21st Sept 2019

Array

C++ | Java

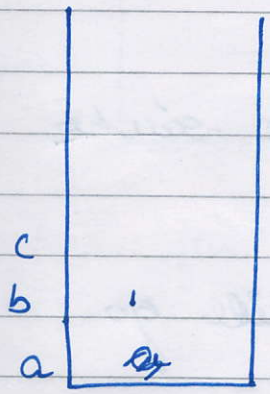
int arr[100]; | int [] arr = new int [n];

stack is being alotted by operating system.

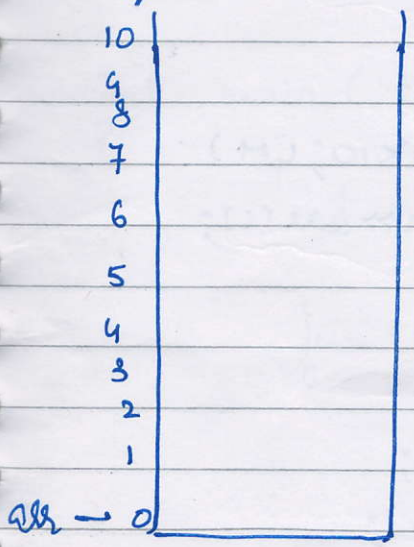
Memory allocation.

stack heap.

int a, b, c, ---



padding concept in stack where empty space is present between variable in stack

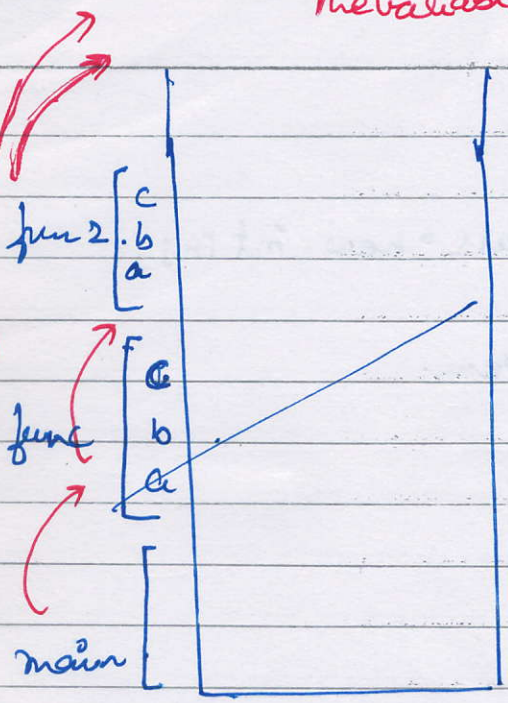


int = 4 byte.

while moving up variable is being created
when function is coming down all the variables will be dead with time.

fun1.cc fun2.cc.

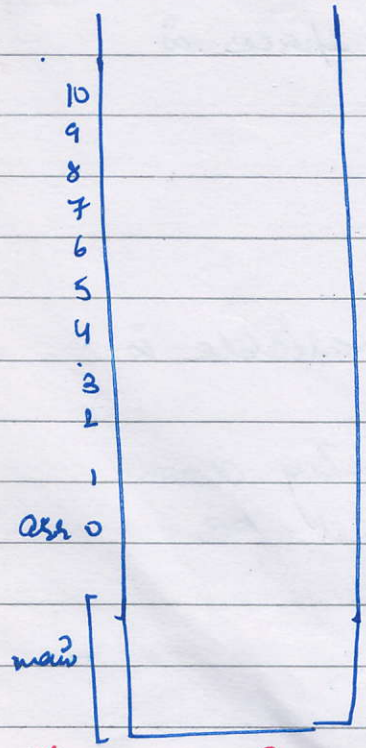
Me variables different according to OS.




```
void fun2(a,b,c)
{
    cout<<a+b-c;
}

void fun()
{
    int a,b,c;
    fun2(a,b,c);
}

int main().
{
    fun();
}
```

→ once fun is completed the variables a,b,c will be wipped out.

→ once the function completes its task it will go back to its parent.



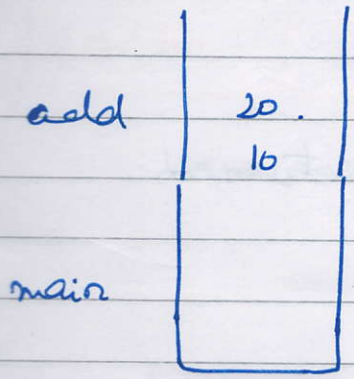

```
int[] fun(){
{
    int arr[10];
    for(int i=0; i<10; i++)
    {
        cin>>arr[i];
    }
}

int main() {
    int arr[];
    arr = fun();
```

This code will work but when will try to access the element of arr it would be different because the arr is already

100 | 36 (136) 40

100
70
30
200 | 20

copy constructor → whenever a function returns something or argument is passed for function. copy constructor is invoked



→ when main is calling add() it is actually copying the values of variable. (by copy constructor)

int *fun(int n) → pointer to first index is being returned (i.e base address)
{
    int arr[n];

    return arr;
}

int main()
{
    int n=10;

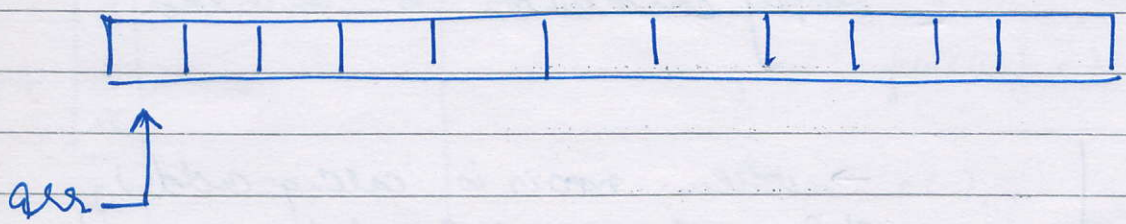
    fun(n);

}

byte: 4 8 1

int, long, char, string } primitive

* when you return it, it is being copied.



base

in Non-primitive ^ address is being returned.

```
void fun (int arr[])
{
    for(int i=0; i<20; i++)
    {
        cout<<arr[i]<<" ";
    }
}

int main()
{
    int n=10;
    int arr[n];
    fun(arr);
}
```

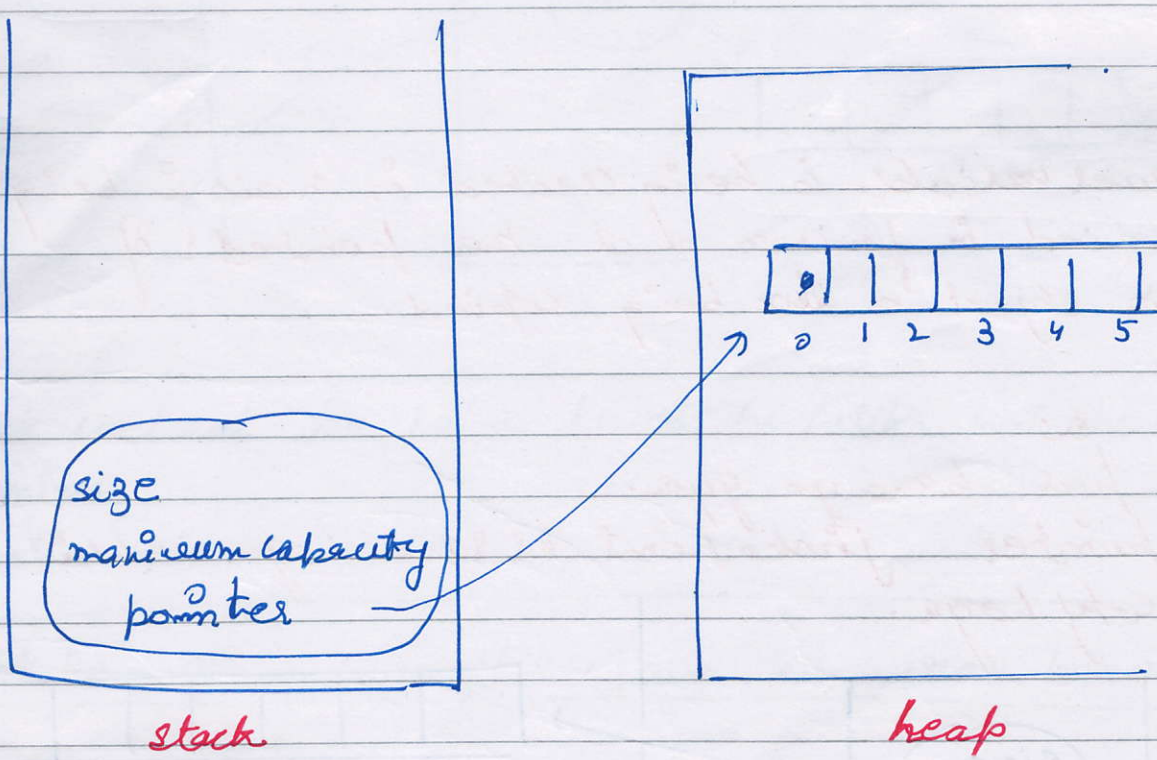
we passed an array of 10 size but the function only copies the base address so when we pass it create it still so it will work.

```
void fun( int arr[], int n)
{
for(int i=0; i<n; i++)
{
}
}
```

→ we should always send size of array because it helps in defining size.

Vector

```
vector<int> arr(n,0);
```

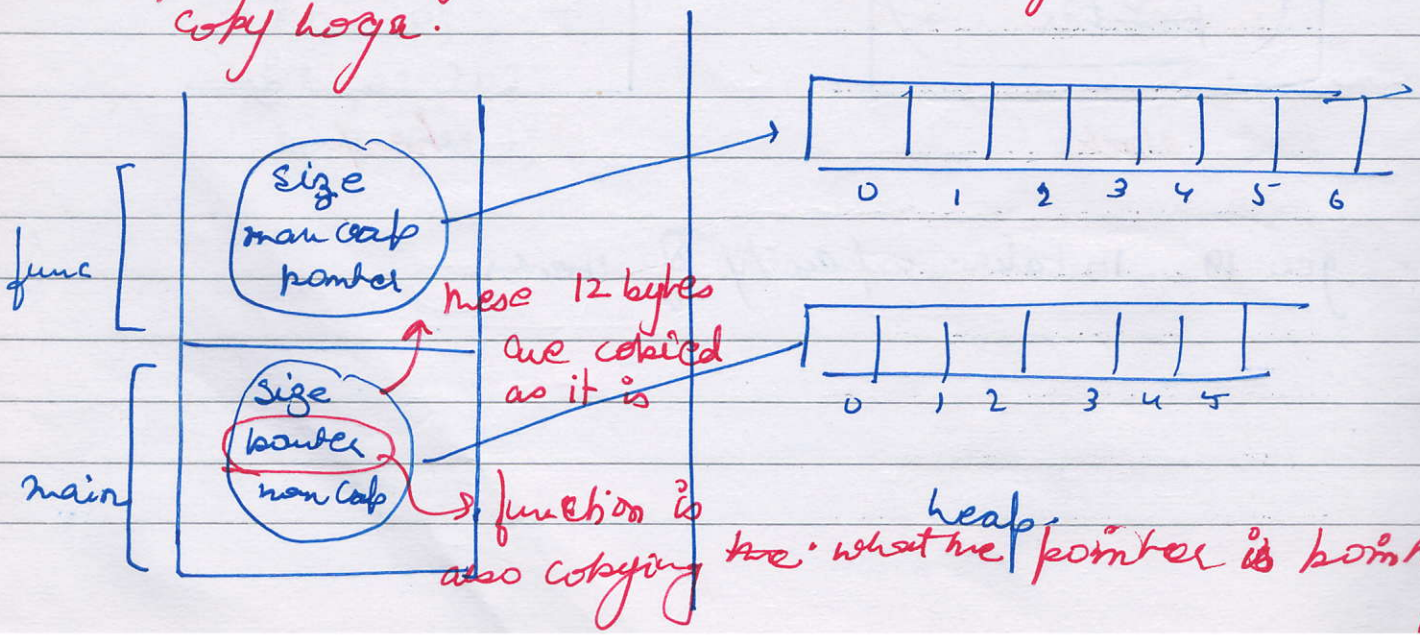


gcc 14 10 lakh. capacity of vector.

```
void fun(vector<int> arr)
{
    for(int i=0; i<arr.size(); i++)
        cout<<arr[i]<<;
}

}

int main()
{
    vector<int> arr(n,0);
    fun(arr);
}.
```

whatever variable is being created in main is being copied in function and the pointer of main object is also being copied.

