DAY-1 - 04/05/2021 - Introduction

#include - Preprocessors

iostream, cmath - headers

using namespace std - Avoid name conflicts

main() - The most prioritised function and start point of program

cin - input function in CPP

SYNTAX: cin >> a >> b; [a,b are any type variables declared]

cout - output function in CPP

SYNTAX: cout << a << b; [a,b are any type of variables]

endl - It is used after cout to give a new line in place of (\n) in strings

SYNTAX: cout << "a" << endl << "b"; [Prints a and b on two lines]

// - Used to comment a single line

/* */ - Used to comment multiple lines [must end]

Variable - It is a container to store data in it

Data types

int	→	4 bytes	→ 10
float	→	4 bytes	→ 2.51
char	→	1 byte	→ 'a'
string	→	(as required)	→ "Hello"
bool	→	1 byte	→ true / false

When we print bool, true - 1 and false - 0 will be printed

return 0; - Shows that work is completed in the (task) program

DAY-2 - 05/05/2021 - Operators

namespace - It is used to avoid the name conflicts.

using namespace std; - It is used to call the cout and cin from the namespace std created in the iostream module. Without using this, we have to use std::cin, std::cout, std::endl in place of cin, cout, endl etc.

Operators:-

1. Arithmetic operators:- +, -, *, /, %

2. Assignment operators:- =, +=, *=, -=, /= etc...

3. Comparision @ Relation operators:- >, >=, <, <=, ==, !=

4. Bitwise operators:- &, |, ^, <<, >>

and

```
  5 - 0101
& 10 - 1010
-----------
  0000 - 0
```

(both 1's gives 1)

or

```
  5 - 0101
| 10 - 1010
-----------
  1111 - 15
```

(atleast one 1)

ex-or

```
  8 - 1000
^ 12 - 1100
-----------
  0100 - 4
```

(diff bits giving 1)

left shift

<< ⇒ Shifts every bit x to left by given number. [shifts 2^n x value if given n]

2 ⇒ 0010 << 1 gives 4 ⇒ 2×2^1 = 4 - 0100

2 ⇒ 0010 << 2 gives 8 ⇒ 2×2^2 = 8 - 1000

right shift

>> ⇒ Shifts every bit to right by given number [divides by 2^n]

5 ⇒ 0101 >> 1 ⇒ 0010 ⇒ 2 gives $5/2^1$ [floor].

10 ⇒ 1010 >> 2 ⇒ 0010 ⇒ 2 gives $10/2^2$ [floor]

DAY-3 06/05/2021 - Operators and Modules

Left shift and right shift formulas :-

num >> k - num/2^k [right shift]

num << k - num * 2^k [left shift]

Count the set bits of a number :-

To check set bits of 10 :-

10 - 1010 ⇒ right shift of 10 ⇒ 0101

& 0001 & 0001

0000 - unset 0001 - set

set bits = 0 set bits = 1

right shift again ⇒ 0010 again >> ⇒ 0001

& 0001 0001

0000 - unset 0001 - set

set bits = 1 set bits = 2

again right shift gives 0. So set bits of 10 = 2.

Process :-

1. We have to check the 'n' with 1 such that n&1, if we get 1, the last bit is setbit else it is not a setbit.
2. Now right shift n and again repeat the above process until we get n=0.

Ex-OR formula :-

1 → 1	1^2^3^4^5 → 1	1^2^... ^9 → 1
1^2 → 3	1^2^... ^6 → 7	1^2^... ^10 → 11
1^2^3 → 0	1^2^... ^7 → 0	1^2^... ^11 → 0
1^2^3^4 → 4	1^2^... ^9 → 8	1^2^... ^12 → 12

So to find exor upto n numbers, ~~if n...~~

like 1^2^3^... ^n ⇒

if $n\%4 = 0$, answer will be n
if $n\%4 = 1$, answer will be 1
if $n\%4 = 2$, answer will be $n+1$
if $n\%4 = 3$, answer will be 0

* Note: $n \wedge n = 0$ always. This is used to find the unique elements.

5. Logical Operators: &&, ||, !

Strings: We can use the string in C++. We can include the string library using #include<string>

Math: To do mathematical tasks, we have to include the cmath library. We can do sqrt, power, etc.

Powerful header:

```
#include<bits/stdc++.h>
```

The above header includes all the libraries like cmath, string, vector, list, map, dynamic functions and all using the same header file.

Day-4 - 07/05/2021 - Conditional Statements and Loops

Conditions: The condition can be either true or false.

Here '0' will be considered as false and any other number is considered as true because there will be atleast one '1' in the binary digits of the number except '0'. Zero has only 0s in it's binary representation so it is considered as false.

- The conditions can be combined with logical operators to compare two or more conditions.

if: The body of if will be executed when condition is true else it is not executed.

else: It must be an extension for if and it has no specific condition. If the condition of "if" is false, automatically "else" part will be executed.

else-if: It is followed by "if" and will be skipped if the first condition is true. It is an extension of else including a condition for it.

Loops:

(i) **Definite loops:** The loops that are iterated over a finite range of values
ex: for

(ii) **Indefinite loops:** Loops that run until a condition is checked and it is false
ex: while, do-while

When there is no syntactical error, the loop runs based on logic and if the logic leads to a non-ending point and leads to infinite loop.

Day - 5 -08/05/2021 - Pascal Triangle & LCM

Find Sum:- To find the sum of elements of n^{th} row of the pascal triangle, sum is 2^n [n starts from 0]

The first column of pascal triangle are 1's and remaining can be obtained by

$$a[row][col] = a[row-1][col-1] + a[row-1][col]$$

Pascal Triangle:-

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
```

[∵ a[1][1] = 1 = a[0][0] + a[0][1] (1 to 0)]

It gives our binomial expressions like:-

$(a+b) = (a+b)$

$(a+b)^2 = a^2 + 2ab + b^2$

$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

⋮ ⋮

LCM:-

To find faster, we can start loop from big number of two and run it upto product of 2 nums & increment by big num.

```
big = num1 > num2 ? num1 : num2;
for(int i = big; i <= num1 * num2; i += big)
{
    if(i % num1 == 0 && i % num2 == 0)
    {
        break; //Save i value or print here
    }
}
```

Day-6 - 10/05/2021 - LCM & HCF (GCD)

Pseudo code for LCM:- (optimized).

```
res = 1, r = 2
if (num1 % r == 0  && num2 % r == 0)
{
        num1 = num1/r;
        num2 = num2/r;
        res = res * r;
}
else
{
        r++;
}
if ( num1 < r || num2 < r)
{
        res = res * num1 * num2;
        break;
}
```

r value { 2 | 12, 24
2 | 6, 12
3 | 3, 6
⇒ 1, 2
↑ num1 ↑ num2
final.

LCM = (res * r in loop)

Pseudo code for GCD:-

Euclid's algorithm:-

GCD(a,b) = (a-b, b) if a > b
⇒ (a, b-a) if a < b (swap)
until 1st becomes zero, b is GCD.

GCD(30, 24) ⇒ (30-24, 24)
= (6, 24)
swap ⇒ (24, 6)
(24-6, 6) = (18, 6)
= (18-6, 6) = (12, 6)
⇒ (12-6, 6) = (6, 6) = (6-6, 6)
= (0, 6) ⇒ 6 is GCD of 30, 24

More optimized:-

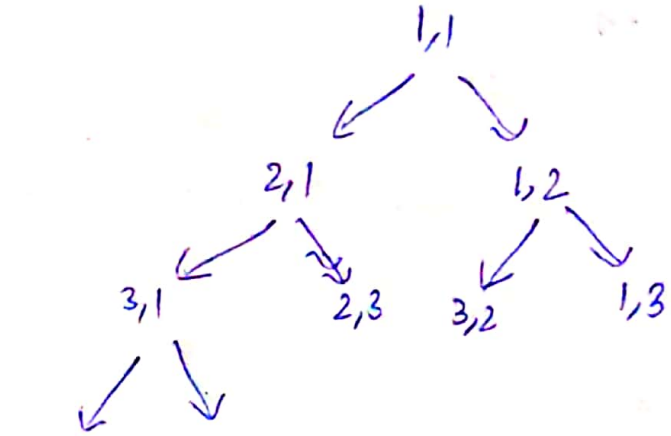
GCD(a,b) ⇒ a > b
(a%b, b)
swap
} repeat
until 1st digit one becomes 0,
b is GCD

24) 30 (1
24
6) 24 (4
24
0

6 is GCD of 30, 24

Question given in AWS: Game of co-ordinates.

A person start from (1,1), input will be given to us like (2,3), (5,4), (3,3) Print "Yes" or "No" if he can move to that point or not. He can move in two directions, (a+b, b), (a, b+a)



and so on.

* By applying reverse formula, from (4,1) to (3,1) is (a-b, a)

(or) (3,4) to (3,1) is (a, b-a) which is gcd formula.

* So the GCD of any co-ordinate is finally 1 because GCD of (1,1) is 1. So if the GCD of two numbers is 1, it is Yes.

* If the GCD of two given input numbers, equals to 1, then he can reach to point given as input starting from (1,1). If the GCD is not 1, then the answer is "No".

(i) GCD of 2,3 is 1 so print "Yes"

(ii) GCD of 5,4 is 1 so print "Yes"

(iii) GCD of 3,3 is 3 so print "No"

Day-7 – 11/05/2021 – Problems on numbers

Reverse of a number:-

⇒ num
⇒ res = 0

r = num % 10
num = num / 10
res = res * 10 + r

} until n!=0 (while(n))

because every num is true except 0

Palindrome or not:-

check reverse of a number equals to actual number.

Even and odd digits:-

We can increment even or odd variables in reverse of number program and reject result part.

~~Even~~ Strongly even or Strongly odd or mixed:-

In the above program if odd=0, strongly even or even=0, strongly odd else mixed.

Happy number:-

Sum of squares of digits until it becomes single digit

13 ⇒ 1^2+3^2 ⇒ 1+9 ⇒ 10 ⇒ 1^2+0^2 = 1 ⇒ happy number

19 ⇒ 1^2+9^2 ⇒ 82 ⇒ 8^2+2^2 = 68 ⇒ 6^2+8^2 = 100 ⇒ $1^2+0^2+0^2$ = 1 Happy numb

7, 1 are single digit happy numbers.

Day-8 - 12/05/2021 - Problems on numbers

1. Check as prime or not:

(i) Check from 1 to n [n operations]

(ii) Check from 2 to n-1 [n-2 operations]

(iii) Check from 2 to n/2 + 1 [optimised]

(iv) Check from 2 to $\sqrt{n}$ [very optimised]

2. To get last digit of power of any number:

ex: 2^{137} , $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$.

Repeatedly we get . [2, 4, 8, 6]

1 2 3 0 [4%4 = 0]

$\therefore$ 137 % 4 = 1

last digit of $2^{137} = 2$

Note: We have to divide the exponent part with the length of the array formed by the repeating values.

In program, we can take [6, 2, 4, 8] and by taking the index of array as 137%4, we can get the last digit.

Day-9 - 13/05/2021 - Problems on numbers

1. Find factors of a number :-

(i) Optimized :- Iterate upto n/2 times

(ii) Very optimized :- Iterate upto sqrt(n) times and on find factors as num/factor = another associated factor.

ex:- 100

1) (2) [4] 5 [10] 20 [25] (50) (100

sqrt.

The above are related by one before and one after the square root. So we can find them easily.

Day - 10 - 14/05/2021 - Problems of numbers

Handling larger numbers :- Continued in advanced topics //

Long long int range is upto 10^{18}

CPP array :- Limits

10^6 for int & 10^7 for char, bool arrays.
10^7 if declared globally.

So we can store them in arrays discussed later.
(using vectors)

Day-11 - 15/05/2021 - Problems of numbers

Prime factors:

$30 \Rightarrow 2 \times 3 \times 5$.

* After dividing any number with 2 for all possible times, then any other multiples of 2 can't divide that number. Similarly all other numbers.

No of factors using prime factors:

$Num = a^p \times b^q \times c^r \times \ldots$ [$\because a, b, c, \ldots$ are prime factors]

No. of factors $= (p+1)(q+1)(r+1) \ldots$

Day-12 - 17/05/2021 - Problems on Numbers

Factorial:

Factorial of a number is a product of numbers from 1 to that number.

To find the no. of zeroes at last (leading):

(i) We can find by using prime factors as no. of zeroes will be as equal to no. pairs of (2,5) available. Always no. of 5's will be less than no. of 2's. So the no. of 5's will be answer.

(ii) The no. of 5's count in prime factors of solution of factorial [optimized]

1-4 = 0

5-24 = 1

25-124 = 2 increased

125-... = 3 increased.

∴ no. of zeroes = $\frac{n}{5} + \frac{n}{25} + \frac{n}{125} + \dots$

$5! = 2^3 \times 3 \times 5$ [One 5]

$10! = 2^8 \times 3^4 \times 5^2 \times 7$ [two 5's].

Since no. of 5's changes according to those numbers. At every interval of 5, 10, 15, 20, ..., the no of 5's increase by 1 but at 25, 125 etc, the number is increasing by one or two excess.

Day-13 - 18/05/2021 - Problems on Numbers

Bit Magics:

To check whether a bit at position :

num = num >> (pos-1) , now check num&1

Unset the p^{th} position bit :

x = 1 << (p-1) , perform num ^ x.

Set the p^{th} position bit :

x = 1 << (p-1) , perform num | x

Day-14 – 19/05/2021 – Problems on numbers.

Best way to check prime or not:

(i) Most of the numbers are divisible by 2 & 3.

(ii) Check from 5 to n, increment 6, check i, i+2

(i=5, i+2=7) i+6 (i=11, i+2=13) i+6 (i=17, i+2=n), ...

So the code will be,

