने का try		
करते उसके लिए		
शीर्षक process		
follow करो unless		
divident no. 1 ना हो जाएगा		

Prime factors

2, 2, 2, 2, 2, 3, 3, 5.

divident को (given number)
i.e 1440
सबसे छोटे number (lets say)
से divide करना शुरू करो और
तब तक divide करो जब तक
उस नो. से (वैसे) further
divide ना हो पाया।

→ फिर अब divident वो है i.e (45)
को (2 से) छोटे no. से divide करें
करते हुए करो अब तक तक divide
करो तब तक current division से
further divide ना हो पाया।

& this process will continue
until the given no.
(divident) becomes 1.

Basically we are dividing the number, it could be
divided.

2	720
2	360
2	180
2	90
3	45
3	15
5	5
	1

सबसे छोटे number से divide करो
और divide करते जाओ तक उससे और divide न
हो पाये।

After next, prime द्वांते जिससे divide हो रहा है,
और उससे divide करते चलो तक और
divide न हो तब तक this process
goes on तब तक number
1 तक हो जाएगा।

(जब तक remainder 0 आएगा
तब तक उसी number से
divide करते चलेंगे। जब
remainder 0 आना बंद हो जाएगा तो next
number से divide करके देखेंगे।)

$$pq = n$$

$$p > \sqrt{n} \text{ & } q < \sqrt{n}$$

$$p = \sqrt{n} \text{ & } q = \sqrt{n}$$

$$p < \sqrt{n} \text{ & } q > \sqrt{n}$$

$$\text{अगर } p * q = n$$

तो पा तो p, \sqrt{n} से बड़ा होगा

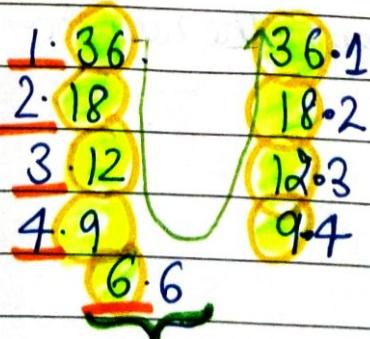
\sqrt{n}, q से बड़ा होगा

p और q कोनो बड़े नहीं हो सकते

मतलब Unique factor हमें मिलेंगे तो \sqrt{n} तक ही मिलेंगे,
उसके बाद नहीं मिलेंगे।

$$\text{eg} \rightarrow n = 36$$

1, 2, 3, 4, 6, 12, 18, 36
की उपर्युक्त
factors की सूची



\sqrt{n} तक ही unique characters
मिलेंगे

We divide the given number by the smallest number and keep on dividing till it no longer could be divided by that number.

As soon as this happen, we check the divisibility of current state of given number with the next greater number as the divisor & continue the process until the given number becomes 1.

- basically "We are reducing the number "(Decomposing the number into smaller number)!"

कभी - कभी जब वह loop चलने के बाद यदि $n > 1$ होता है।

eg → $\frac{46}{\sqrt{n}}$ → \sqrt{n} will be nearly 6 तो nearly 6 तक loop चलने के बाद यदि n की value 23 होती है।

$2 \times 23 \rightarrow$ 3 तक विभाग नहीं होता 2 point करता है।
now $n=23$. Because from 2 to 6 , 2 is a prime factor

→ तो इस case में current n की value (23) यह एक prime factor है लेकिन n as it is.

import java.util.*;

public class Main

{

 public static void main (String [] args)

 Scanner scan = new Scanner (System.in);

 int n = scan.nextInt();

 for (int i = 2; i * i <= n; i++)

 while (n % i == 0)

 System.out.print (" " + i);

 n = n / i;

 if (n != 1)

 System.out.print (" " + n);

eg. $\sqrt{46}$ लेकिन
2 is printed as a factor
 2×23 .
new $n = 23$.
 $n = \sqrt{n}$.

परफ न
no. के unique
prime factors
 \sqrt{n} तक ही
भलते हैं

prime
factor \sqrt{n} से
अधि-काफ़ी further
होते हैं तो कर सकते
divide जाएं कर सकते

योग्यी दो prime
होते हैं तो
prime n
and
is

The Curious Case of Benjamin Bulbs

PAGE NO.	
DATE	

Rules :-

→ Let's say there are n bulbs given.