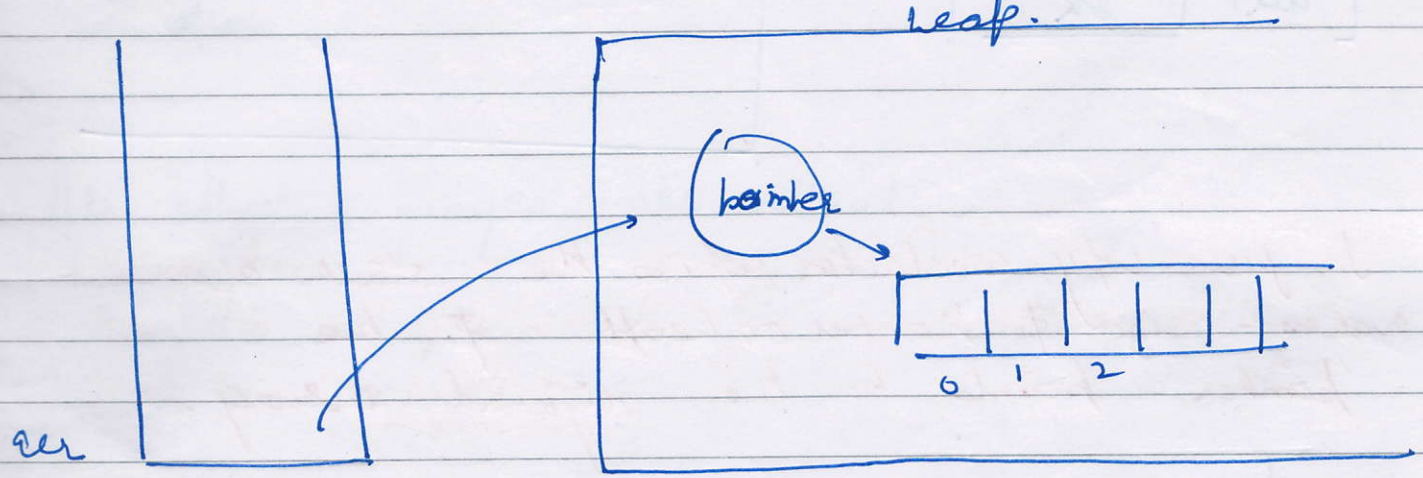
∴ a
stack par banaya gaya.
↳ pointer jiske paas hoga wo bhi copy hoga.



→ behind the scene there is a loop to copy vector which of O(n)
→ It is being envoked twice while going up while going down.

Java

pointer ko heap pe banana do.



stack pe heap pe jane ka ek hi tarika hai hoise pointer

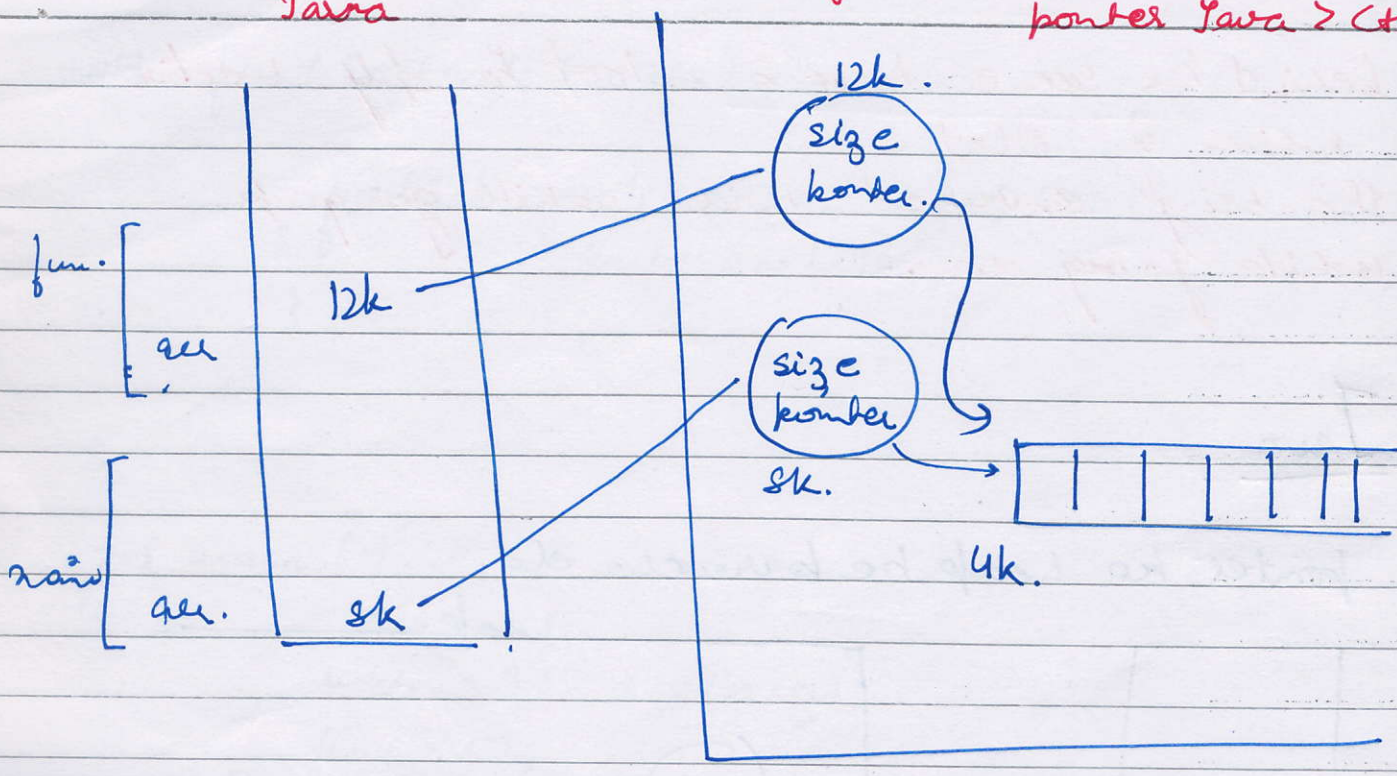
project:

C++ ka array use krke java ka array banana

[new keyword → creation on heap.]

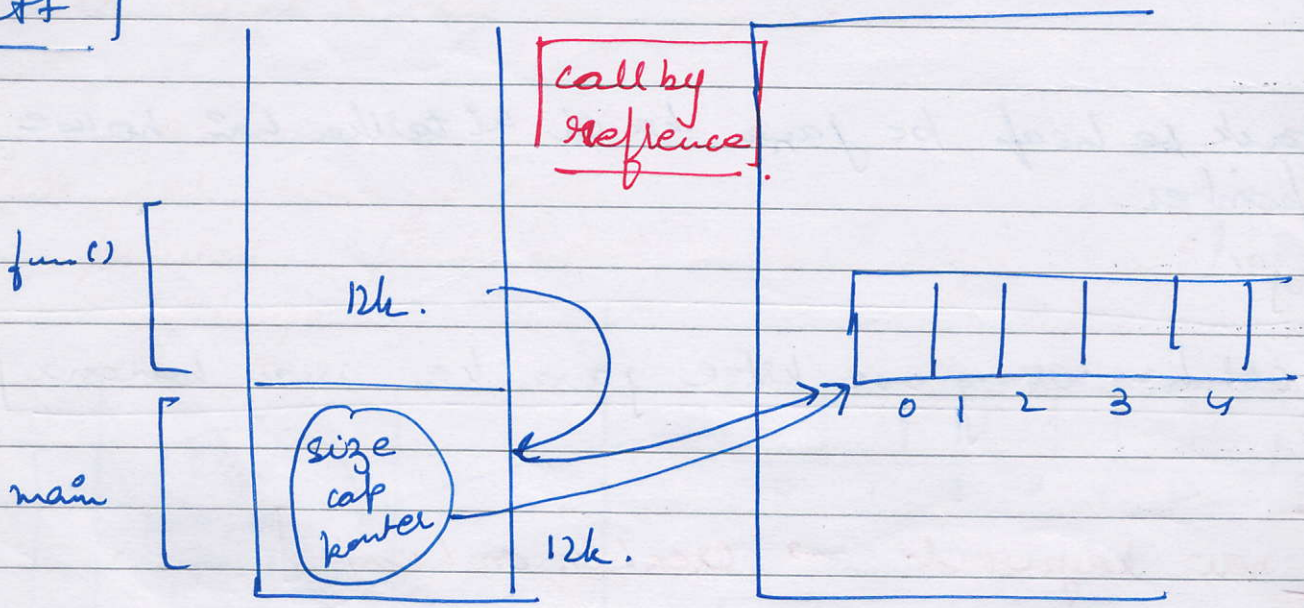
No. of pointers to access array element is greater in java than C++.

pointer Java > C++.

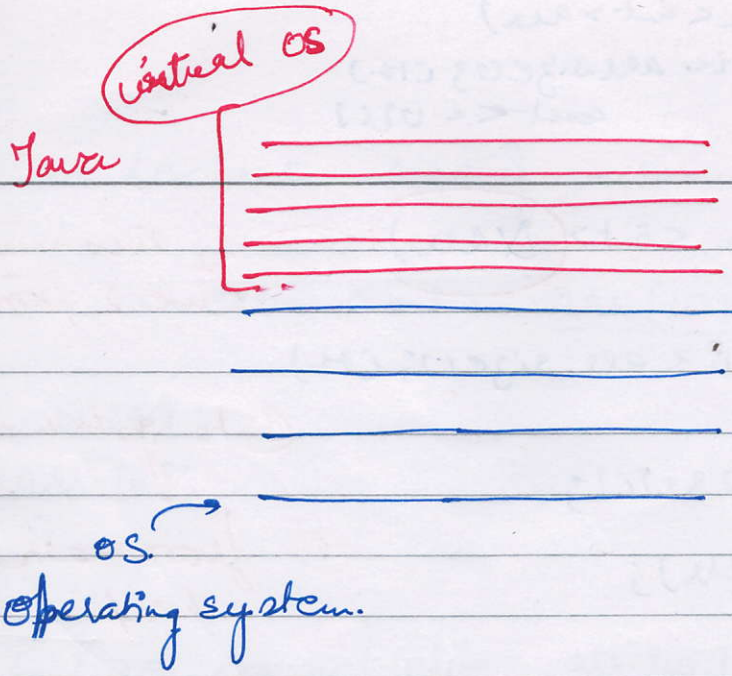


In java copy constructor copies the stack element makes a size & pointer on heap & the pointer points to the original array.

C++



The function copies the address of main and which is further pointing to our heap.



OS → Operating system.

as C is nearest to OS hence it will execute fast than others.

All other languages make virtual OS.
Since there are a No. of layers of abstraction hence it is not preferred for competitive coding.
same is with python.

In C++ vector is having fast implementation because it uses less pointers than that being used in other language.

* Accessing elements on stack is easiest job i.e if we make array on stack.

```
void display (vector<int> arr)
{
    for(int i=0; i<arr.size(); i++)
        cout << v[i]
}
```

```
void input (vector<int> &arr)
{
    for(int i=0; i<arr.size(); i++)
    {
        cin >> arr[i];
    }
    display(arr);
}

int main()
{
    vector<int> arr(n);
```

this will have 2 pointers

① to go to main

② from main to heap.

```
void display (vector<int> & arr)
{
    for (int i : arr)
    {
        cout << i << " ";
    }
}
```

(for each) value loop

index be lekin loop be kaam leta hai

to get values of array

vector is dynamic → grows in size.

push_back()	O(1)

(n)

Once we reach capacity we take O(n) operation to copy n size vector & O(1) to push_back

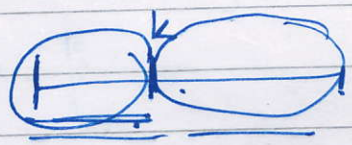
If we declare size of vector.

vector <int> arr(10,0);

arr[9];
arr[6];

but we cannot write arr[10]
because its size is fixed.

So if we use push_back it will grow the size of vector by 2. and push back.



(0, n-k-1).

(n-k, n-1)

(0, n-1)

(i) max
min.

(ii) find (int [] arr, int data)

reverse (int [] arr, int si, int ei).

↳ $\frac{(k-1-0+1)+(l-1-k+1)+(l-1-0+1)}{2}$.

$\frac{2l}{2} \rightarrow l$.

O(n)

reverse (0, n-k-1);
reverse (n-k, n-1);
reverse (0, n-1);

22nd Sept 2024.

Binary Search.

→ whenever we have monotonically increasing or decreasing data.

Lower bound.

```
while (si <= ei)
{
    int mid = (si+ei)/2;
    if (arr[mid] == data)
    {
        if (mid-1 >= 0 && arr[mid-1] == arr[mid])
        {
            ei = mid-1;
        }
        else{
            return mid;
        }
```

at the end of while loop return -1.