```
if (n==1)
    return false
if (n==2 || n==3)
    return true
if (n%2==0 || n%2==3)
    return false
for (int i=5; i<=n; i+=6)
    if (n%i==0 || n%(i+2)==0)
        return false
return true
```

Day-15 - 20/05/2021 - Problems on numbers

Problem: If a bucket has 7 chocolates, and we can take 1 or 3 chocolates at a time. in how many ways, we can empty the bucket?

To remove 1 chocolate → (1) → 1 way

2 → (1,1) → 1 way

3 → (1,1,1), (3) → 2 ways

4 → (1,1,1,1), (3,1), (1,3) → 3 ways

5 → (1,1,1,1,1), (1,3,1), (3,1,1), (1,1,3) → 4 ways

From above,

To, give 4, it takes value from 3 & 1

Similiarly for 5, it takes values from 4 & 2.

$\therefore ways(4) = ways(3) + ways(1)$

$ways(5) = ways(4) + ways(2)$.

$\therefore ways(n) = ways(n-1) + ways(n-3)$

Day-16 - 21/05/2021 - Problems on numbers

Problem: If we are finding n & (n-1) & (n-2) & ... upto 1, print number when it becomes 0.

Normally we iterate from n-1 and perform the & operation and break if it becomes 0.

But,

1 & 2 = 0

2 & 3 = 2 ; 2 & 3 & 4 = 0.

4 & 5 = 4 ..., 4 & 5 & 6 & 7 = 4, ... 4 & 5 & 6 & 7 & 8 = 0.

8 & 9 = 8 ..., 8 & 9 & ... & 15 = 8, 8 & 9 & 10 & ... & 15 & 16 = 0

when we reach the 2^n, we get 0. [2 & 1 = 0 8 & 7 = 0 32 & 31 ... 4 & 3 = 0 16 & 15 = 0 64 & 63 ...

So, we can find the least nearest 2^n value and return the $2^n - 1$ value, it will be the result.

To make 17 as 0, we get 5 right shifts. So to get 16, we make (count-1) left shift of 1.

Similiarly 5 as 0, we get 3 right shifts, So to get 4, we get (count-1) left shifts of 1

for 17, 5 count → 1 << (5-1) = 16.

for 5, 3 count → 1 << (3-1) = 4

Then return 16-1 or 4-1 as required.

Problem:- If a large array is given and only one element will be present single and all other will be even pairs.

ex: 1 2 2 3 3 4 4 5 5

By applying all elements with "^" we get the result.

Problem:- To calculate the power of a number, to represent it into the powers of multiples of 2

$3^{19} = 3^{16} \times 3^{2} \times 3^{1}$

write the binary of 19 ⇒ 10011

$3^{16} 3^{8} 3^{4} 3^{2} 3^{1}$

base of base applied.

Take $3^{10} = 3^{8} * 3^{2}$

binary of 10 ⇒ 1010

$3^{8} 3^{4} 3^{2} 3^{1}$

We take base, exp and by using above formula, we solve the power.

```
∴ while(exp)
  {
      if(exp&1) // To check the bit set or not
          res = res * base;
      exp >>1; // to shift to check the next bit
      base *= base; // base is changing from position to position [3^1, 3^2, 3^4, 3^8 squared]
  }
```

Here if there is a set bit, result will be multiplied with base value and every time exponent will be right shifted. base will be multiplied by itself to satisfy the exponet value of base power.

Day-17 - 22/05/2021 - Arrays

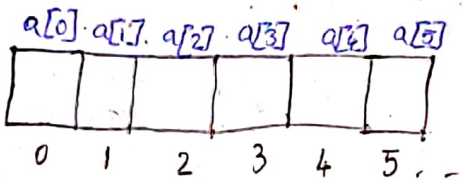
Array: *An array is a data structure where we can store multiple data of same datatype.

*Array has an index value from 0 to n-1 where the n is size of array.

* We can access elements of an array by index value of the element like a[0], a[1], etc..

Structure:

a=



* If we declare an array globally, you will get more space to store elements than declared locally and not needed to pass it as arguments.

* for(auto it : arr) - is a new way of accessing arr elements.

Day-18 - 24/05/2021 - Arrays

Problem: To get the most visited customer details based on the id of the customer.

* We can write a loop in loop.

Normal

* We can get a max element from array and increase it's value by taking that id value as index and the frequency will be the element of the array and we can print the ~~elements of~~ the index of the maximum element of the frequency to get the customer id.

Optimized

* We can use a map data structure to reduce the memory usage of the frequency array by ~~arr~~ key, value pairs

Day-19 - 25/05/2021 - Arrays & Vectors

* There are two types of arrays. Static and dynamic.
* For static arrays, we declare int arr[10] and it will not increase it's size upon adding more than 10 elements.
* For dynamic arrays, we use vector in which we can push pop elements. We declare as vector<datatype> v; and we have to include #include<vector>. It contains the def for vector.
* #include<vector>

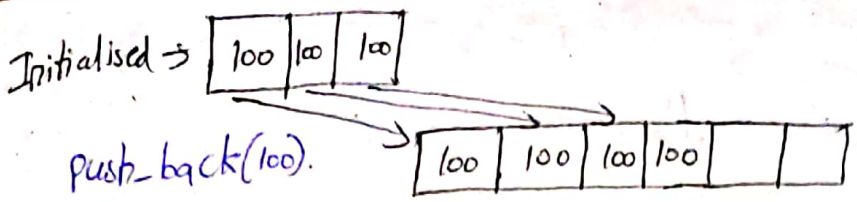
This header file contains all the functions of a vector like search, sort, merge etc.

* v.push_back(x) - push x to vector v.
* emplace_back(x) - Similiar to push_back
* size() - returns the size of the vector.
* pop_back() - deletes the last inserted value.
* If we want to declare same number of element and length to be n [like arr={100,100,100} 3 hundreds], we can declare `vector<int> arr(3,100);` 3 is the length of vector and 100 is the element to be appeared 3 time after initialized.
* If we didn't give second parameters, all zeroes will be given.

vector<int> arr(5,100); - {100, 100, 100, 100, 100}

vector<int> arr(5); - {0, 0, 0, 0, 0}.

Notes: If we declare size like this, after append another element, the size will be doubled and inserted at the position after initialised.



Like this, it always doubles the size of the vector when we push an elements.

* vector<int> arr(ve) - Copies ve elements to arr during the initialisation.

* We can access the elements of a vector using iterator.

vector<int> :: iterator it ;
↑ iterator name

We can use arr.begin() & arr.end() to access the first and last elements.

arr.begin() points to 1st element of arr.

arr.end() points to the last elements next address.

We can take it= arr.begin();

To print address, cout << ⁢

cout << *it; gives the element at that position.

Giving cout << it; throws an error where it is iterator.

To move to another element, we can increment the it to get the next value. [it++]

We can use auto keyword inplace of vector<int>::iterator.

Later it is updated again.

* for(vector<int>::iterator it = arr.begin; it != arr.end; it++)
  cout << *(it);

↓

* for(auto it = arr.begin; it != arr.end(); it++)
  cout << *(it);

↓

* for(auto dit : arr)
  cout << it; //doesn't require *.

↑ it returns the value but not address.

* To find an element in a vector,

(must be declared before) it = find (vec.begin(), vec.end(), element);

cout << it - ve.begin();

↑ address of element

↑ starting address of vector.

Because all elements are stored in continuous locations, the difference of these two will give the index value.

Or element at 1st position.

Starting address-ve.begin() - ve.begin() gives index 0

Day -20 - 26/05/2021 - Arrays & Vectors.

1. To find the index of an element if it is present in vector or not.

auto it = find(arr.begin(), arr.end(), x)

* Now we can find the memory address by &it and element by *it.

* But we know that the difference of continuous indices of an array or vector is 1. So we can get the index by

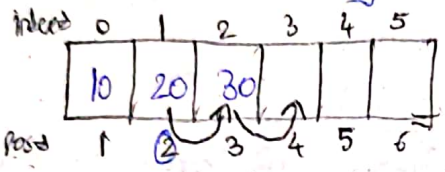
it - arr.begin()

* if the element is not found in the vector, our it will be at the arr.end(). By this, we can write a condition such that if it != arr.end(), we can print the index. If not, we can print -1 in else condition.

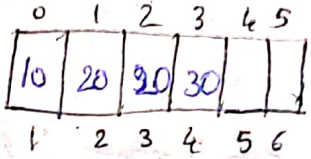
Day-21 - 27/05/2021 - Arrays & Vectors

1. Insert an element at the position "pos" of an array:-

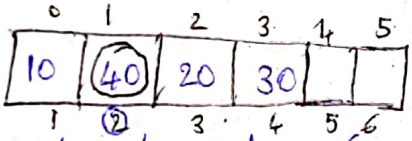
* To insert an element at position, we have to move the elements from pos to n from left to right by one step. [pos = index-1]



⇒



Insert 40 at 2nd position



Add 40 at index 1 (Pos 2)

element copied:

⇒ 40 inserted at 2nd position

2. Insert an element at the position "pos" of a vector:-

* To insert an element into a vector, we can use insert method to insert an element.

SYNTAX:- V.insert(V.begin()+pos-1, n) //insert(position, value)

n is the element, pos = position to be inserted.

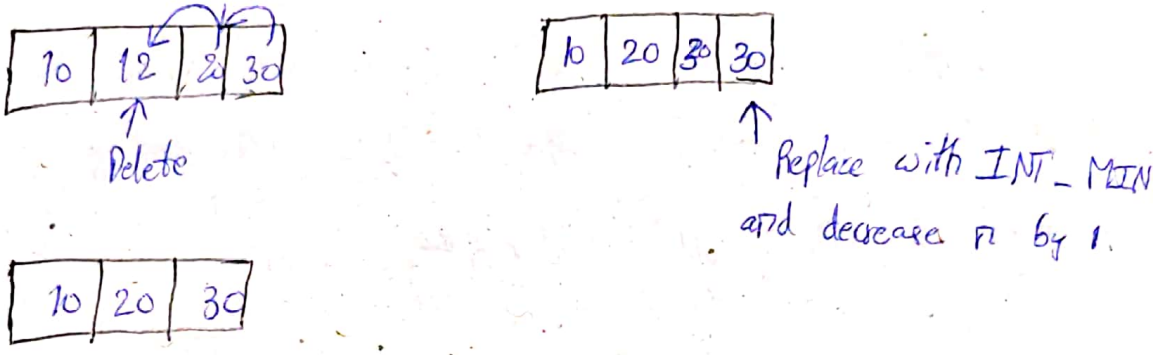
V.begin+pos-1 gives the index of the element's position at which element to be inserted.

* We can pass multiple values as value, to insert into vector.

Day-22 - 28/05/2021 - Arrays & Vectors.

1. Delete an element from an array:-

* Find the index of the element in the array.
* Shift all the elements from index to left by 1.



* We will get the final array after deletion.

2. Delete an element from a vector:-

* We can use ve.erase(itr).
* If index is given, itr = ve.begin() + index.
* If element is given, itr = ve.find(ve.begin(), ve.end, element)

3. Find the second largest element of array:-

val1 = a[0], val2 = INT_MIN.