→ ~~CTR~~ voltage n times fluctuates.

→ Initially all bulbs are off.

→ ~~CTR~~ voltage fluctuation of ~~all~~ bulbs at station toggle.

Bulbs off \rightarrow \uparrow \downarrow on \uparrow \downarrow

Voltage fluctuation 1 → all bulbs state toggled (off \rightarrow on).

Voltage fluctuation 2 → Bulb no. (multiple of 2) (state get toggled)

Voltage fluctuation 3 → Bulb no. (multiple of 3) (state get toggled)

Voltage fluctuation 4 → Bulb no. (multiple of 4) (state get toggled)

& so on...

$n=25$

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

Now we need to tell which bulb no. is still on.

The Bulbs which are fluctuated odd no. of times are still on.

So → Bulb no. → 1, 4, 9, 16, 25 will still remain on

Because of odd no. of fluctuation

ता है

Bulle no. 1 print करने के लिए

4

9

16

25

because they
will be an

1

4

9

16

25

}

What is common in all these
Bulle no.

bcz 25 सम Perfect Square है

$$1 \rightarrow 1 * 1$$

$$4 \rightarrow 2 * 2$$

$$9 \rightarrow 3 * 3$$

$$16 \rightarrow 4 * 4$$

$$25 \rightarrow 5 * 5$$

Perfect Square के Bulle की number की odd no. of times
को fluctuate हो रहे हैं।

इसे हम example से समझते हैं।

Let's say 24, 36

Not a
Perfect Square

Perfect Square

अब 24 के 36 के factors निकलते

$$\begin{array}{r}
 \underline{1029} \\
 \underline{2012} \\
 \underline{308} \\
 \underline{406}
 \end{array}
 \quad
 \begin{array}{r}
 2401 \\
 \underline{1202} \\
 \underline{803} \\
 \underline{604}
 \end{array}$$

$$\begin{array}{r} \underline{1} \cdot 36 \\ \underline{2} \cdot 18 \\ \underline{3} \cdot 12 \\ \underline{4} \cdot 9 \end{array} \quad \begin{array}{r} 36 \cdot \underline{1} \\ 18 \cdot \underline{2} \\ 12 \cdot \underline{3} \\ 9 \cdot \underline{4} \end{array}$$

Hans factors

Have 9 factors

24 → 1, 2, 3, 4, 6, 8, 12, 24

२५ इन 8 no. की tables में
आता है

at bulb no. 27 & 8 वाले (even times). voltage fluctuation की state change करें।

So even no. of fluctuations it will

off state में होता था क्योंकि इसका अविद्युत रुप था।

~~off → on → off → on → off → on → off~~

off at the end.

Similarly for all other fluctuations

$$36 \rightarrow 1, 2, 3, 4, 6, 9, 12, 18, 36,$$

36 इन 9 no. की table में आता है

36 वाले 9 no. की तालिका है।
No. Bulb no. 36 की 9 ग्रेड (odd) voltage fluctuation & static change की

odd no. of fluctuation \Rightarrow Bell on state of cell bcz initially it was off

~~off → ON → off → ON → off → ON → off → ON → off → ON~~

N
1. on at the end

Similarly for all odd fluctuations

* तो हमें बस वो ही no. print कराने हैं जो perfect squares हैं।

import java.util.*;

public class Main {

public static void main (String []args)

8 scanner scan = new Scanner (System.in);
int n = scan.nextInt();

for (int i=1; i*i <= n; i++)

System.out.println (j*i);

3

पूर्ण वर्ग वाले
perfect square
प्राप्त करने का तरीका
उत्तम व निम्नलिखि
less & gt;
equal &

Ques. 4: Pythagorean Triplet

PAGE NO.		
DATE		

* यदि 3 numbers given होंगे।

We need to tell whether the given numbers can form a right-angled triangle or not.
(Pythagorean Triplet).

a, b, c } lets say these are 3 values.

तो यदि पहले इन तीनों values में से max value find करनी होगी।

lets say a is the max value.

$$\text{परिवर्तन } a^2 = b^2 + c^2$$

(max). कि square वाली दोनों values के square का sum के equal होगा।

तो परिवर्तन

Pythagorean triplet के equal होगा।

A Pythagorean Triplet consist of 3 positive integers a, b, c such that $a^2 + b^2 = c^2$.