Upper bound.

```
if (arr[mid] == data)
{
    if (mid+1 < arr.length && arr[mid+1] == arr[
    {
        si = mid+1;
    }
    else {
        return mid;
```

data : 10^6

$n, \frac{n}{2}, \frac{n}{4}, \frac{n}{8}, \frac{n}{16}, \frac{n}{32}, \ldots \ldots 4, 2, 1.$ ← GP from behind.

$n = 1.2^k$

$2^k = n.$

$\log_2 2^k = \log_2 n.$

$$\boxed{k = \log_2 n}$$

No. of steps to find data

$= \log_2 (10^6)$

$= \frac{\log_{10} 10^6}{\log_{10} 2} = \frac{6}{0.346} \approx 20.$

Nearest element to given No. i/ not element is not present in array.

2 | 2 | 2 | 8 | 8 | 8 | 10 | 11 | 15 | 27 | 52 | 68 | 76 | 98

(1)

```
if (data < arr[0])
    return arr[0];
else if (data > arr[arr.size()-1])
    return arr[arr.size()-1];
```

if ei reaches at -1
answer will be arr[si];

| 2 | 2 | 2 | 8 | 8 | 8 | 8 | 10 | 11 | 15 | 27 | 56 | 68 | 76 | 98 |

ei = -1 (si = arr.length)

```
while (si < ei)
{
    int mid = (si+ei)/2;

    if (arr[mid] == data)
    {
    }
    else if (arr[mid] < data)
    {
        si = mid+1;
    }
    else {
        ei = mid-1;
    }
}
```

if si reaches arr.length arr[ei] will be answer.

(2) when b/w range.

```
if (data - arr[ei] <= arr[si] data)
{
    return arr[ei];
}
else {
    return arr[si];
}
```

15 → 22 → 27
22-15 = 7
27-22 = 5
cout << 27.

either at first while base case. or. while check si & ei at end.

(1)

```
if (ei == -1) return arr[si];
else if (si == arr.size()) return arr[ei];
```

Find missing pair of shoes.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	2	4	4	8	10	10	25	25	48	48	100

1	1	2	2	2	2	2	2	3	3	4	4	4	4	5	5	5	6	6	7	7	7	7	7	7	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

si

mid1 si2

mid.

$\frac{0+26}{2}$ (13)

B odd
check on left
if element is pair it is paired up
move si to right midt.

even position
has check on
right

1	1	2	2	2	2	2	2	3	3	4	4	4	4	5	5	5	5	7	7	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

1	1	1	2	2	3	3	4	4	4	4	5	5	6	6	7	7
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

```
int BS-three (vector<int>& arr).
{

while( si<=ei )
{
    int mid =(si+ei)/2;

    if(mid %2==0)
    {
        ① if (arr[mid]!=arr[mid+1])
        {
            ei=mid-1;
        ② else { si=mid+1;}
    }
    else if ( mid %2 !=0 )
    {
        if (arr[mid]!=arr[mid-1])
        {
            si=mid+1;

        else
            ei=mid-1;

    }

}

if( ei==-1) {    return arr[si];
else if (si==arr.length) { return arr[ei];
else    return arr[si];
```

Square Root.

$\sqrt{27} = 5$.

For square root of 27.

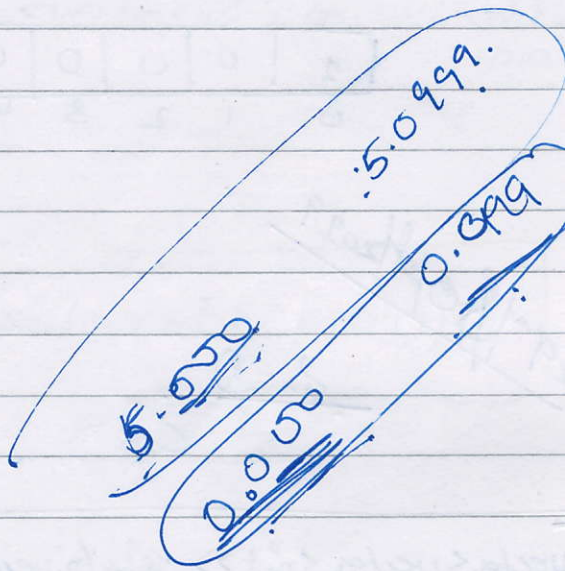
```
si = 0;
ei = length - 1;

mid = (si + ei) / 2;
```

take half

1 2 3 4 5 ... 13

si ↑ ei ↑

```
while(si < ei)
{
    if (mid * mid == data)
        return mid;
    }
    if (mid * mid < data)
    {
        si = mid + 1;
    }
    else
    {
        si = mid - 1;
    }
}
```

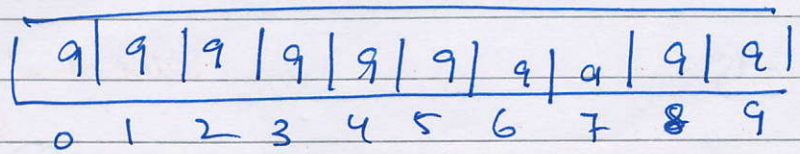


(mid, 0.9999)

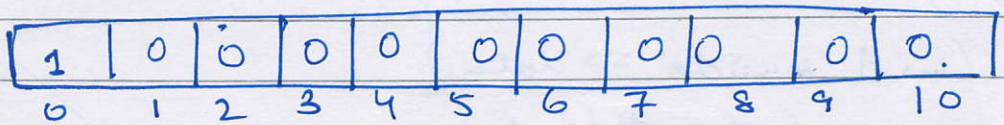
binary search for 4 positions.

mid + (0.000 + 0.999) / 4 → binary search.

Odd Array :



1.



29/Dec/2022

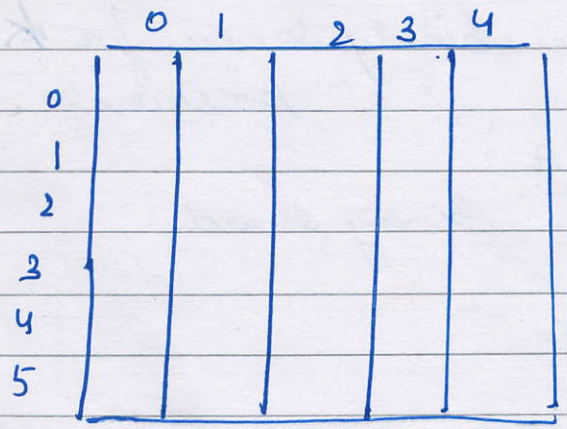
2-D

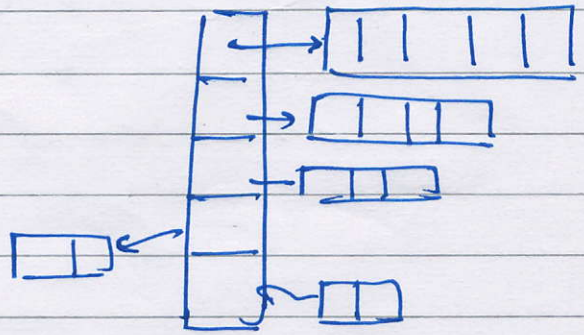
C++

```
vector<vector<int>> arr(n,vector<int>(m,0));
```

Java.

```
int[][] arr = new int[n][m];
```





[Actual representation]

3-D.

int [][][] arr = new int [n][m][p]; // Java.

C++.

vector<vector<vector<int>>> arr(n, vector<vector<int>>(m, vector<int>(p,0)));

vector<vector<vector<int>>> arr.

(n, vector<vector<int>>(m, vector<int>(p,0)));

4-D.

C++.

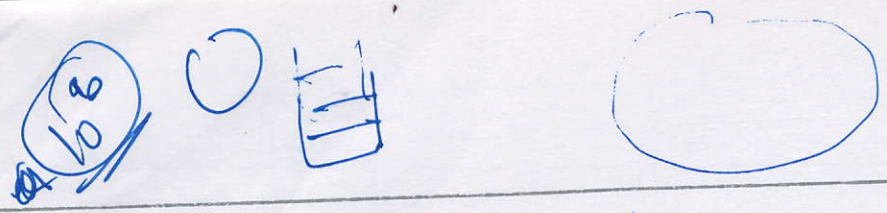
vector<vector<vector<vector<int>>>> arr.

(n, vector<vector<vector<int>>>(m, vector<vector<int>>(p, vector<int>(q,0))));

V<V<V<V<int>>>> arr(n, V<V<V<int>>>(m, vector<vector<int>>(p, V<int>(q,0));

Java

int[][][][] arr = new int[m][n][p][q];



single dimension 10^6 array.

2D. $10^3 \times 10^3$

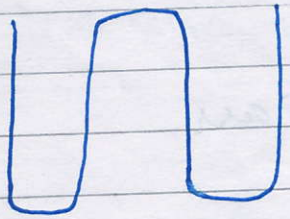
3D $10^2 \times 10^2 \times 10^2$

} max size allow.

(1) Addition of two ~~matrix~~ vector/matrix/str.

(2) ~~Spiral~~ Wave print

1	2	3	4
5	6	7	8
9	10	11	12.
13	14	15	16

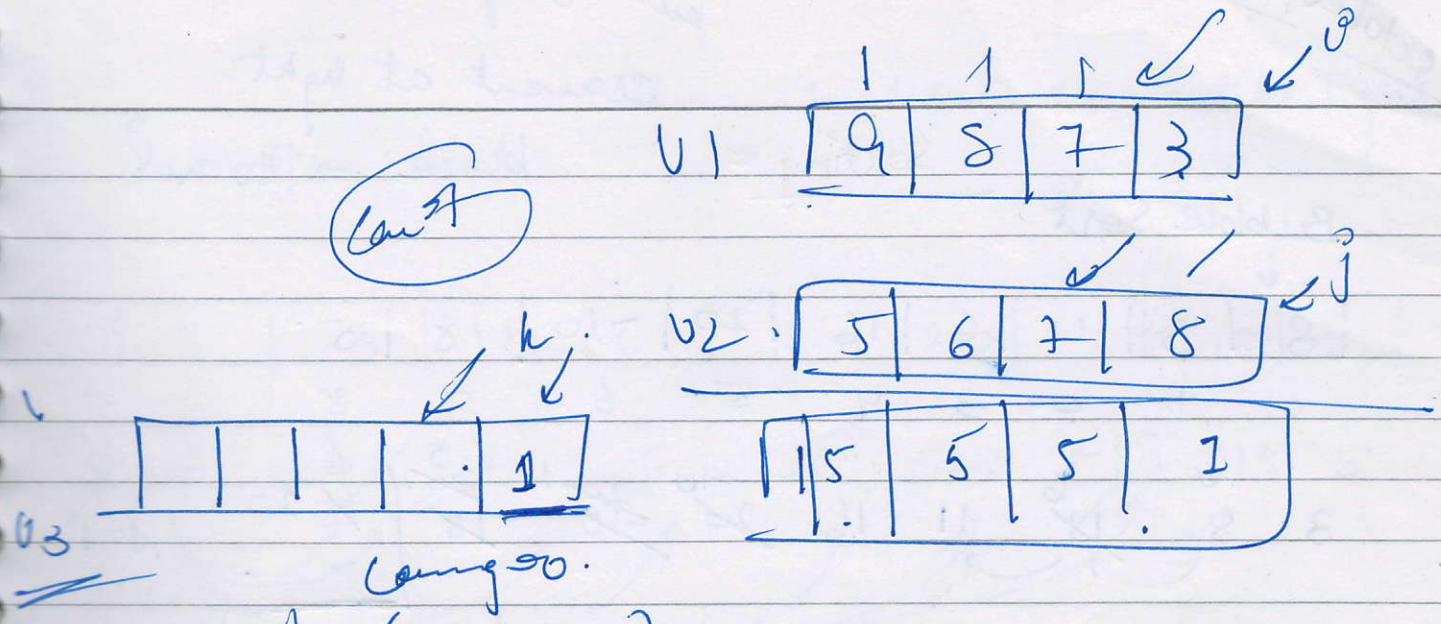


(3) Exit point -

0	0	0	0	1
1	0	0	1	0
0	0	1	0	0
0	0	0	1	0
1	0	0	0	1

4. Spiral print

5. Matrix Multiplication




```
while (k >= 0)
{
    int val = carry;
    if (i >= 0)
    {
        val += v1[i];
    }
    if (j >= 0)
    {
        val += v2[j];
    }
    digit = val % 10;
    carry = val / 10;
    v3[k] = digit;
    i--;
    j--;
    k--;
}
```

3rd October part 1

Sorting

Bubble Sort

put largest element at right place -> to end

| 8 | 3 | 11 | 6 | 16 | 20 | -10 | 18 | 5 |

0 1 2 3 4 5 6 7 8

3 8 6 11 16 -10 18 5 | 20 ... j-1

3 6 8 -10 11 16 5 | 18 | 20.

3 6 8 -10 11 5 16 | 18 20.