```
for loop {
    if (a[i] > val1)
    {
        val2 = val1;
        val1 = a[i];
    }
    else if( val2 < a[i]
    {
        val2 = a[i];
    }
}
```

Day-23 - 29/05/2021 - Arrays.

1. To check whether array is sorted or not.

* We can search using two for loops by checking each element with the remaing array.

Optimized

* We can use a single forloop by taking or comparing adjacent (side-by-side) elements.

Day-24 - 31/05/2021 - Sorting.

1. sort() - sort(v.begin(), v.end())

can take any two iterators.

Sort in descending order:- sort(v.begin(), v.end(), greater<int>())

* Bubble Sort:-

Let elements:- 2, 6, 4, 3, 1

Pass:-1

(2) (6) 4, 3, 1

2, (6) (4) 3, 1 (swap)

2, 4, (6) (3) 1 (swap)

2, 4, 3, (6) (1) (swap)

2, 4, 3, 1, 6

Pass:-2

(2), (4), 3, 1, 6

2, (4), (3), 1, 6 (swap)

2, 3, (4), (1), 6 (swap)

2, 3, 1, 4, 6

Pass:-3

(2), (3), 1, 4, 6

2, (3), (1), 4, 6 (swap)

2, 1, 3, 4, 6

Pass:-4

(2), (1) 3, 4, 6 (swap)

1, 2, 3, 4, 6.

∴ Finally the sorted array is 1, 2, 3, 4, 6.

We can ignore last element every iteration as they will be occupied with their actual positions

Day-25 - 01/06/2021 - Functions

Without return type :-

We cannot return values and it is of type void and we can't get values back to the main().

With return type :-

We can return values by declaring return type as its data type and we can get those in the main().

Day-26 - 03/06/2021 - Functions

Void :- The function doesn't return anything is void type.

Note:- The function must be declared above the function call.

Call by reference :-

use the '&' before variables in function argument.

Day-27 - 04/06/2021 - Functions & Recursion

Function with &

While we are using '&' before function name, we cannot create and modify a local variable but we can only change the argument passed to it.

Return array

Using

```
int * fun()
{
    int* arr = { ... };
    .
    .
    return arr;
}
```

in main, int *p = fun(); we can get the array.

By using pointers, we can get the array to be returned by the function.

Most of the times, vector is used instead of int* to return an array.

Recursion:

Function that calls itself. We must give a base case to stop

ex:

```
int fact(int n)
{
    if (i>=1)
    return n*fact(n-1);
    return 1;
}
```

```
int fact(int n)
{
    if(i<=1)
        return 1;
    return n * fact(n-1);
}
```

The if case is base case where the recursion stops

Day - 28 - 05/06/2021 - Explanation on recursion.

Day - 29 - 07/06/2021 - Explanation on recursion

1. Check prime or not using recursion.

```
bool is_prime(int n, int i)
{
    if(n<=2)
    {
        if(n==2)
            return true;
        return false;
    }
    if(n%i==0)
        return false;
    if(i*i>n)
        return true;
    is_prime(n, i+1)
}
```

2. Print n^{th} fibonacci.

```
int fib(int n)
{
    if(n==1)
        return 0;
    else if(n==2)
        return 1;
    return fib(n-1)+fib(n-2);
}
```

```
if(dp[n] != -1)
    return dp[n];
return dp[n]= fib(n-1)+fib(n-2);
```

The dp array is declared globally so that to reduce the time complexity and storing each value at respective positions.

Day-30 – 08/06/2021 – Explanation on recursion

1. Print n to 1

```
Void print(int n)
{
    if (n==0)
        return;
    cout<<n<<" ";
    print(n-1);
}
```

↓

Tail recursion

↓

After child call, nothing to do with parent call.

↓

Less complex

2. Print 1 to n.

We can take two parameters but by changing n to 1 function, we can get..

```
print(n-1);
cout<<n<<" ";
```

} Changing this, we can print 1 to n numbers.

↓

Non-Tail recursion

↓

After child call, there is something to do in parent calls.

↓

More complex.

Day-31 - 10/06/2021 - Recursion

1. To print all the combinations of a given number.

N=4 ⇒ [1,1,1,1 1,1,2 1,2,1 2,1,1 2,2 1,3 3,1] 4

We can use recursion to get this by decreasing N one by one in each recursion and in each we can take another two recursive calls. Base case is N=0 & then we have to print.

We have to for element in loop because we have to remove the last element when back tracking.

Task:

There is a rope, of length N, we have to cut that into parts using lengths given as input a,b,c. Find maximum value of parts, we can get.

N=5, a=2, b=5, c=1.

for a=2, 5 = 2,2,1 [3 parts] (or) 2,1,1,1 [4 parts]

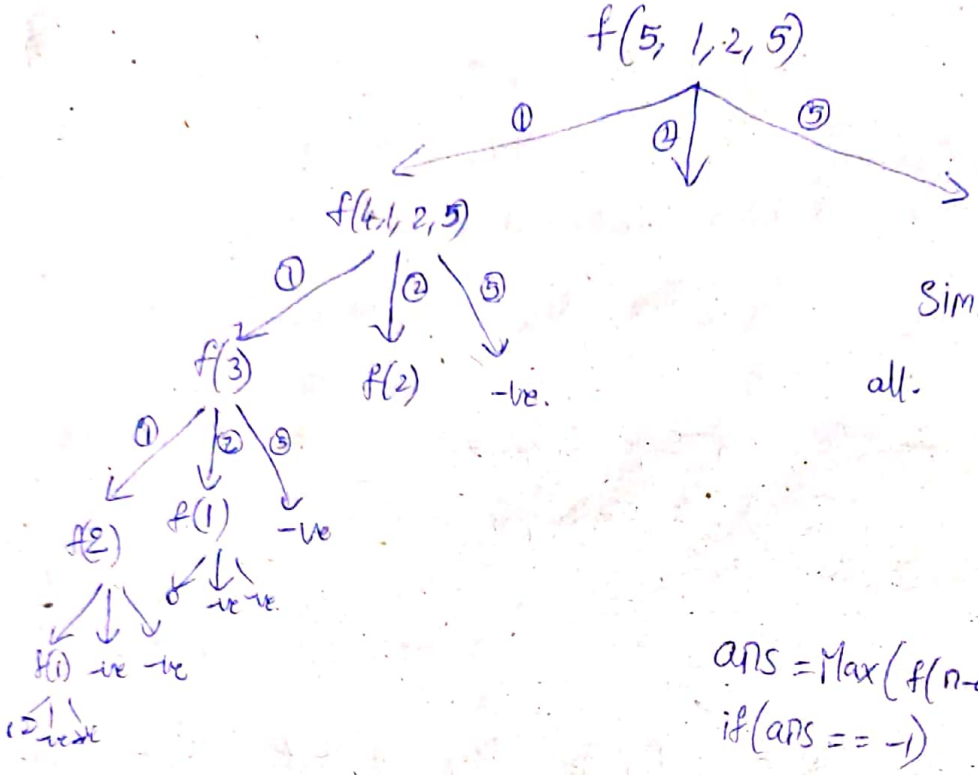
b=5, 5 = 5 [1 part]

c=1, 5 = 1,1,1,1,1 [5 parts] maximum = 5 parts.

Day-32 - 11/06/2021 - Task Explanation

Day-33 – 12/06/2021 – Task explanation, Sub array, Subset

Rope cut: Let n=5, a=1, b=2, c=5.



Similiarly we recurse for all.

```
if(n==0)
    return 0;
if(n<0)
    return -1;
ans = Max(f(n-a), f(n-b), f(n-c));
if(ans == -1)
    return -1;
return ans+1;
```

For f(1), f(0) returns 1 because 0 is present, and 0+1 is 1 and at f(2), f(1) returns 1+1 since 1 is max of all. Thus returns max to top by returning 5 from 1.

Sub-array:

The sub-arrays are the contiguous parts of the array.

ex: Let a={1,2,3},

Sub arrays are:

1
1 2
1 2 3
2
2 3
3

No. of subarrays of array of length n are,

$$\text{Sub arrays} = \frac{n(n+1)}{2}$$

Sub-sets

The every possible combinations of array elements even non-~~contiguous~~continuous elements.

ex: Let $a = \{1, 2, 3\}$

1
1 2
1 3
1 2 3
2
2 3
3.
[ ]

∴ Sub-set count $= 2^n$.

We can get these using power-set.

Let $a = \{1, 2, 3\} \rightarrow 2^{3 \leftarrow n} = 8$.

	Index 2	1	0		
0	0	0	0	→	[ ]
1	0	0	1	→	[1]
2	0	1	0	→	[2]
3	0	1	1	→	[1,2]
4	1	0	0	→	[3]
5	1	0	1	→	[1,3]
6	1	1	0	→	[2,3]
7	1	1	1	→	[1,2,3]

We write two loops, 1st loop to run for 2^n times and inner loop runs for n times. [∵ use $2 << (n-1)$ or $1 << n$]