A Triangle whose sides form a Pythagorean Triple is called a Pythagorean Triangle and is necessarily a right triangle.

This is a fairly straightforward problem.

* इसे बताना होगा कि whether the given numbers can form a right-angled triangle or not.

* We find the greatest number out of three numbers.

We sum the squared values of the smaller values (lets say b & c were smaller)

and try equating them with the squared value of the greatest value (a is ^{lets say} the greatest)

If the equation is valid then the given numbers are the Pythagorean triplet & we can print "true".

else, we can print "false".

→ सबसे पहले वे check करें a,b,c में से max value किसकी है।

let's say a की value b और c से बड़ी है।

तो a की value maximum है।

अब इसे check करें कि $a*a = (b*b) + (c*c)$

यदि $a^2 = b^2 + c^2$?

if yes → then print true
if no → then print false

import java.util.*;
public class Main

{
public static void main (String [] args)

{
Scanner scan = new Scanner (System.in);

int a = scan.nextInt();

int b = scan.nextInt();

int c = scan.nextInt();

if (a > b && a > c)

{
if (a*a == (b*b) + (c*c))

System.out.println (true);

}
else

System.out.println (false);

}
else if (b > a && b > c)

{
if (b*b == (a*a) + (c*c))

System.out.println (true);

}
else

System.out.println (false);

}

→ BTTR
a max &
it check OR
if $a^2 = b^2 + c^2$ ->
it yes then print
true else
print false.

→ BTTR
b max & it check
if $b^2 = c^2 + a^2$
if yes then
print true
else
print false.

```
else if (c>a & b>c)
{
    if (c*c) == (a*a) + (b*b))
```

store max &
then check
if if

```
{ System.out.println (true);}
```

$c^2 = a^2 + b^2$
if yes.
then print
true else
print false.

```
} else
```

```
{ System.out.println (false);}
```

Ques. 5. Min Max 2

DATE

→ यहाँ पर array given होता, यहाँ उस array में second max value और second min value क्या होगी।

eg → 3 | 7 | 2 | -1 | 4 | 9 |

second max
second min (वाकी सब value एवं end value except min).

(वाकी सब value एवं value except max)

Approach

→ पहले हम array पर loop लगाकर max और min value find करें।

→ फिर हम array पर loop लगाकर
वह maximum value find करें जो array में
वाकी सब value एवं (अ) लेकिन max से होली है।
→ This will be Second Max.

→ फिर हम array पर loop लगाकर
वह minimum value find करें जो array में
वाकी सब value एवं (अ) लेकिन min से होली है।
→ This will be Second Min.

```
import java.util.*;
```

```
public class Main
```

```
{ public static void main(String[] args)
```

```
{ Scanner scan = new Scanner(System.in);  
int n = scan.nextInt();  
int arr[] = new int[n];
```

```
for (int i=0; i<n; i++)
```

```
{ arr[i] = scan.nextInt();  
}
```

} array
की values
get करती है.

```
int max = Integer.MIN_VALUE;  
int min = Integer.MAX_VALUE;
```

max को -∞
पर set करता है.
करते कि maximum
की identity
होती है -∞

```
for (int i=0; i<n; i++)
```

```
{ if (arr[i] > max)
```

```
{ max = arr[i];  
}
```

```
if (arr[i] < min)
```

```
{ min = arr[i];  
}
```

+ की identity
होती है
* की identity
1 होती है
- की identity
-∞ होती है
min की identity
+∞ होती है

unit SecondMax = Integer · MAX_VALUE;
unit SecondMin = Integer · MAX_VALUE;

for (unit i=0; i < n; i++)

} if (arr[i] > SecondMax && arr[i] < max)

} SecondMax = arr[i];

} second
max
value
= 0 by default

} second
min की
value
= 00

} if (arr[i] < SecondMin && arr[i] > min)

} SecondMin = arr[i];

} arr[i] की value
arr[i] से कम है.
Secondmin से ज्यादा हो.
arr[min] से ज्यादा हो.
SecondMin की value
update करते हैं.

System.out.println(SecondMax);
System.out.println(SecondMin);

} Second
max
SecondMin
of value
update करते हैं.