3 6 -10 8 5 11 | 16 18 20

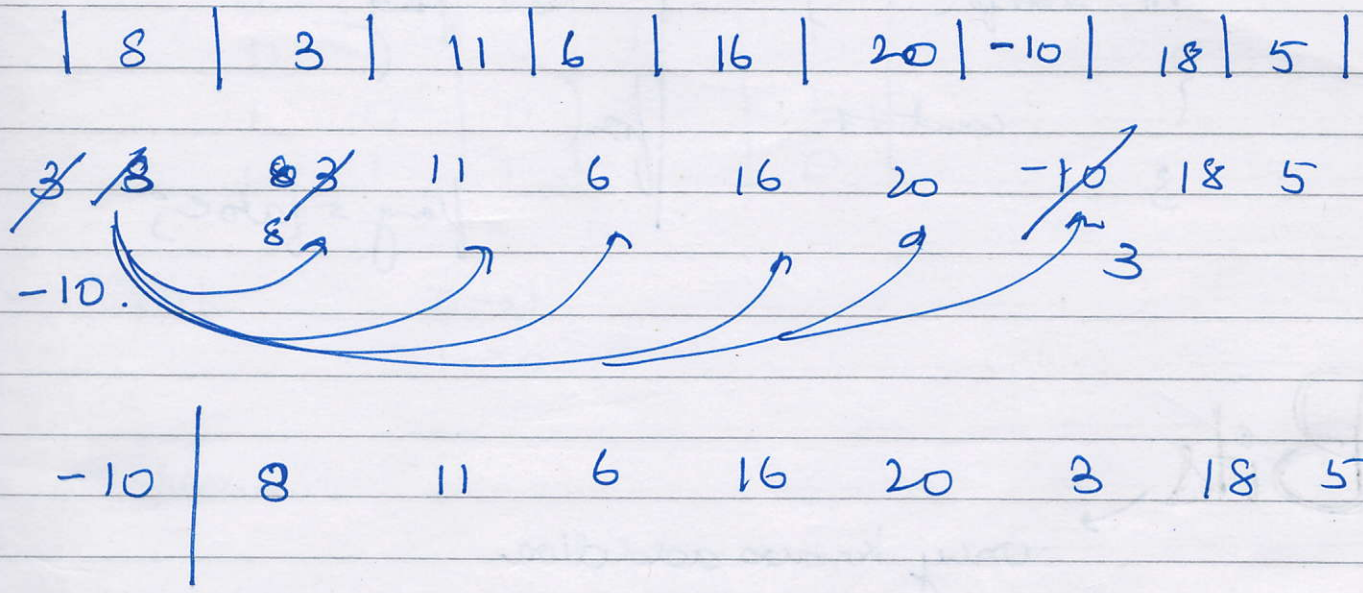
3 -10 6 8 5 | 11 16 18 20

-10 3 5 6 | 8 11 16 18 20.

```
{ for(int i=0; i<n; i++)
  { for(int j=0; j<n-i; j++)
    { if(arr[j-1] > arr[j])
      { swap(arr, j-1, j);
      }
  }
}
```

Selection Sort.

takes smallest element to right position.




```
for(int i=0 ; i<n ; i++)
{
    for(int j=i+1 ; j<n ; j++)
    {
        if(arr[i] > arr[j])
            swap(arr, j, i);
    }
}
```

No. of swaps	No. of valid swaps.
in swap( )	use flag.
{ count++	for(
}	flag = false;

Bits →

only knows addition

size. int = 4 byte.
1 byte = 8 bits

value 1 bit = 1/0.

C++	Java.
int = 4	int = 4
long long = 8.	long = 8.

(10110 11101111.)

To represent 5 ——— 101

only 3 bits required
rest of the bits are zero.

&

XOR.

		&	|	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Not '0 → 1

1 → 0.

XOR

complements.

1 complements

Negative No bit representation

① take positive no. bit representation

② take 1's complement

③ add 1.

1 0 1 1

3 2 1 0 ←

$2^0 \times 1 + 2^1 \times 1 + 2^2 \times 0 + 2^3 \times 1$.

$1 + 2 + 8 = 11$

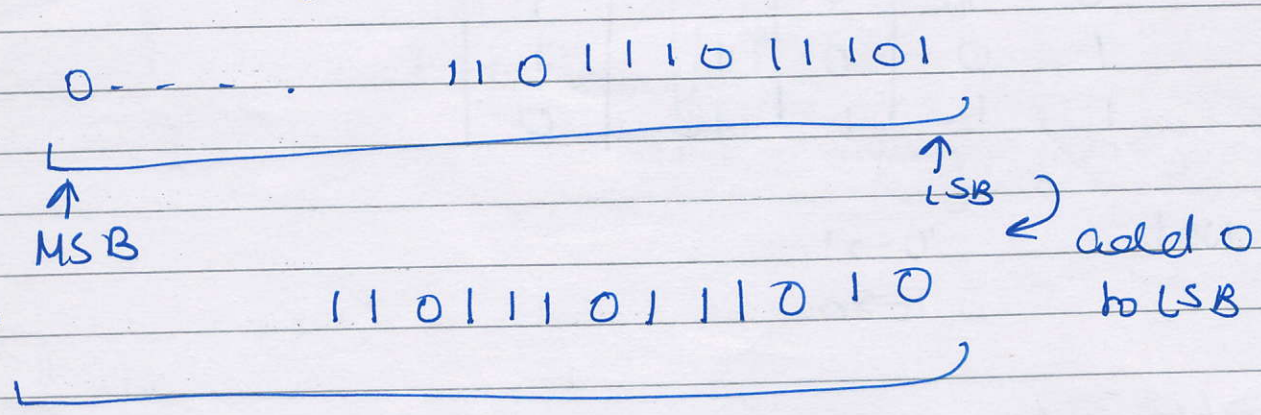
String

10 → int

10.

int n = 273 n <<= 1;

left shift -



leftshift => appending 0 to the end. LSB.

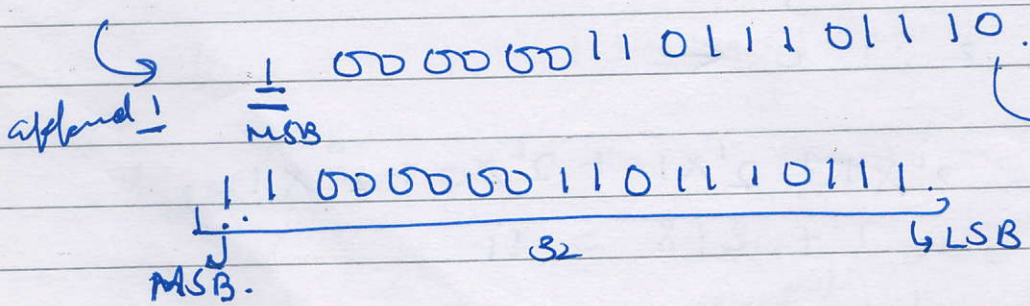
Right shift -

n >>= 1;

if MSB is 1 it will append 1

if MSB is 0 it will append 0.

and LSB is discarded.



↳ 1 0 0 - - .
1 1 0 0 - - .

↳ 0 0 1
↳ 0 0 0 1

① n >>= 2 ;

② n <<= 1 ;

③ n >>= 1 ;

1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1

32.

n >>= 2 append 2 → 11 & discard last 2 elements

↳ 11, 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1.

n <<= 1

1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0.

n >>= 1

↳ 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0

ONLY Java.

[Simple right shift]

↓

only append 0 from starting.

[NO. simple left shift]

left shift → append 0 at LSB.

right shift → append 0/1 at MSB.

Java triple right shift → append only 0 at MSB.

Q1.

1 1 0 0 0 0 1 1 1 0 1 1 1.

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

① [ON → ON
  Off → ON] — 4th bit — (|) OR.

② [Off → Off
  ON → Off] — 4th bit — (&) AND.

① int mask = 1

mask <<= k; // left shift with k append k '0's on end.

num |= mask;

[bit would be ON]

1 1 0 0 0 0 1 1 1 0 1 1 1

mask 0 0 0 0 0 0 1 0 0 0 0 0 0 0 leftshift k.

mask complement:

1 1 1 1 1 1 1 0 1 1 1 1 1 1 1

1 1 0 0 0 0 1 1 1 0 1 1 1

1 1 1 0 1 1 0 1 1 1 1 1 1

1 1 0 0 0 0 0 1 1 0 1 1 1.

bit ON → off.

```
int num = ·;
int mask = 1;
mask = (~ (mask << k));
num = num & mask;
```

No

101 5

210

·k=1

bit ON. 111 7

210

No

101 5

210

if k=2.

off 001 1

No. of set bits

1 1 0 1 1 0 1 1 1 → No. of setbits = 7

7 →	111	→ 3
5 →	101	2
4 →	100	1
10 →	1010	2.

```
int num = n;
for(int i=0; i<32; i++)
{
    int mask = 1<<i;
    if((mask & num) != 0)
    {
        count++;
    }
}
```

as this loop is being executed 32 times

```
1
10
100
1000
10000
```

This will run infinite time.

optimized

```
while(num!=0)
{
    if((num&1)!=0)      // check last bit
    {
        count++;
    }
    num = num>>1;       // right shift discard LSB.
}
```

if we have

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .

above code will not work because always 1 will be append from MSB.

```
int l=0;
while ( num!=0 && l<32 )
{
    if(( num & 1 )!=0 )
    {
        count++;
    }
    max >>= 1;
    l++;
}
```

Java will be preferable for right shift because it will always append '0'.

1. If a No. is power of 2 or Not

No. ① Take 1st complement.
② XOR complement & original No

```
int temp=num;
num ^= (~num);
if ( num==-1)
{
}
```

Inf

return ((n & (n-1)) == 0) ? true : false ;

for 8. 1000.

take 7. 111

(& 0000)

if NO. becomes 0 it is of form 2^n. i.e can be expressed in power of 2.

table in database.

Id	Name	UN	P	Dates.

organisation on file system.

In file system we have extents and

Extent → pages →

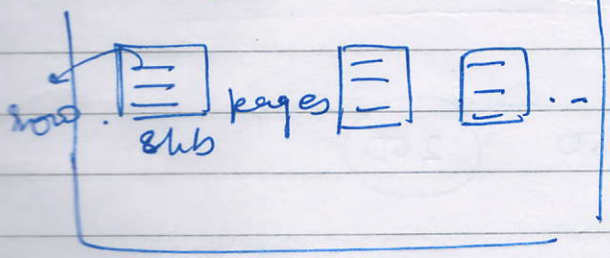
↳ 1 page = 1 file

In page we have rows.

Table → Extent → Pages → Rows. → col data

(64kb) (8kb) (8kb)

Extent (64kb) — Data pages



Row cannot be split into 2 pages it will be in one page

because of no index searching is slow here.

8×10^9 Rows | 1 Row size = 1kb | 8×10^9 kb = 8 Tb data

10^9 pages

Normal search it will load 10^9 page

it will pick one page & read each row.

Harddisk se RAM mein page main other he lenge

This search is called Table stand

① select * from table where id b/w 100 AND 200

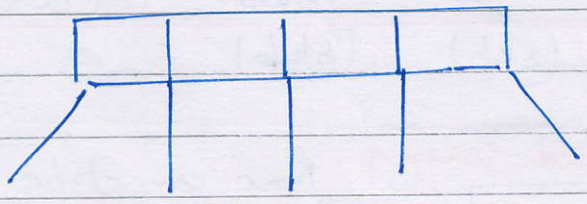
It will have to read all the pages reads 10^9 pages.

B - Tree.

BST degree - 2.
kada bhai?

degree 5

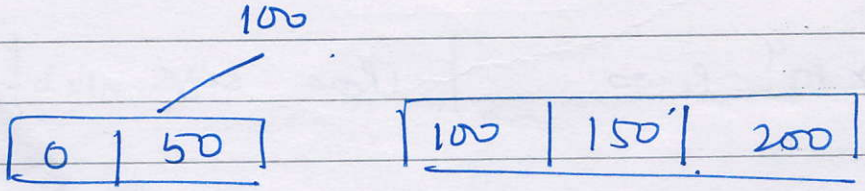
one one level 4 nodes can be there



Inorder is sorted.

0 50 100 150 200

only 4 elements are allowed the moment 200 comes we do page split takes place



add 25 100

0 · 25 50 100 150 200

add 75

100

0 25 ~~50~~ 75 100 150 200

add 90. shift.

50 100

0 _25 50_75_90 100_150_200

add 30, 35, 40.