```
for (int num=0 ; num < (1<<3) ; num++)
{
    vector<int> ans;
    for (i=0; i<n ; i++)
    {
        if (num & (1<<i))
        {
```

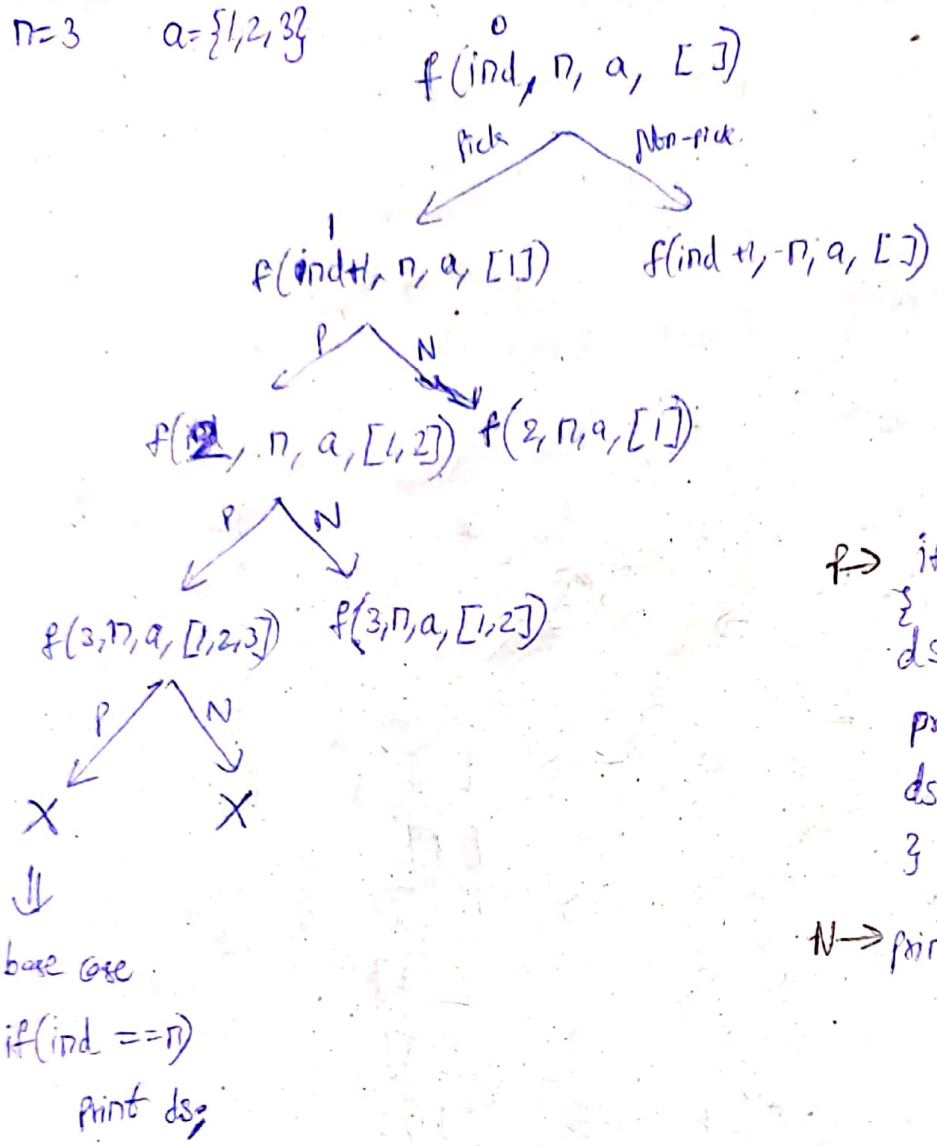
// We are getting indices, so we have to ~~left~~ shift with 1 so we get multiples of 2 to check.

we push the elements.

Note: Time complexity is $O(2^n * n)$.

Day-34 - 14/06/2021 - Subsets using recursion

n=3 a={1,2,3}



base case

```
if(ind == n)
    print ds;
```

it is not necessary because base case already checks.

```
P-> if(ind < n)
    {
    ds.push_back(a[ind]);
    printSubset(ind+1, n, a, ds);
    ds.pop_back();
    }

N-> printSubset(ind+1, n, a, ds);
```

Note: The complexity of this recursion is 2^n. This code is optimised by n times.

Combinations vs Permutations:

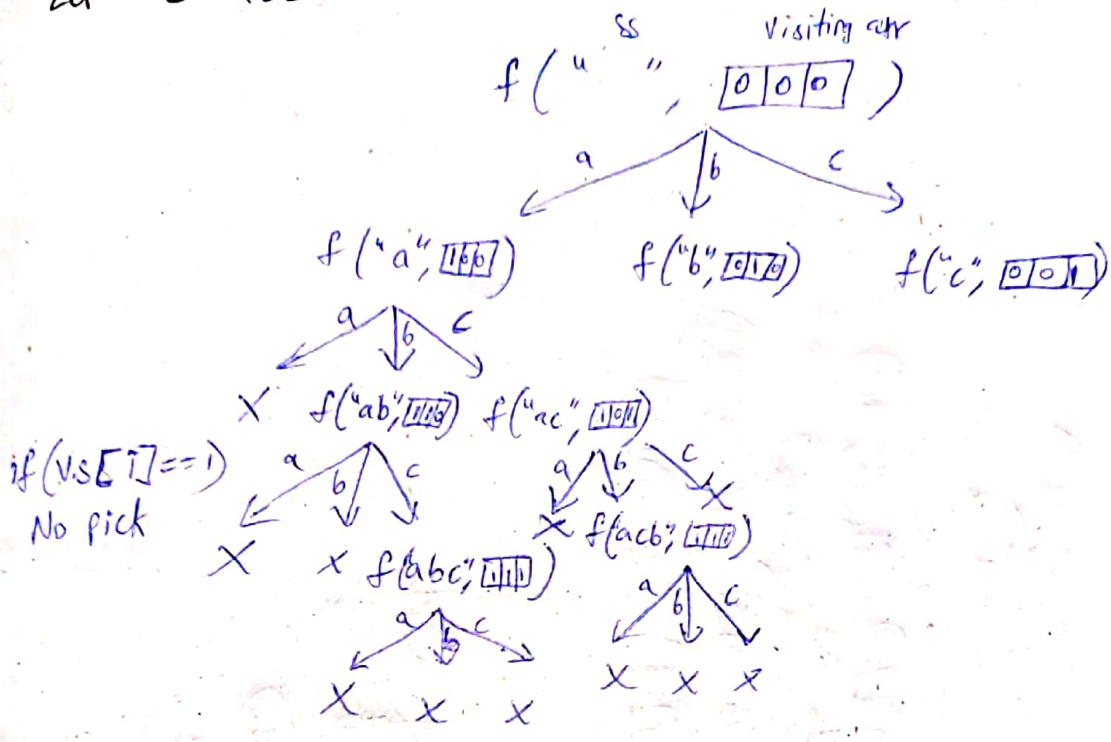
Combinations are selections and permutations are ordered.

Recursion:

1. Clarity on parameters
2. Base case

To print all permutations of a string:-

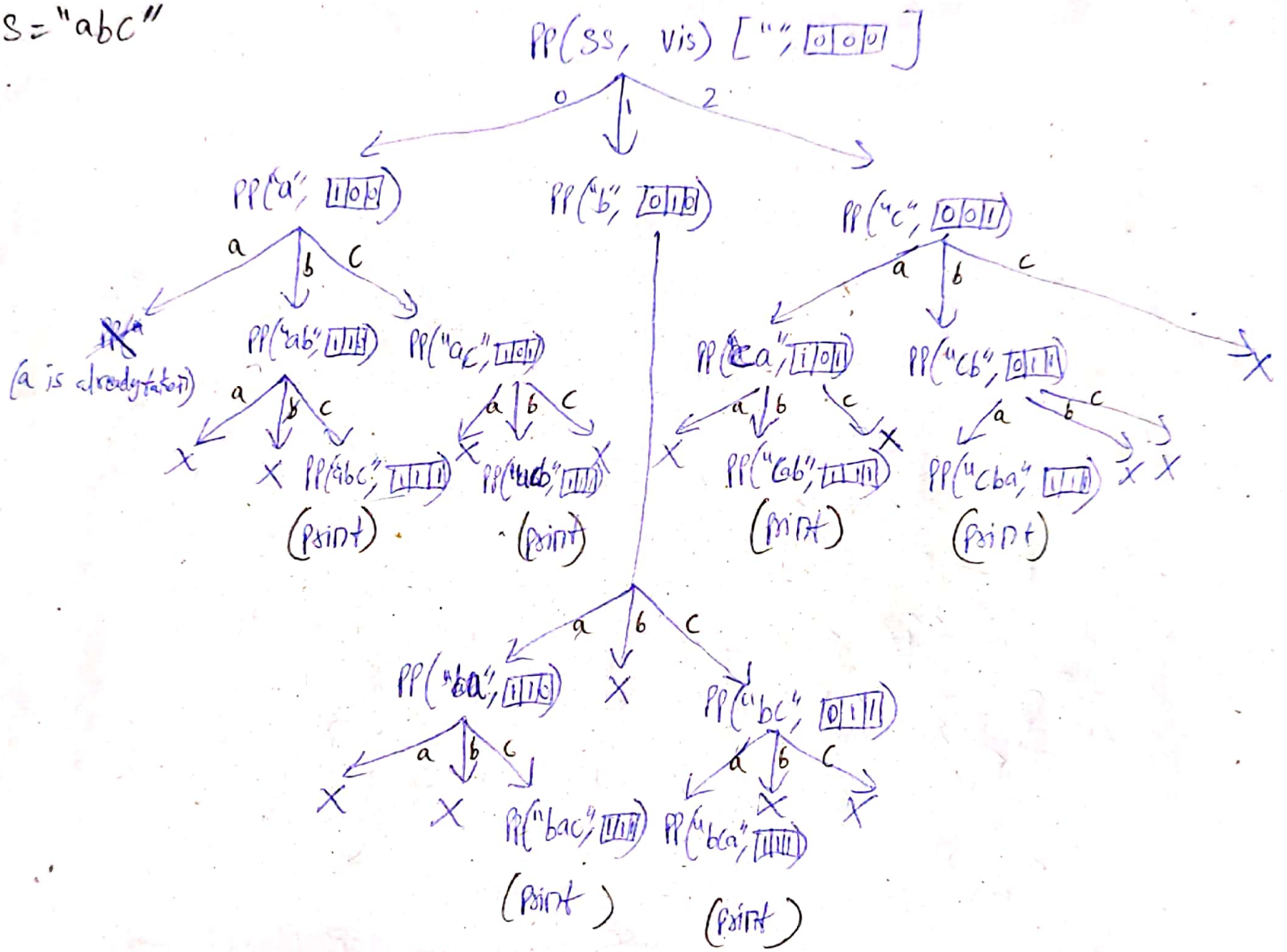
Let S="abc"