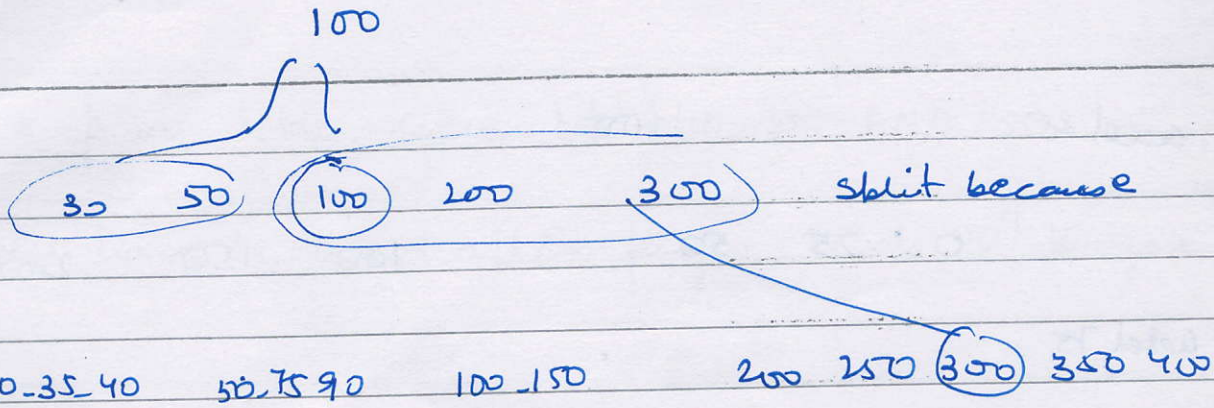
30 50 100

0 25 30 35 40 50 75 90 100 150 200

add 250 300

30 50 100 200

0 _25 30 _35 _40 50 _75 _90 100 150 200 250 300



$\log_d(n)$ searching.

we store index pages

Create Non cluster index on table id.

extra pages will be added.

from all the pages one id will be taken out.

10^9 Data pages.

every row is read and id is taken out.

id will be of 4 byte.

2000 id. $\frac{8 \times 1000}{4} = 2000$.

In one page we can have 2000 id

$\frac{8 \times 1000}{4} \leq 2000$

$$\frac{4\times10^6}{2000} = 2000$$

$$\frac{8\times1000}{4} = 2000.$$

Total rows 8×10^9

In one page we can have only 8 rows. If we preserve only id and not other information

$$\frac{8\times1000}{4} = 2000.$$

[we get 2000 equally spaced rows.]

$$\frac{8\times10^9}{2000} = 4\times10^6$$ one block.

only 3 pages are to be searched.

and 1 data page

To reach $8\times10^6 + 4015$

store starting point of each block.

Index pages.

4000 apart

2000 apart

divided in equal 2000 parts.

8×10^9

8×10^9 | 2000 | 4×10^6 apart

1

$4\times10^6 + 1$

$8\times10^6 + 1$

$12\times10^6 + 1$

We have selected 2000 rows which are 4×10^6 ap

$8\times10^6 - 12\times10^6$ — block size

$$\frac{4\times10^6}{2000} = 2000.$$

2000 rows.

It will store

$8\times10^6 + 1$ id are 2000 apart.

$8\times10^6 + 2000$.

$8\times10^6 + 4000$

They are duplicate ids

① extent id

consecutive

$8\times10^6 + 1$ id → rid { ② page id, ③ row offset }

$8\times10^6 + 2$.

$8\times10^6 + 2000$

dictionary

pages $4\times10^6 + 2\times10^3 + 1$] $\times$ 8 kb

4 gb data is being served

Extra

Extent: 2000
Page 7
Row offset 6.

(1) select * from table
where id = 40000.

$8 \times 10^6 + 4015$.

(2) select * from table where id between

$4 \times 10^6 + 5000$ to $4 \times 10^6 + 5100$.

3 index page to be read.

& 100 data pages { 100 row id. to be loaded.

(3) we can also define cache

compile new.
provide the input file

java client : \client. 20.

Non leaf
stores the next
page index.

root page

meta leaf page stores data page
name.

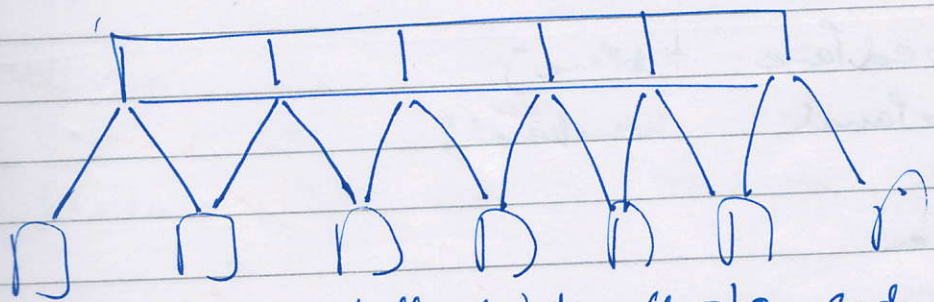
Rbuk = 6 | 3 | 133.

leaf page

sorted.

↓ ↓ ↓.

extent id page id offset.

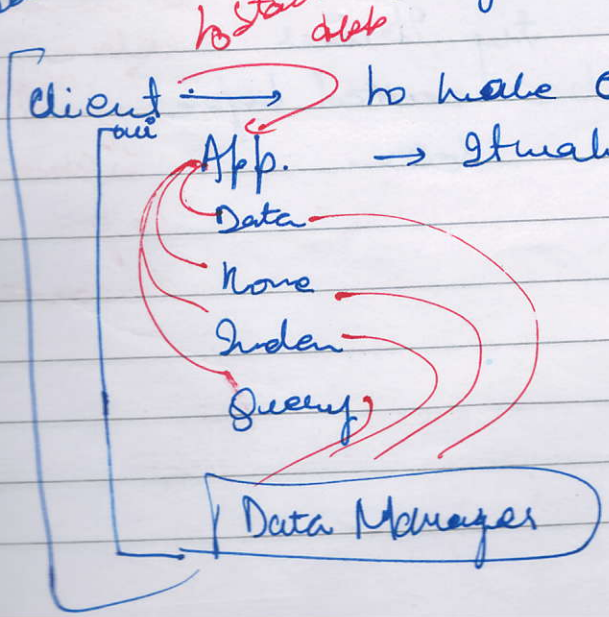


at every tree left right check and traverse beside.

Top pe client.java

5 file App.java
DataManager.java
Datapanel. "
Homepanel. "
Indexpanel. "
Querypanel. "

index



public awt   awt
plus swing.   swing.

java client · D:\Indenty

JFrame object makes window.

public class App extends JFrame. start

JTabbedPane tabPane;
HomePanel homePanel;
Data
Index.
Query

can be override

final vs finally vs finalize.

finally → very strong
even if you do return in try it will be executed before return

finalize → virtual function → destructor of java
↳ if we have to do something before object function is about to get discarded.

try {

}
catch ( ) {

}
finally {

}

Hadoop.

in java

OLAP

Online Analytical Processing.

Modules.

1. HDFS
2. YARN
3. MapReduce
4. Hadoop common.

delete aut
blur scoring.

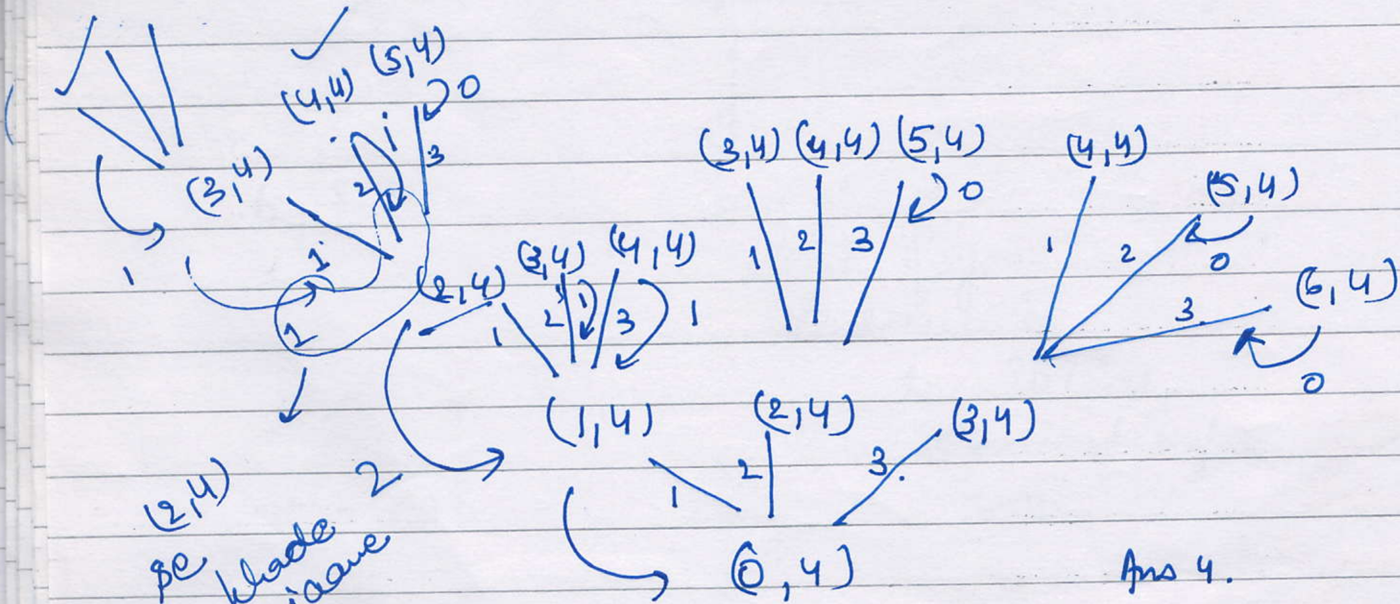
aut
Swi...

java client : D:\Indentfij

public class 1   App entr...

...sp.
go in java
OLAP
Analytical Processing.

3.



(2,4) se 2 pe khade hoke 4 jaane ke 2 raasta hain.

Ans 4.

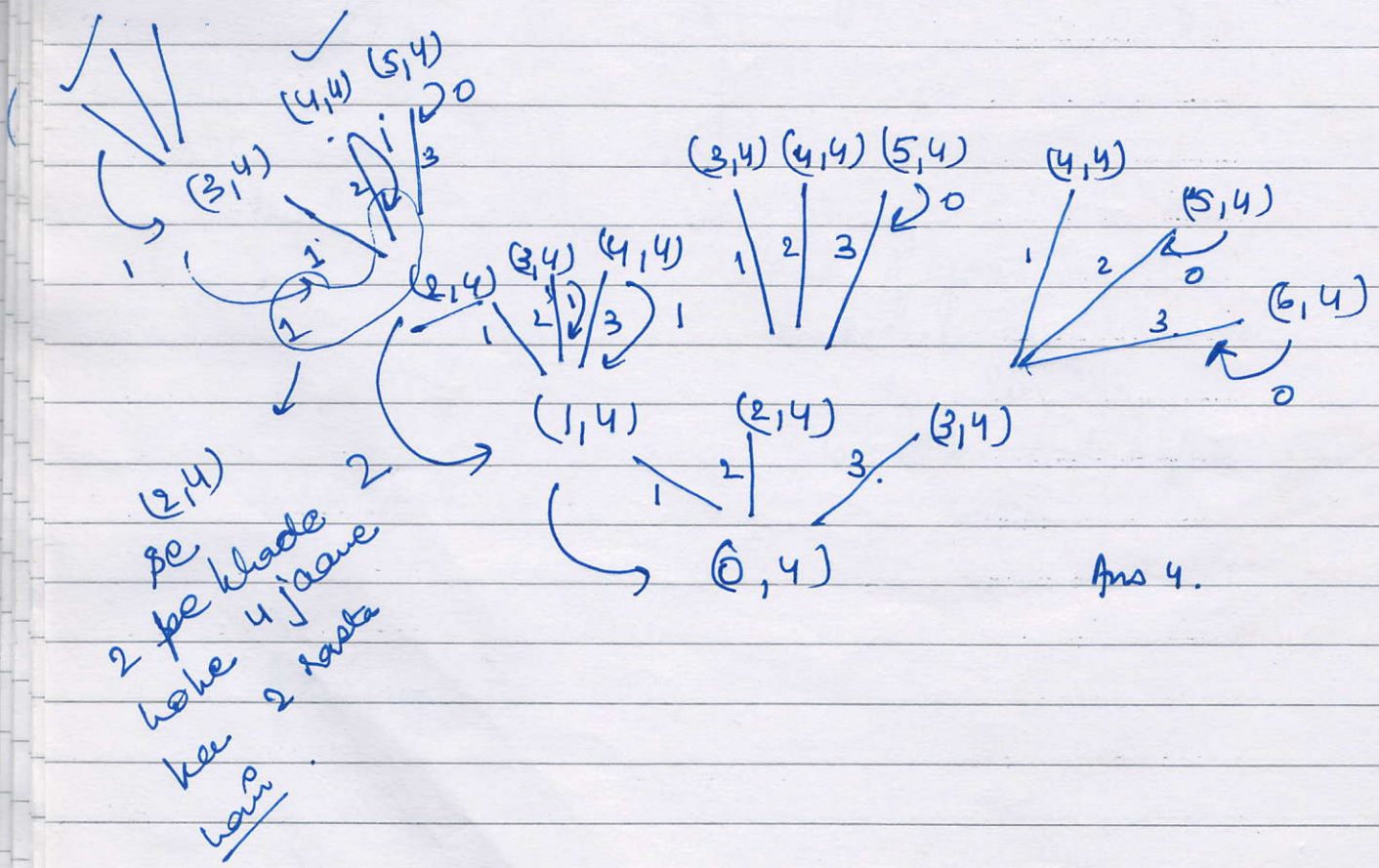
10th/Oct/2019.

stairs

(1,2,3) → t.

```
int stepscount(int steps, int tar)
{
    if(steps > tar) return 0;
    else if(steps == tar) return 1;
    int count = 0;
    count += stepscount(steps+1, tar);
    count += stepscount(steps+2, tar);
    count += stepscount(steps+3, tar);
    return count;
}
```

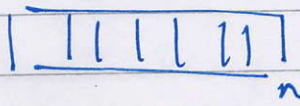
0 → 1 → x
0 → 2 → x.
0 → 3 → x



(2,4) se 2 pe kitne 4 jaane wale 2 raste hai.

import java.util.ArrayList;

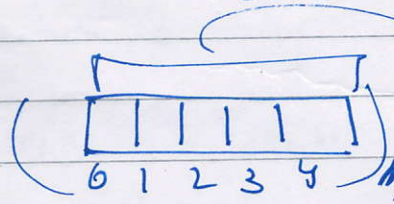
java. Integer	C++.
ArrayList<Integer> arr = new ArrayList<>();	vector
grow with 1.5 x speed.	grow with 2X speed.
get arr.get(idx);	arr(idx)
without size arr.add(val);	arr.push_back(val);
with size. arr.set(idx, val);	arr[idx] = val.
we prefer without size.	
remove. arr.remove(idx); O(n) function.	arr.erase(arr.front()+idx



if we ask to remove first element it is condition dependent because entire array will be copied

Each Remove → O(1)

size .size() .size()



to access in given size use [] → vector .set in AL.

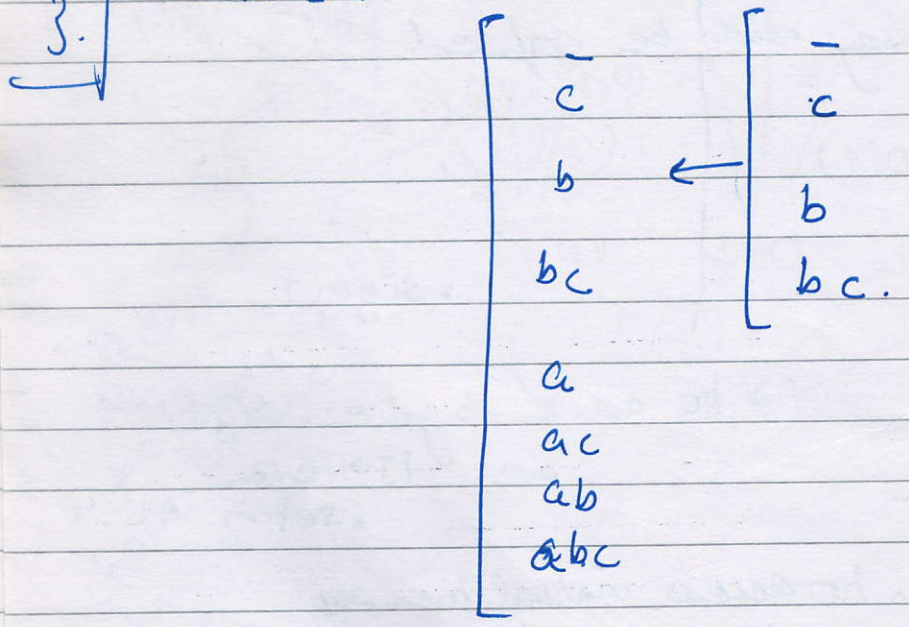
to access virtual memory we have to use push_back & .get.

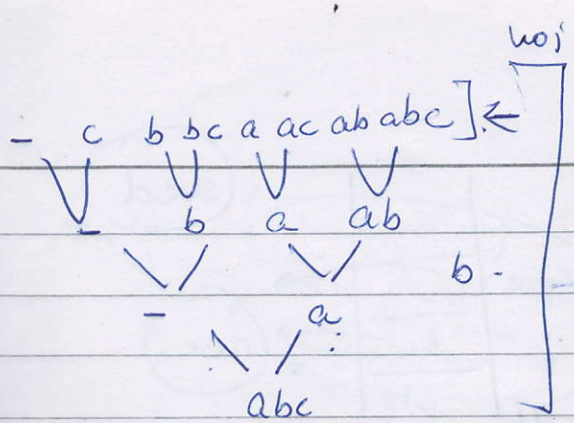
abc → -, a, b, c, ab, bc, abc

return type : Java

```
public static ArrayList<String> subsequence (String str);
{
    if (str.length() == 0)
    {
        ArrayList<String> base = new ArrayList<>();
        base.add("");
        return base;
    }
    char ch = str.charAt(0);
    String ros = str.substring(1);

    ArrayList<String> recAns = subseq(ros);

    ArrayList<String> myAns = new ArrayList<>();
    myAns.addAll(recAns);                          // add all arraylist

    for (String s : recAns)
    {
        myAns.add(ch + s);
    }
    return myAns;
}
```

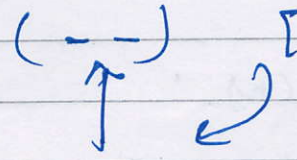
abc



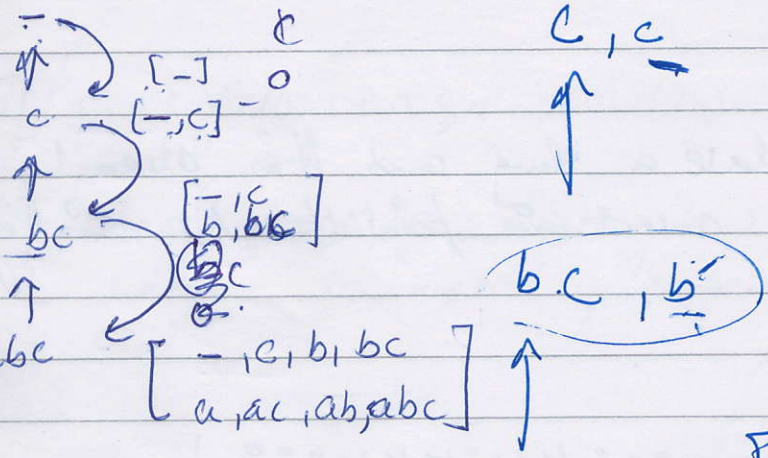


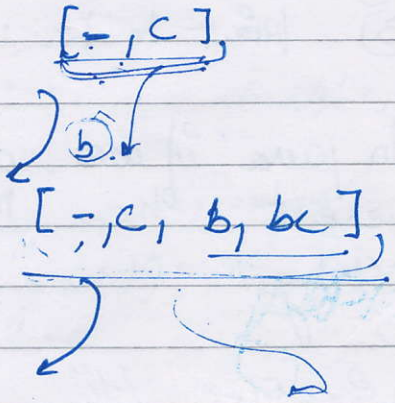
void

return;



empty ArrayList





Java

```
String str = " ";
             new String();
char ch = str.charAt(i);
```

abcdc / gh

0 1 2 3 4 5 6

str.length();

str.substring(2,6);

↳ +1 extra

(2,5)

C++

```
String str = " ";
```

str.substr(2, 4)

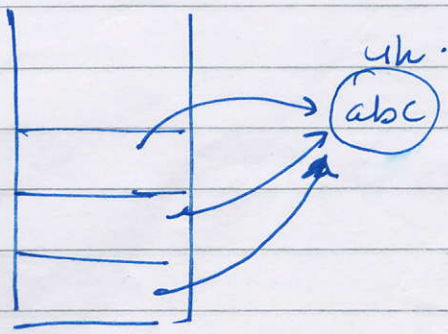
↓ index ↓ length

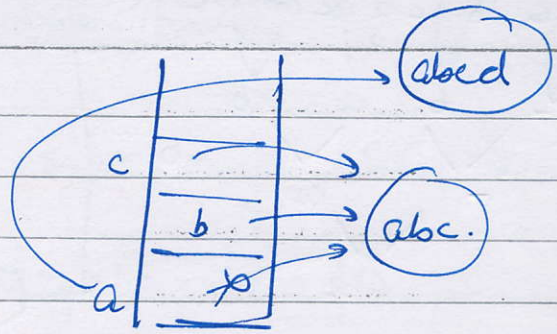
→ addition is costly.

addition is deep

```
String a = "abc";
&  b = a;
   c = "abc";
```

all points to same string.





a += "d";

① copy a. — O(n).
② add d to it.
③ put to it.

In java if we declare a string and it is already present then the current will point to the existing string.

Remove Hi.

(HiiiHiiiHiiHHiiiiiHHiHHHiii)

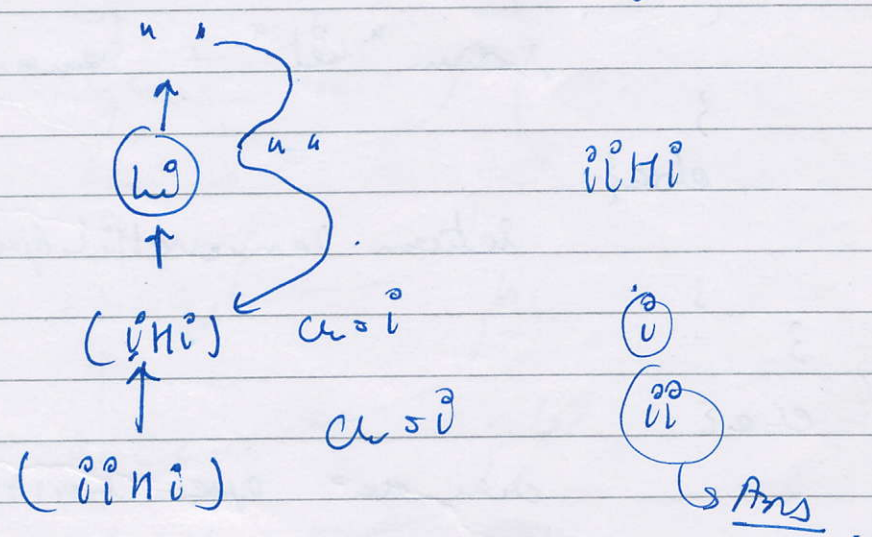
↓

assume yaha se leke yaha tak ki string mein. "Hi" nd chuke hain.

if dar does not match with Hi we will add it and return it.

```
public static String RemoveHi(String ques)
{
    if(ques.length()==0)
    {
        return "";
    }
}
```

```
// if(ques.length()>1 && ques.charAt(0)=="h" && ques.charAt(1)=="i
if(ques.length()>1 && ques.substring(0,2).equals("hi"))
{
    return RemoveHi(ques.substring(2));
}
else{
    char ch = ques.charAt(0);
    return ch + RemoveHi(ques.substring(1));
}
```



Hi ✗

Hit — leave as it is.

HiHit HiHii iHit.

possibilities.

Hit → Hit + ans.

Hi → (i + ans).

iHi

```
public static String removeHi(String ques)
{
    if(ques.length()==0)
    {
        return "";
    }

    if(ques.length()>1 && ques.substring(0,2).equals("hi"))
    {
        if(ques.length()>2 && ques.substring(0,3).equals("hit"))
        {
            return "hit" + removeHi(ques.substring(3));
        }
        else{
            return removeHi(ques.substring(2));
        }
    }
    else{
        char ch = ques.charAt(0);
        return ch + removeHi(ques.substring(1));
    }
}
```

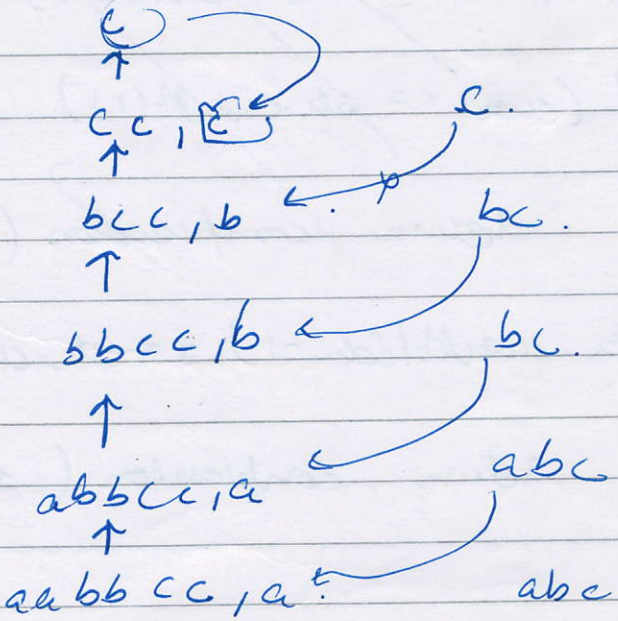
Remove duplicated:

aaaabbcdcfggg → abcdcfg

```
public static String removedup(String ques)
{
    if(ques.length()==1)
        return ques;
}
```

|| if string is of 1 length we need not check for duplicates.

```
char ch = ques.charAt(0);
String recAns = removeDuplicates(ques.substring(1));

if(ch == recAns.charAt(0))
{
    return recAns;
}
else{
    return ch + recAns;
}
}
```



13th Oct 2019

aaaa bbbcdef.

a4b3cdef.

aabaabbccdeff.