```
if( ss.size() == s.size())
    print(s);
for(int i=0; i<n; i++)
{
    if(visited[i]==0)
    {
        ss += s[i];
        vis[i]=1;
        func(ss, vis);
        vis[i]=0;
    }
}
```

Day-35 - 15/06/2021 - Task Explanation & Task Given.

Let S = "abc"



Output:-

abc
acb
bac
bca
cab
cba

Day-36 - 16/06/2021 - ~~This~~ Kadane's Algorithm

1. Task explanation (Maximum Subarray - Leetcode)

ex: [-1, 2, -3, 4, -5]

To find in O(n) using single loop (Kadane's algorithm)

Taking two variables

Tracing:

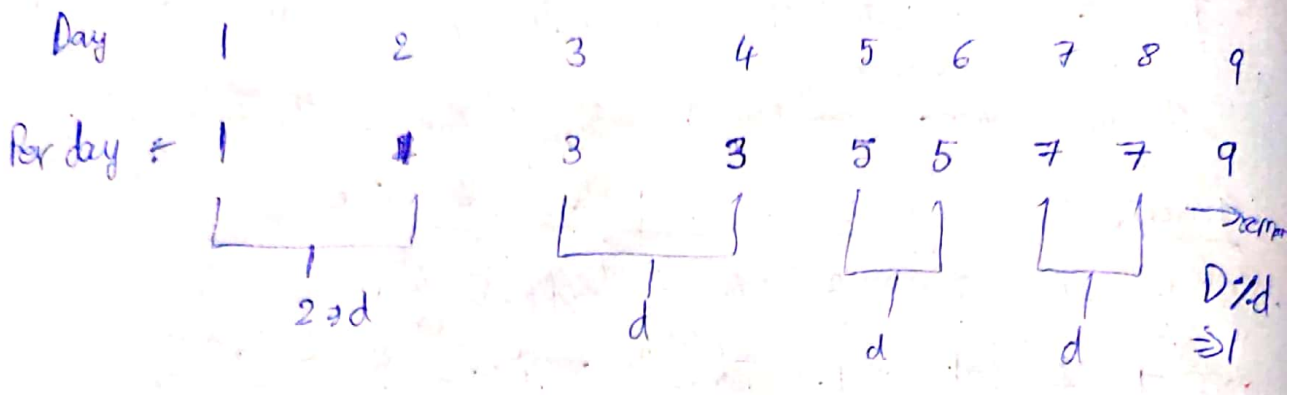
index	0	1	2	3	4
Max_upto (before if)	-1	2	-1	4	-1
Max_so_far	-1	2	2	4	4
Max_upto (after if)	0	2	0	4	0

```
Max_so_far = INT_MIN
Max_upto = 0
Loop i -> 0 to n
    Max_upto += a[i];
    if Max_upto > Max_so_far
        Max_so_far = Max_upto
    if max_upto < 0
        Max_upto = 0
```

Day-37 - 17/06/2021 - Task explanation (CHEFHIEST).

Codechef task:

Let D = 9, d = 2, P = 1, Q = 2.



$$d \times [P + (P+Q) + (P+2Q) + (P+3Q)]$$

1 3 5 7

D/d.

Series in AP → sum of series → $\frac{n}{2}(2a + (n-1)d)$

P, Q

n = D/d [∵ we have D/d distinct series].

∴ sum = d * [n * (2*P + (n-1)*Q)] / 2;

But there is remaining 9th day.

∴ For remaining days, sum = (D%d) * (P + n*Q);

∴ Complete formula is

sum = d * [n * (2*P + (n-1)*Q)/2] + [(D%d) * (P + n*Q)]

or we can write an if condition for D%d to check equals to 0 or not and we can reduce time.

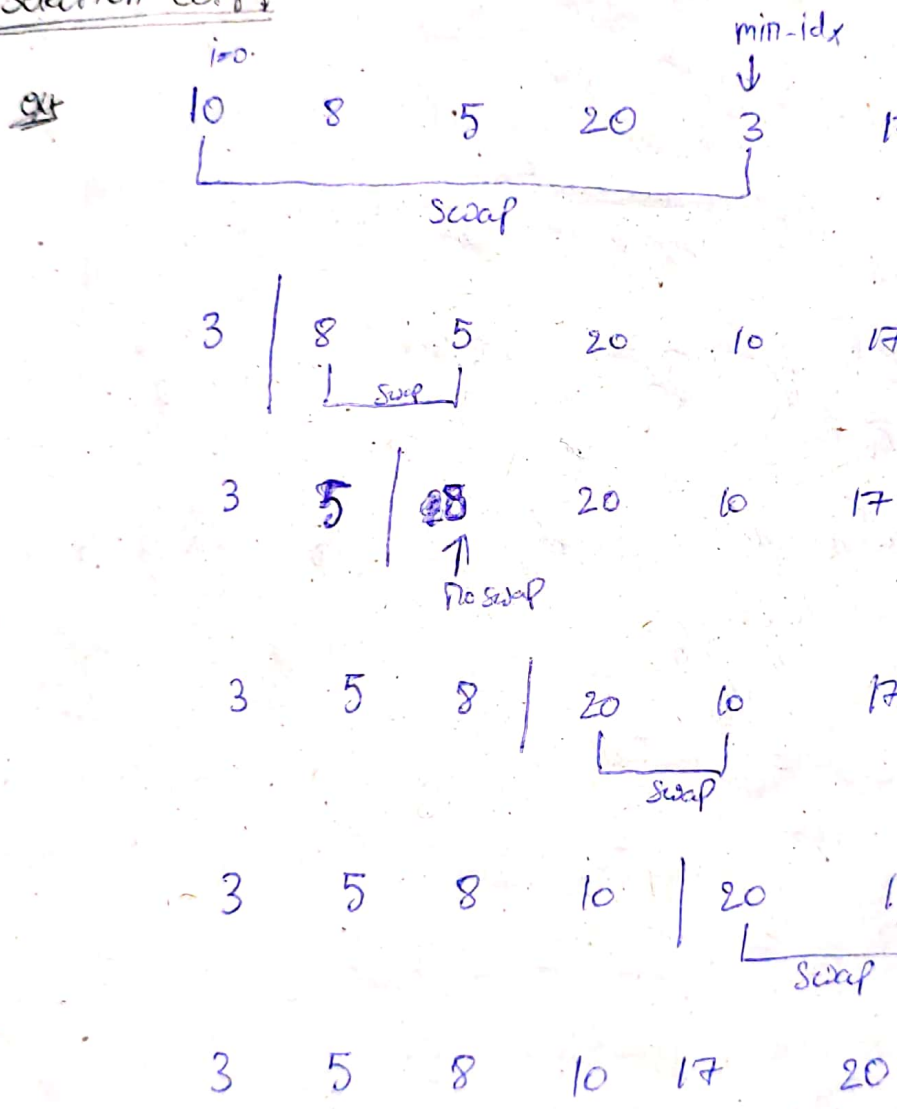
Day-38 - 18/06/2021 - Sorting:-

*Bubble Sort:- Explained on Day 24.

Bubble sort is the stable sort.

ex:- 6 6 6 8, then these are equal elements and it doesn't swaps.

* Selection Sort:-



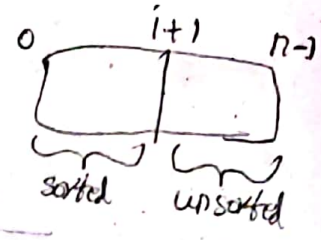
Pseudo-code:-

```
for (i=0 -> n-1)
    min-idx = i
    for(j = i+1 -> n)
        if(arr[j] < arr[min-idx]
            min-idx = j
    swap(arr[i], arr[min-idx])
```

* Selection sort is not stable since it changes the positi-ons are changed for repeated elements.

Day-39 - 19/06/2021 - Sorting.

*Insertion Sort:



Ex: a = {20, 5, 40, 60, 10, 30}

20	5	40	60	10	30
sorted	unsorted				

Swap.

5	20	40	60	10	30
sorted		unsorted			

5	20	40	60	10	30
sorted			unsorted		

5	20	40	60	10	30
sorted				unsorted	

By checking 10 with all, we move 20 to 60 to right.

5	10	20	40	60	30
sorted					unsorted

5 10 20 30 40 60.

Pseudo code:

```
for (int i=1; i<n; i++)
    int key = arr[i];
    int j = i-1;
    while (j>=0 && arr[j] > key)
        arr[j+1] = arr[j];
        j--;
    arr[j+1] = key
```

Note: It is also a stable sort.

*Merge Sort:

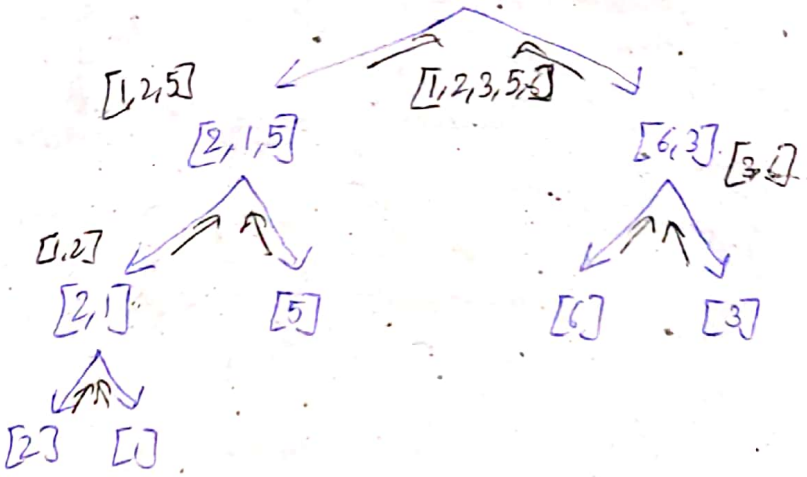
It follows Divide & Conquer algorithm.

ex: arr = {2, 1, 5, 6, 3}
0 1 2 3 4

Complexity
divide into two → $O(\log_2 n)$ → $O(n \log n)$
To merge n elements → $O(n)$

D&C
l = 0
h = n-1
m = (l+h)/2.

l m h
2 1 5 6 3



Then by checking left pointer and right pointer values, we can conquer after dividing.

Merge function:

```
// Starting points for two sub arrays
left = low, right = mid+1.
while( left <= mid && right <= high)
    if (arr[left] < arr[right])
        vector.push_back(arr[left]);
        left++;
    else
        vector.push_back(arr[right]);
        right++;
// To copy remaining elements if
while (left <= mid)
ver.push_back(arr[left]);
    left++;
while(right <= high)
ver.push(arr[right]);
    right++;
```

Sort function:

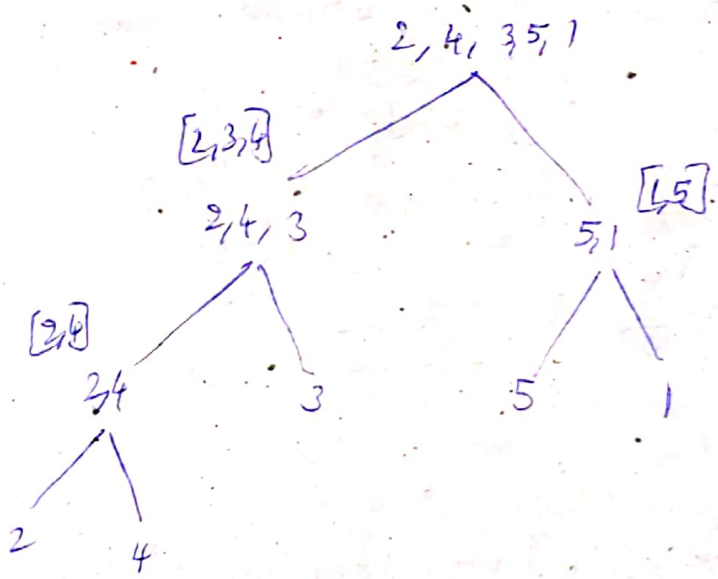
```
if (low = high)
    return;
mid = (low+high)/2;
sort(low, mid, arr)
sort(mid+1, high, arr)
merge(low, mid, high, arr)
```