↑

start from here.

count

int idx.

public static string compress (String str, int idx, int count).

{

if (str.length() == idx)

{ return String ans = str.charAt(idx-1) + (count > 1 ? count+"" : "");

}

char ch = str.charAt(0);

if (ch == str.charAt(1))

{ return compression (

if (str.charAt(idx-1) == str.charAt(idx))

{ return compression (str, idx+1, count+1);

}

else{

String ans = str.charAt(idx-1) + (count > 1 ? count

Checking if count is 1 or not-

return ans + compression (str, idx+1, 1);

}

aaaaaaabbccdeff.

0 1 2 3 4 5 6 7 8 9 10 11 12 13

(6,6) → (7,1) → (8,2) → (9,1) → (10,2) → (11,1) →.

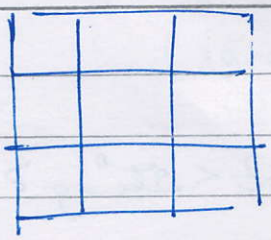
a6 b2 c2

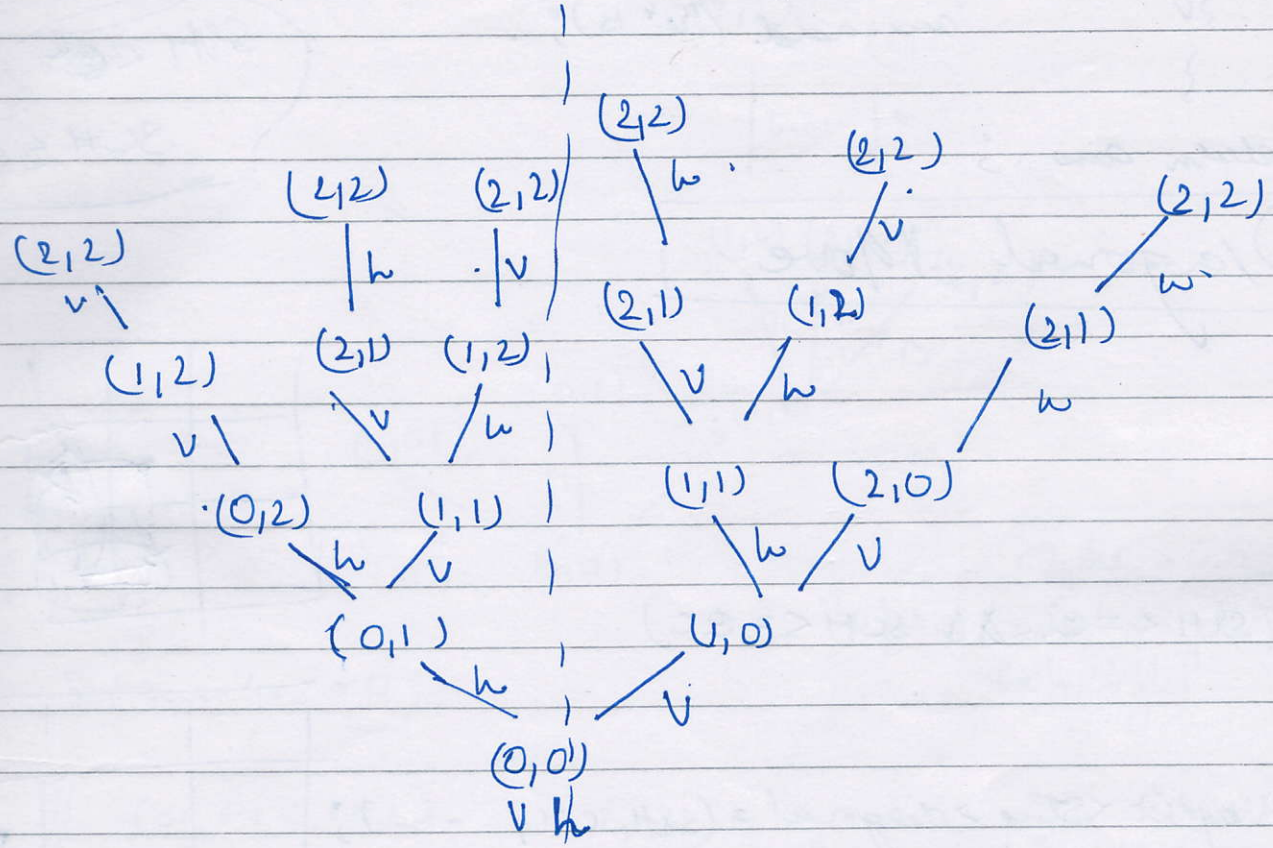
(6,6)
↑
(5,5)
↑
(4,4)
↑
(3,3)
↑
(2,2)
↑
(1,1)

idx ↳ count.

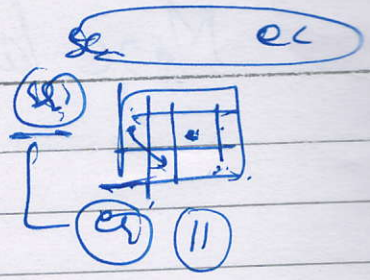
(abcdef)((Caabc(())) inp (pqr)

Maze Path



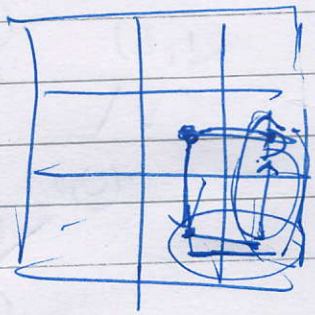



```
public static ArrayList<String> mazePath(int sr, int sc, int er,
                                         int ec).
{   if(sr==er && sc==ec)
        ArrayList<String> base=new ArrayList<>();
        base.add("");
        return base;
    }
ArrayList<String> ans=new ArrayList<>();
if(sc+1<=ec)
{   ArrayList<String> Horizontal=mazePath(sr,sc+1,er,ec);
    for(String s:Horizontal)
    {   ans.add("H"+s);
```

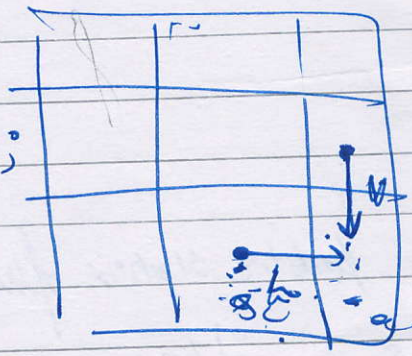


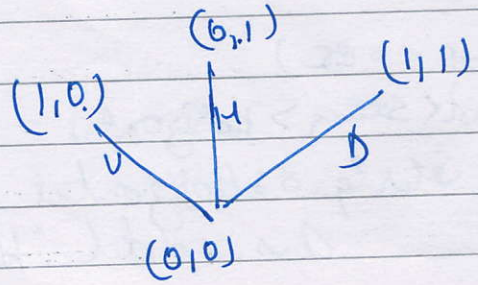
```
if (srH <= er)
{
    ArrayList<String> vertical = ...
    for (String s: vertical)
        ans.add ("V" + s);
}
return ans;
```

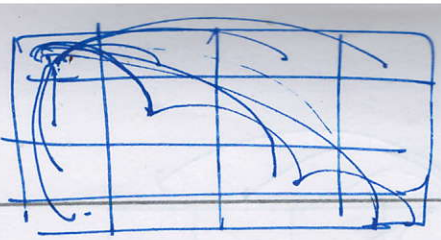
Diagonal Move



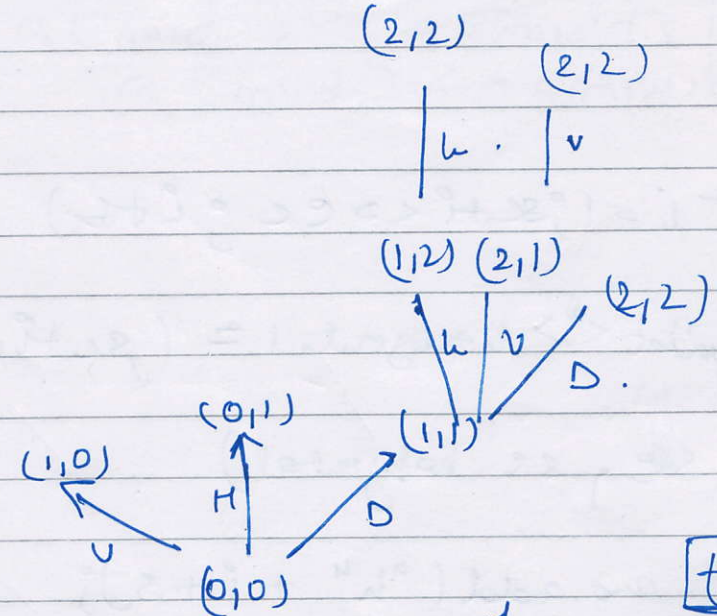
```
if (srH <= er && scH <= ec)
{
    ArrayList<String> diagonal = (srH, scH, ---);
    for (String s: diagonal)
    {
        ans.add ("D" + s);
    }
}
```







Max weight of tree

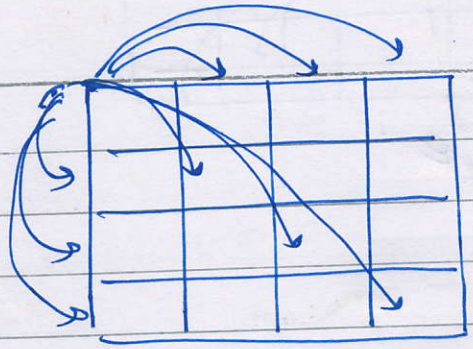


take min = 1.

To find min weight use min weight.

```
int maxhei = 0;

if (sc+1 <= ec)
{
    maxhei = Math.max(maxhei, mazepath(sr, sc+1, er, ec));
}
if (sr+1 <= er)
{
    maxhei = Math.max(maxhei, mazepath(sr+1, sc, er, ec));
}
if (sr+1 <= er && sc+1 <= ec)
{
    maxhei = Math.max(maxhei, mazepath(sr+1, sc+1, er, ec));
}

return maxheight;
```

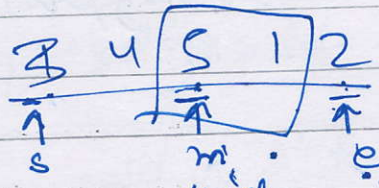
Multiple jump.



```
{ for(int i=1; i<=cc; i++)
    ArrayList<S> horizontal = (s.c+i, );
    { for( String s: horizontal)
        ans.add("h" + i + s);
    }
}
```

Binary Root

```
if(s>e)
    return -1;
```



arr[mid] > arr[end]
so mid+1

arr[mid] < arr[s]
e = mid-1;

1 2 3 4 5 1 0

if (a[mid] >= a[mid+1])

if(a[mid] < a[mid+1] ||
a[mid] < a[mid+1])

1 2 3 4 5

mid = 0

a[mid] >= [mid+1]

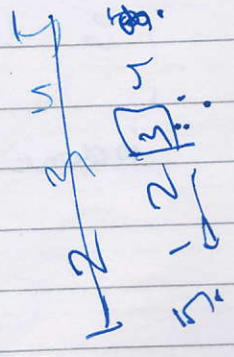
else if mid = ... len - 1

mid-1 > mid

min [mid];

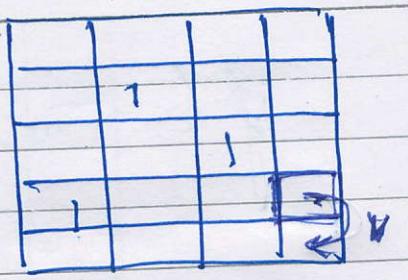
else if

else return -1



17th Oct 2019.