```
// Firstly insert sorted elements to array.
for (int i=low; i<=high; i++)
    arr[i] = vector[i-low].
```

Day-40 - 22/06/2021 - Problems on Merge sort.

1. Leetcode - ~~Reverse pair~~ Local and global conversions.

Take 2,4,3,5,1



At first, 2,4 and 2, 2<3 ∴ left++.

4>3 ∴ count=1.

Similarly.

L mid R
2,3,4 1,5

* Here 2>1 ∴ 2,3,4 are sorted array now and all can form a pair with 1. ∴ we add the mid-left+1 to count.

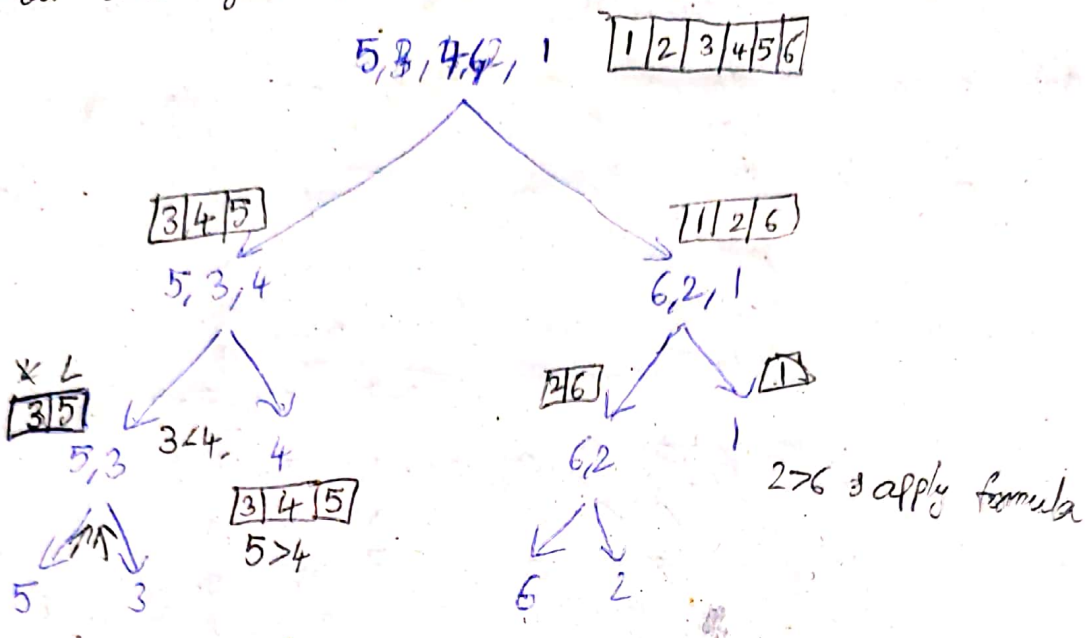
∴ count += 3 [∵ value equals]

* Now we compare with R+1, ∴ 5 and check until we get the less than within loop.

* By using conditions, we can increase either local or global. Now if global=0, then return true else false.

Day-41 - 23/06/2021 - Problems on MergeSort

1. Local and global conversions



count = mid-low+1 [L>R].

At final,

L>R ⇒ all in left array could be greater as it is sorted.

similiar to above.

Here 3<6 ⇒ L++, 4<6 ⇒ L++, 5<6 [no more]

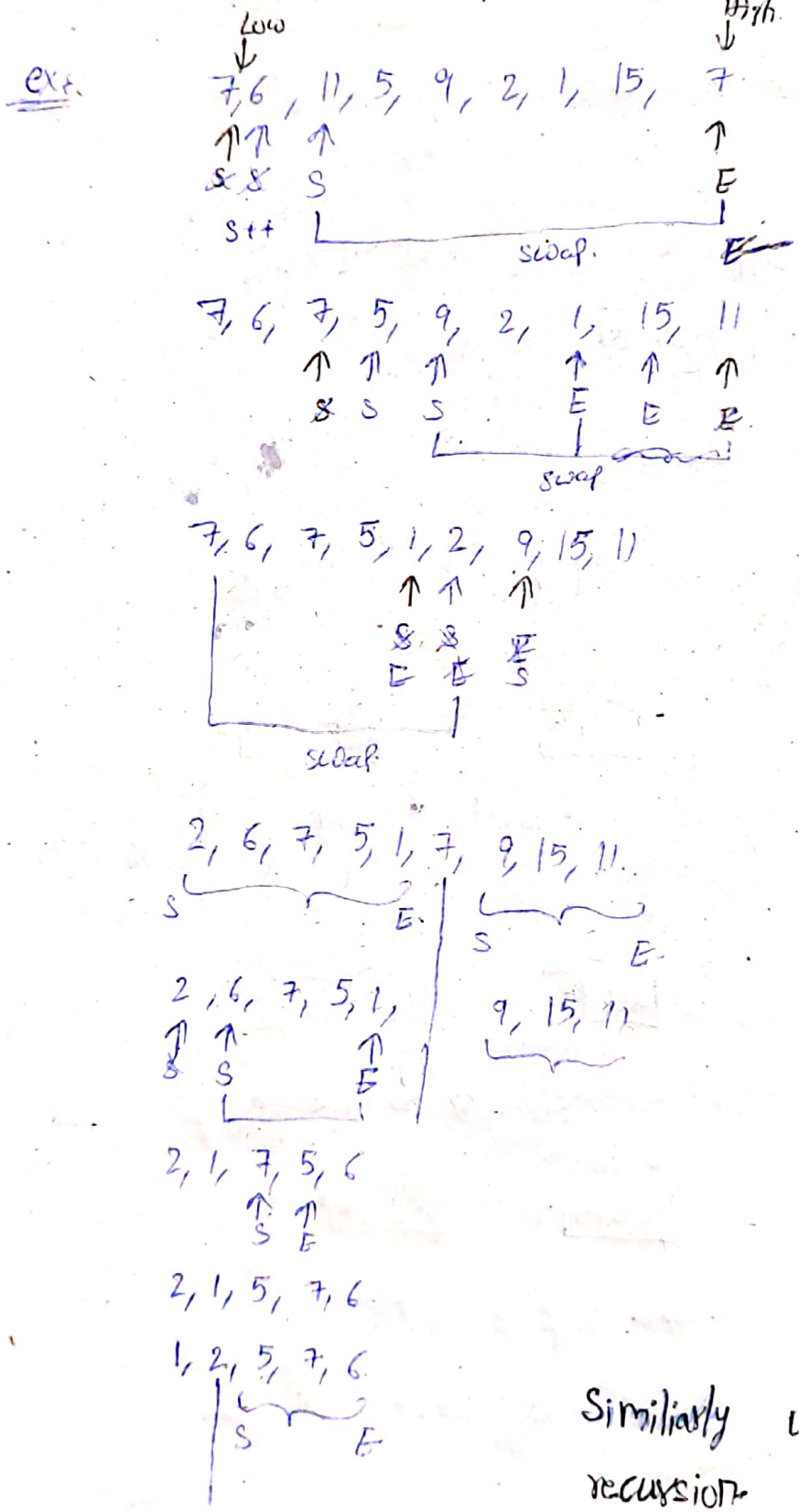
Finally sorted and count will be 11.

Contd...

Quick Sort

Arr = { 10, 12, 2, 3, 8, 15, 11 }.

We take an element as pivot and sort using it.



Similiarly works in the recursion.

Day-42 - 24/06/2021 - Searching

* Linear Search:

Loop from 0 to n-1 elements.

```
Loop i -> 0 to n
    if (a[i] == target)
        break;
```

Complexity is O(n). But can be applied on any type.

* Binary Search:

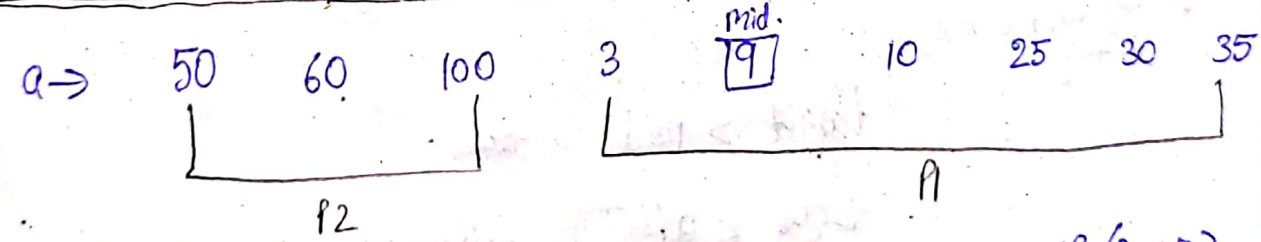
By taking mid values and checking.

Complexity is O(logn) but can be applied on sorted array.

We can take $low + \frac{(high-low)}{2}$ [>> 1].

Because of very large numbers.

* Binary search on Rotated Sorted array:




```
if (a[mid] == target) return mid;
else if (a[mid] > a[low])   //exists in part 1.
    if (target > a[mid] || target < a[s])   // in part 2
        low = mid+1;
    else
        high = mid-1;
else
    if (target >= a[0])   // in part 2 [also || target < a[mid]]
        high = mid-1;
    else
        low = mid+1;
```

if (9 < 50)
if

Day -43 — 25/06/2021 — Binary Search for rotated Sorted array.

	Part-1			Part-2					
index	0	1	2	3	4	5	6	7	8
array	50	60	100	3	9	10	25	30	35
	Low				mid				High

mid = ((Low + High)/2);

* Target = 25

if (arr[mid] == target) return mid;

check target with mid ⇒ target > mid

check target with 0th element ⇒ target < 0th elem.

So target is in part 2 because the element less than 0th will be at right part.

⇒ if (target > arr[mid] && target < arr[0])

low = mid+1; (or) target > 0

Now mid = (5+8)/2 = 6

⇒ arr[mid] = target [∴ found at pos 6]

* Target = 100

target > mid true.

target < a[0] false.

∴ high = mid-1. = 3

mid = (0+3)/2 = 1.

Here they are at same part

⇒ low = mid+1.

mid = (2+3)/2 = 2

arr[mid] = target.

* Target = 3 :

target < mid

high = mid - 1. = 3.

mid = 1

if (target < mid && target < a[0]) also.

low = mid + 1.

mid = 2.

target > mid & target < a[0]

low = mid + 1.

mid = 3.

arr[3] = target

found at pos 3.

Day-44 : 26/06/2021 - Task Explanation

1. Leet code Sort colors :

* By counting 1's & 2's & 0's and append to a new array or vector and add those using while loop.

* 2 1 1 2 0 1 0

L m h

if m=2
swap(m,h)
h--;

m=0
swap(m,L)
m++;
L++;

m=1
m++;

swap(m,L) ⟹

0 1 1 2 0 1 2
m h
L

0 1 1 2 0 1 2
m h
L

0 1 1 2 0 1 2
L m h

0 1 1 1 0 2 2
L m h
m++

0 1 1 1 0 2 2
L m
h

0 0 1 1 1 2 2

2. Task : nth magical number in leetcode.

a=2, b=4, n=5. 2, 4, 6, 8, 10, 12.

L m h

The seq will be from low(a,b) to min * n.

If a=4, b=6, n=4.

4 6 8 12 16

2 4.

in the loop.

$$c = \frac{mid}{a} + \frac{mid}{b} - \frac{mid}{Lcm(a,b)}$$

Here c = 3 ⇒ nums[c] = 12.

Mid = (l+h)/2 = 10.

```
if(c == n)
    if(m%a==0 || m%b==0)
        return mid;
else if(c<n)
    low = mid+1;
else
    high = mid-1;
```

Day-45 - 28/06/2021 - AWS SDE Question.

*** allocation of books :-** The maximum pages allocated should be minimum.

Rules:

* A book will be allocated to only one student.
* Each student min of 1 book.
* Allocation should be in a continuous order.

ex: n=4, a={12, 34, 67, 90}, Sf=2.

S1	S2	
12	34, 67, 90	⇒ max = 191
12, 34	67, 90	⇒ max = 157
12, 34, 67	90	⇒ max = (113) minimum possible

Pseudo code:

```
int low = min(array)
int high = max(array)
int ans = -1
while (low <= high)
    mid = (low+high) >> 1;
    if (isPossible(array, mid, students)) {
        ans = mid;
        high = mid-1;
    else
        low = mid+1;
return ans;
```

```
bool isPossible(array[], barrier, std)
{ int count = 0;
  int sumofalloc = 0;
  for(int i=0; i<n; i++)
  {   if(sumofalloc + array[i] > barrier)
      {  count++;
         sumofalloc = array[i]
         if(sumofalloc > barrier)
            return false.
      }
      else{
         Sumofalloc += arr[i];
      }
  }
  if(count > Students)
      return false;
  return true;
```

Day-46 – 29/06/2021 – Prefix Sum Algorithm.

*Range problem:

If $a = \{20, 15, 17, 34, 67\}$

If we give queries (number) and low & high,

Low	High		
1	3	= 66	[sum]
0	4	= 153	
2	3	= 51	

Bruteforce:

*To use this algorithm, complexity is $O(n^2)$, in worst case, $n = 10^6$, $q = 10^6$ ⇒ $O(n^2)$ ⇒

$O(10^{12})$ ⇒ TLE

little optimised:

* We can solve using binary search but it also gives TLE.

Highly optimised: (Using Prefix Sum)

By creating prefix array = { 20, 20+15 = 35, 35+17 = 52, 52+34 = 86, 86+67 = 153 }

Now to get sum from 1 to 3, we have to remove 0th sum because our sum should contain element at 1.

But we have to check for 0th index because 0th is first, so if we remove 1 from 0, it becomes -1.

```
: if (low == 0)
      return prefix[high];
  return prefix[high] - prefix[low-1];
```

In worst, O(2×10^6) ⇒ O(n+2).

Day-47 - 9/07/2021 - Prefix Sum Algorithm

a = {2, 6, 1, 4, 8, 7}

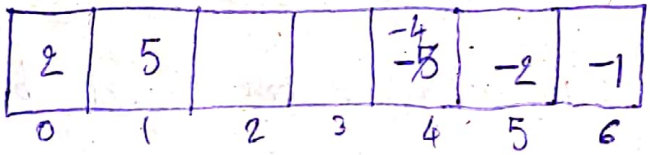
[ios_base::sync_with_stdio(false);
cin.tie(NULL);
Use to make fast input.

Queries	left	right	num	
	0	4	2 [add]	[4, 8, 3, 6, 10, 7]
	1	3	5	[4, 13, 8, 11, 10, 7]
	4	5	1	[4, 13, 8, 11, 11, 7]

Adding the number from left to right to all each. like above example.

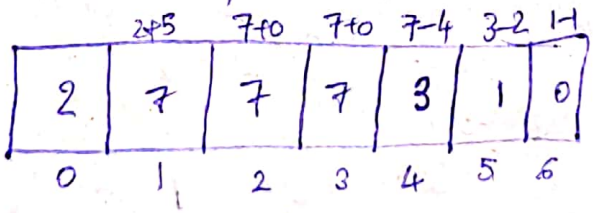
By using bruteforce, we will get a TLE.

Optimized:-



add K at arr[left], -K at arr[right+1]

∴ To get prefix array,



So by adding this prefix array to array, we get the resultant array.

Day-48 - 02/07/2021 - GFG Equilibrium Point

If a = {1}, n=1, output = 1 [∵ only one element, position returned]

If a = {1, 3, 5, 2, 2}, n=5, op = 3 [∵ 1+3 = 4, 2+2 = 4 & return 5 position]

Brute force:

If n ≠ 1, the 1st element and last element cannot be the equilibrium points.

```
in for i -> 1 to n-1
    for j -> 0 to i
        Left sum += arr[j];
    for j -> i+1 to n
        Right sum += arr[j];
    if ( Left sum == Right sum)
        return i+1;
return -1;
```

Optimized: (Prefix Sum).

Prefare a prefix array.

```
if(n==1)        if(n==2)
return 1;       return -1;
for(int i=1 ; i< n-1; i++)
    if( prefix[i-1] == prefix[n-1] - prefix[i])
        return i+1;
return -1;
```

Here if 1, 3, 5, 2, 2 is array, 1, 4, 9, 11, 13 Prefix.

Here if i=2, prefix[i-1] = 4 [sum upto i-1],

prefix[n-1] - prefix[i] = 13-9 = 4 [sum from i+1 to n]

Day-49 - 03/07/2021 - GFG Explanation (Coding Contest)

Day-50 :- 05/07/2021 - Subarray Sums equals k (Prefix sum)

4	-2	-1	1	-2	3	-3

Prefix: 4 2 1 2 0 3 0 a=0.

x=4

x-a=

Take a map with key as elements of prefix and value as count.

if (Prefix[i]-k) is in dict(map).

count += dic[Prefix[i]-k]

dic[Prefix]++;

Day-51 - 06/07/2021 - Subarray sum divisible by K (Prefix sum)

Let {4, 5, 0, -2, -3, 1} K=5.

4 → 4
4,5 → 9
4,5,0 → 9
4,5,0,-2 → 7
4,5,0,-2,-3 → 4
4,5,0,-2,-3,1 → 5

5 → 5
5,0 → 5
5,0,-2 → 3
5,0,-2,-3 → 0
5,0,-2,-3,1 → 1
0 → 0
0,-2 → -2

0,-2,-3 → -5
0,-2,-3,1 → -4
-2 → -2
-2,-3 → -5
-2,-3,1 → -4
-3,1 → -2
-3 → -3
1 → 1

count = 7.

Brute force:

By writing three loops, we can calculate each subarray sum and check if it is divisible by k.

* It gives TLE.

optimised (Prefix sum):

Prefix array is: 4 9 9 7 4 5. K=5.

count = ~~0 1 3 6~~ 7

4%5 = 4
9%5 = 4 } equal → 9-4=5, 5%5=0 divisible
9%5 = 4.
7%5 = 2
4%5 = 4. → incr.
5%5 = 0

dic[sum]++;

Key	value
default → 0	✗ 2
4	✗ 2 3 4
2	1

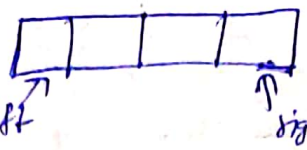
count += Prefix[sum]

Note:

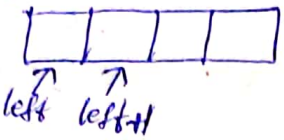
In CPP, if we execute -9%5 gives -4 & -1%5 = -1
But in python, -9%5 is 1 & -1%5 is 4.
So we have to do (-9%5)+5 because we have to get the positive keys for our map. E: (x%K)+K

Day-51 - 07/07/2021 - Two Pointer

Opposite direction:-



Unidirection:-



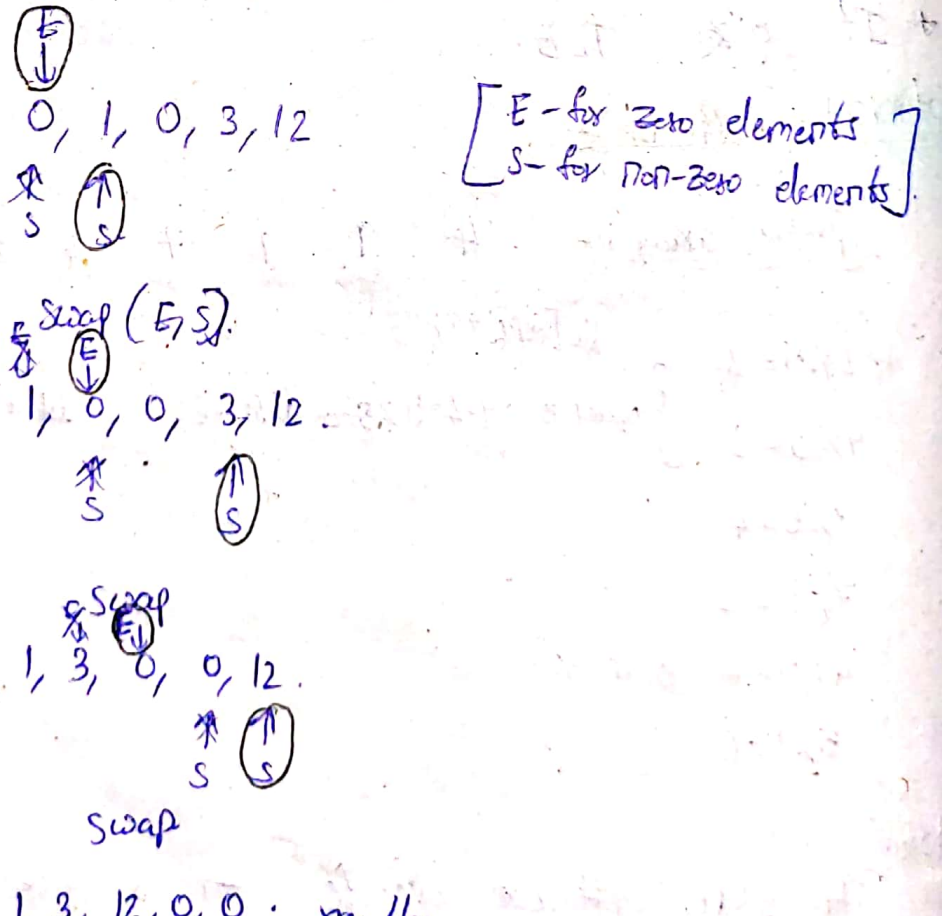
Day-52 - 09/07/2021 - Two pointer.

Move Zeroes (Leetcode):-

Let nums = {0, 1, 0, 3, 12}

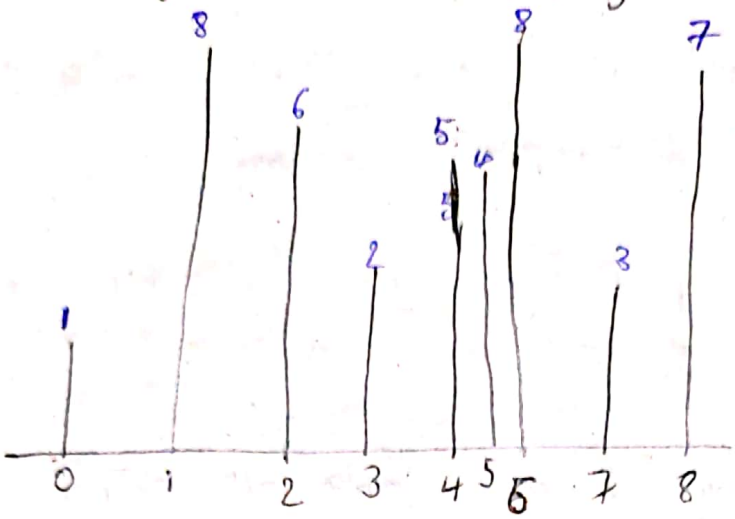
we have to move all zeroes to last without changing order of others.

nums = {1, 3, 12, 0, 0} will be output.



container with most water :- (Leetcode).

{ 1, 8, 6, 2, 5, 4, 8, 3, 7 }



formula = (1-0) x min(1,8). like that.

we can take start at 0th, end at 8th.

∴ formula = (E-S) x min(nums[E], nums[S]).

if nums[E] is less

E--

else

S++

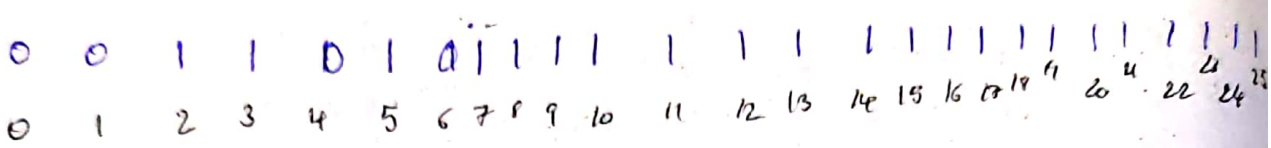
We get max (8-1) x 7 = 49.

Day-53 - 10/07/2021 - Sieve of Eratosthenes.

To get all primes less than n

Create an array of length n+1. (ex n=25).

a[0], a[1] = 0. [∵ not prime] all are 1's.



We check from 2 and make all of multiples 0.

3 and make all 3 " 0.

4 is already 0 so ignore.

5 and make all of multiples 0.

until sqrt(n), here $\sqrt{25} = 5$. so we can stop.

Now we get, 2 3 5 7 11 13 17 19 23

are primes.

* Take vector<bool> sieve(N+1, true) [∵ N is number, initially all are prime]

```
sieve[0] = false
sieve[1] = false
for(int i=2; i*i<=N i++) {
    if (sieve[i]) {
        for(int j=i*i; j<=N; j+=i) {
            sieve[j] = false;
        }
    }
}
```

Note: We take j = i*2 but for 3*2 is already false, 5*2, 5*3 will already be false, so we check from i*i.

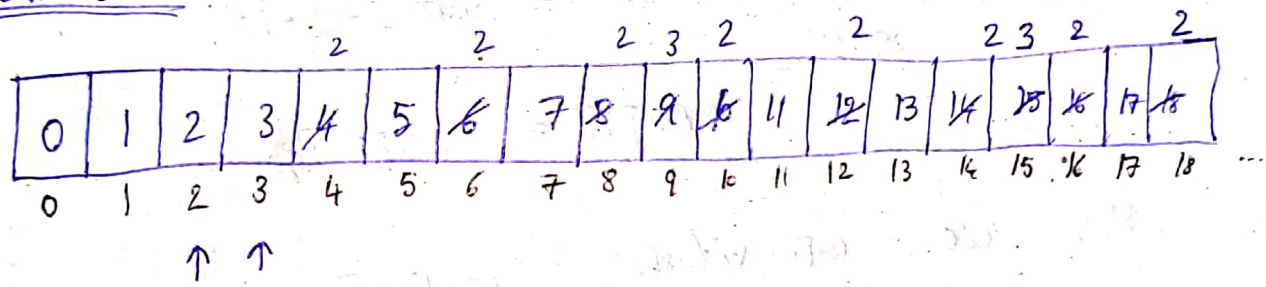
Day-54 - 13/07/2021 - Sieve Problems

Number of primes (hackerearth)

We can create a prefix, we can count easily. if given queries. else we can count in Sieve array.

Day-55 - 14/07/2021 - Finding Prime factors.

create sieve:



We change all like this.

Now for 100 → Sieve[100] = 2. add 2 to factors,

∴ 100/sieve[100] = 50 " 2.

50/sieve[100] = 25 " 5 at [25]

∴ 25/sieve[25] = 5 5 at [5].

5/sieve[5] = 1.

we get 1 so get out of loop.

∴ 2 x 2 x 5 x 5 are factors

```
Sieve[0] = Sieve[1] = 0.
for(int i=2; i<=N; i++) Sieve[i] = i;
for (int i=2; i*i<=N ; i++)
    if(sieve[i]==i)                    //only primes can be inserted.
    for(int j= i*i; j<=N; j+=i)
        if(sieve[j]==j) //if not equal, we can't change it
            sieve[j] = i;
```

Day-56 - 15/07/2021 - Sieve problems

1. Sum of primes :

We can generate sieve and also a prefix array. taking sums[0] = sums[1] = 0.

```
for i -> 2 to N
    if (sieve[i])
        sums[i] = sums[i-1] + i;
    else
        sums[i] = sums[i-1];
```

(or)

```
for i -> 2 to N
    sums[i] = sums[i-1]
              + i * sieve[i];
```

if sieve is 0, not added.

Now we can return sums[end] - sums[start-1]

where start is starting prime ; end is ending prime.

We take start-1 because we have to use start value also in the sum to get complete sum [= inclusive [start, end]]

Day-57 - 16/07/2021 - Strings

Day-58 - 17/07/2021 - Strings

Day-59 - 20/07/2021 - Strings

Day-60 - 21/07/2021 - Strings

Day-61 - 22/07/2021 - Strings

Day-62 - 26/07/2021 - Data Structures in STL.

Array:

```
assay < int, len> a;
a.fill(10); // inserts 10 at all index values of a.
```

We can access elements using the methods of vector.

Day-63 - 27/07/2021

Set: Collection of unique elements in a sorted order.

```
set < int> st;
```

* We can use insert(), emplace() [faster]. We cannot use index. We can iterate as same as vector.
* st.erase (iterator1, iterator2); both are inclusive iterators.
* st.clear(); clears the set completely.
* Any set function takes O(log n).

Unordered set:

* Same as set but unsorted set.
* unordered_set <int> st;

Multiset:

* multiset < int > mt;
* Stores multiple occurences in a sorted manner.
* mt.count(elem) // to find no. of occurences of an element.

Map:

* Key → value pairs. [Sorted keys and unique keys]
* can insert using mp.emplace(key, val);
* Time complexity O(log n)

Day-62 - 28/07/2021 - STL Functions.

Unordered Map:

* Same as map but unsorted. unordered_map<int, int> data;
* Time complexity in most case O(1) but in worst O(n).

Multimap:

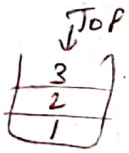
* Same as map but takes duplicate keys. multimap<int, int> ds

Pair class:

* pair<int, int> pr = {1, 2};
* pr.first; pr.second;
* We can nest a pair inside a pair.

Stack:

* stack<int> stk; --> Last In First Out
* st.push(val);
* st.pop();
* st.top() --> returns last inserted.
* st.empty(); --> returns empty or not.



Queue:

* queue<int> qu; ---> First In First Out
* qu.push(val);
* qu.pop(); --> removes front.
* qu.front();

Priority Queue:

* priority_queue <int> pq; Store in reverse.
* pq.push(val);
* pq.pop();
* pq.top();

Day-65 - 29/07/2021. - Stack & Queue implementation.

Day-66 - 31/07/2021 - Queue implementation

Day-67 - 02/08/2021 - BFS Algorithm

BFS Algorithm:

① Implement Queue

② Try out all the combinations

③ When reach the end, stop.

Problem:

1. Given start, end, array.

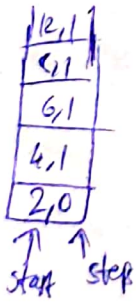
The least combination of multiplication of numbers to get from start to end. use %100000 to get limited values.

Let start = 2, end = 24, arr = {2, 3, 4, 6}

⇒ 2 × 2 = 4

4 × 6 = 24 ⇒ opt 2. [∵ least steps]

BFS:

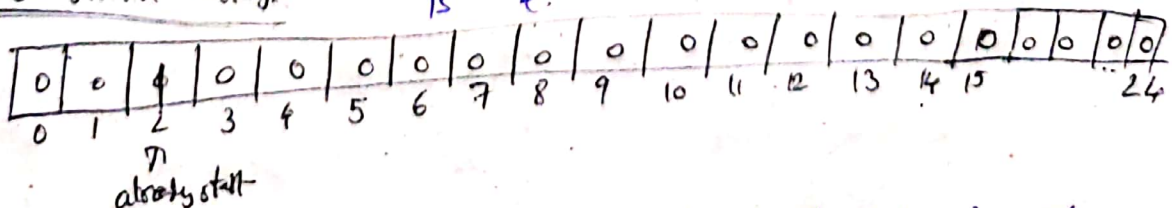
① 

2x ← all elements, push to queue and make visited.

check all. if not visited, add to queue.

finally we get after 2, next queue front is 4.

Take visited array:



Now all combinations with 4, we get 4×6 = 24. is end.

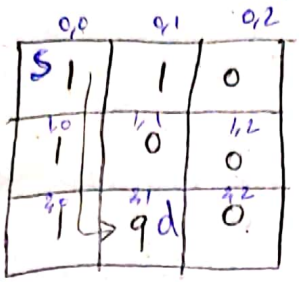
now we reached end, steps at 24 are 2 ⇒ opt 2.

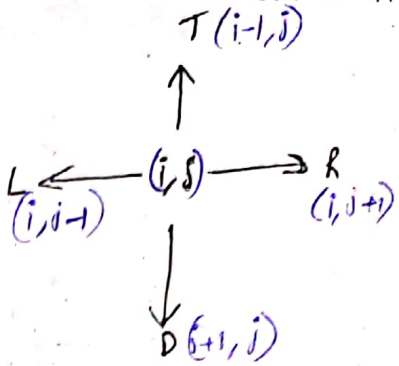
Code ⇒ contd in

```
int BFS(int start, int end, int a[], int n) //n is length of arr.
{
    int visited[100000] = {0};
    visited[start] = 1;
    queue<pair<int, int>> Q;
    Q.push({start, 0});
    while(!Q.empty)
    {
        int newstart = Q.front().first, steps = Q.front().second;
        Q.pop();
        for(int i=0; i<n; i++)
        {
            int req = (newstart * a[i]) % 100000;
            if(req == end)
                return steps+1;
            if(!vis[req])
            {
                Q.push({req, steps+1});
            }   }   vis[req] = 1;
    }
    return -1;
}
```

Day-68 - 03/08/2021 - BFS Problem

Problem: To find the minimum path from store to home. If there is a route, then will be 1 else 0. destination is 9.





One can move in 4 ways. Top, bottom, left, right.

Now we will take a bool function isvalid to check whether the coordinate lies in grid or not and if the position is valid to move (1) or not. Then return true or false to make it valid path.

Then check if visited or not. if not visited, add to queue and make it visited.

Code:

```
Void amazon()      // we can pass parameters, here taking hard coded values
{
int matrix[3][3] = {{1,0,0},{1,1,0},{1,9,0}}, n=3, m=3;
queue < pair < pair<int, int>, int>> Q;   [ Queue of {{1,0}, 1}  ]
                                            node  count
if (matrix[0][0] == 0)
{
        cout << -1;
        return;
}
if (matrix[0][0] == 9)
{
        cout << 0;
        return;
}
Q.push({{0,0},0});
int vis[m][n];
memset(vis, 0, sizeof vis);
```

```
vis[0][0] = 1;
int di[] = {-1, 0, 1, 0};
int dj[] = {0, 1, 0, -1};
while (!Q.empty())
{
    int i = Q.front().first.first, j = Q.front().first.second;
    int steps = Q.front().second;
    Q.pop();
    for (int k=0; k<4; k++)  //4 ways  T B L R
    {
        int newi = i + di[k];
        int newj = j + dj[k];
        if (isvalid(newi, newj, matrix) && !vis[newi][newj])   // At last
        {
            if (matrix[newi][newj] == 9)
            {
                cout << steps+1;
                return;
            }
            vis[newi][newj] = 1;
            Q.push({{newi, newj}, steps+1});
        }
    }
}
cout << -1;
}
bool isvalid(int i, int j, int matrix[][3])
{
    if (i >= 0 && i < 3 && j >= 0 && j < 3 && matrix[i][j] != 0)
        return true;
    return false;
}
```

Day-69 - 05/08/2021 - Circle Problem.

Find intersection points of two circles.

print all the possible points in the intersection region of two circles even on border.