Floodfill (karo)



public static ArrayList<String> floodfill (int sr, int sc, int er, int ec,
boolean[][] isdone)

up
down
left
right

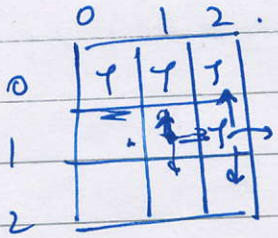
if (sr == er && sc == ec)
{

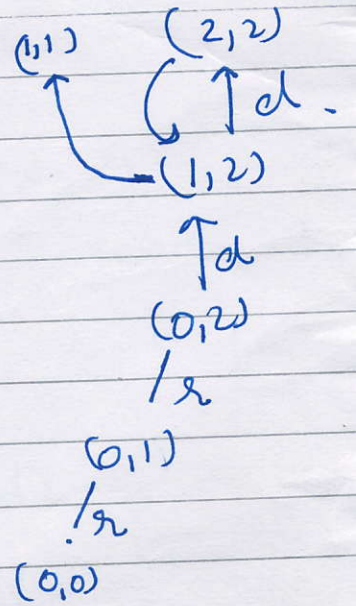
}

ArrayList<String> recAns = ArrayList<>();
isdone[sr][sc] = false;

if (sr-1 >= 0 && !isdone[sr-1][sc])
{

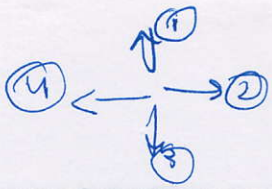
}



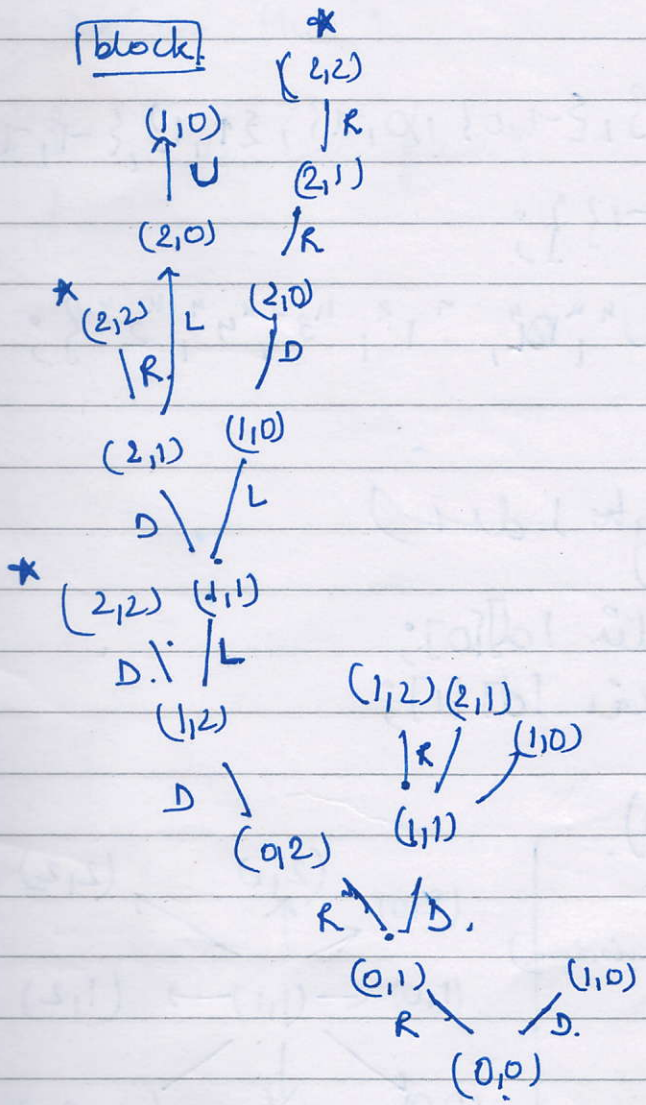


while coming back from (2,2) we see at (1,2) 3 calls have been done only one call is left which is left call.

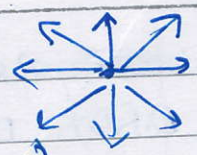
isdone[sr][sc] = true;



RRDD, RRDLDR, RRDLDRR, RDRD, RDDR, RDLDRR, DRURDD,
DRRD, DRDR, DDRUURDD, DDRURD, ——.



Stuff



int [][] dir = {{1,0}, {0,1}, {-1,0}, {0,-1},
{{-1,-1}, {1,1}, {-1,1}, {1,-1}}

{ if (sr == er && sc == ec)

Add <s> nose.

3

int [7][7] dir = {{1,0}, {0,1}, {-1,0}, {0,-1}, {1,1}, {-1,-1},
{-1,1}, {1,-1}};

String [] jump = { "D", "R", "U", "DL", "", "", "3", "4", "2" };

isdone [sr][sc] = true;

for (int d = 0; d < dir.length; d++)

int x = sr + dir[d][0];
int y = sc + dir[d][1];

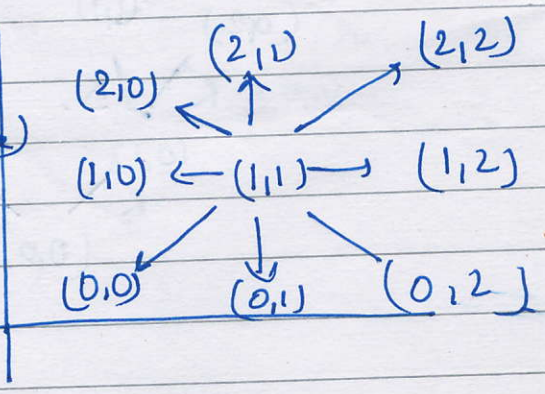
if (isvalid (x, y, path, isdone)).

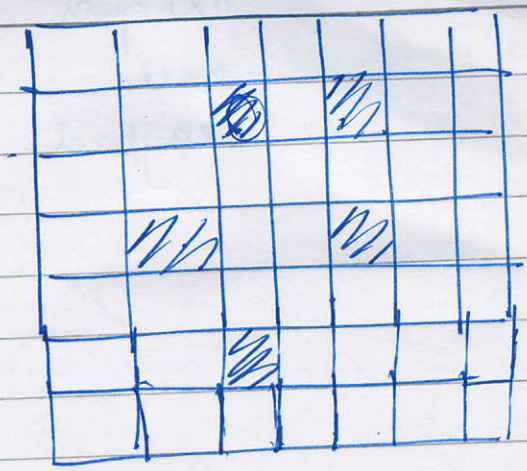
{ AL <S> calls = floodfill (x, y, er, ec, path, isdone)

for (String s: calls)
{ myans.add (s + jump[d]);
}

}

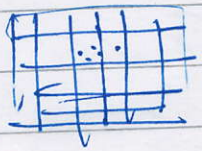
isdone [sr][sc] = false;





static boolean isValid.
(x, y, [][] path , , [][] isdone)

if(x >= 0 && y >= 0. &&
x < path.length
&& y < path[0]. length.
&& !isdone[x][y] && path[x][y])



{
return true;
}
else { return false;
}.

19th/Oct/2019.

n Knights
path
keypad.

20th Oct 2019.

Hacker earth

ke contest aa rahe hai karna hai

NCR

Permutation

generic → exactly meaning?

c c.

_c. [c] - c

(bc) [bc, cb.]

abcd. abc. abc | acb
bac | cab
bca | cba. } bc

public static ArrayList<String> permut(String str).
{.
if (str.length()==0) return new ArrayList<>();
if (str.length()==1)
{ AL<S> base = .
base.add(str);
return;
}

char ch = str.charAt(0);
String roq = str.substring(1);

AL<S> myAns = new AL<>();
AL<S> rec = permut(roq);

for (String s : recAns)
{ for(int i=0; i<=s.length(); i++).
{ String ans = s.substring(0, i) + ch + s.substring(i);
} myAns.add(ans);

[':-,

0 = ".'

1 = a.

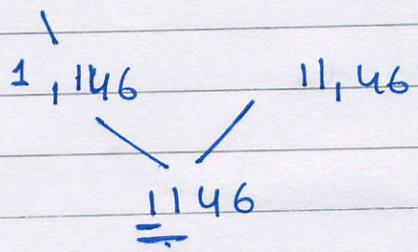
2 = b.

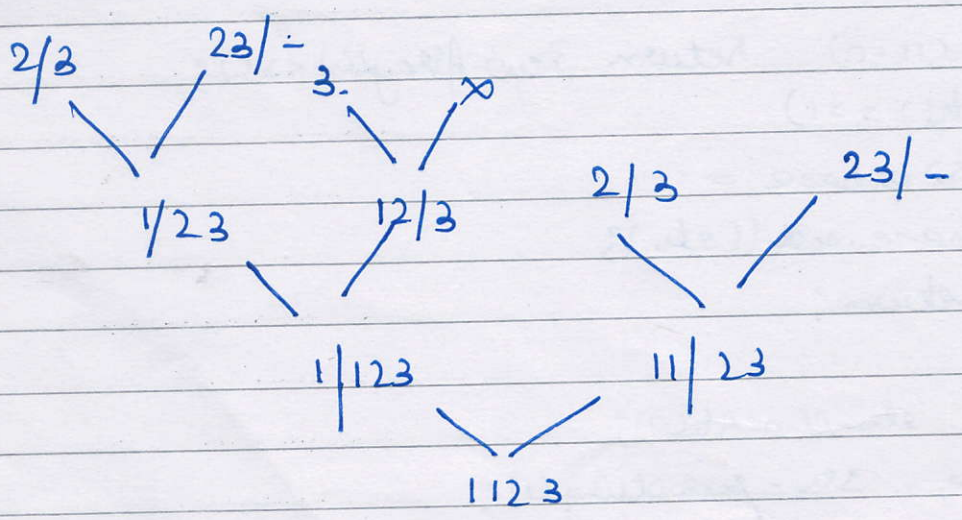
⋮

26 = z.

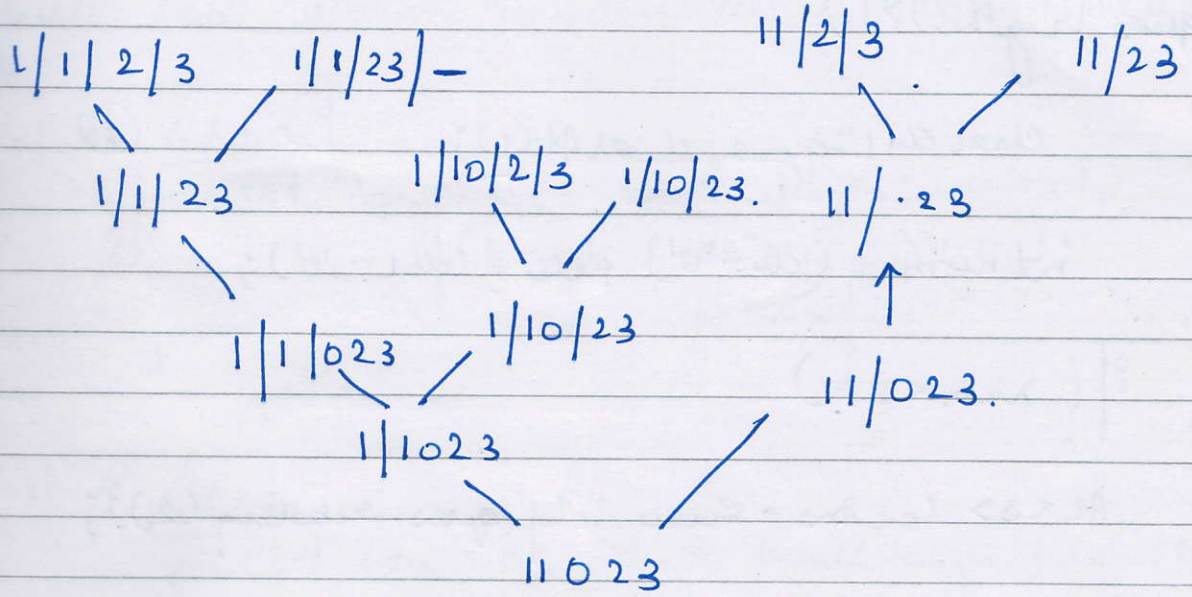
1146.

10001





empty string is of size 0 but size of all arrlist will be 1.




```
public static ArrayList<String> encoding(String ques)
{
    if(ques.length()==0)
    {
        ArrayList<String> base = new ArrayList<>();
        base.add("");
        return base;
    }

    char ch = ques.charAt(0);
    ArrayList<String> myAns = new ArrayList<>();
    if(ch=='0')
    {
        return encoding(ques.substring(1));
    }
    else{
        ArrayList<String> recAns = encoding(ques.substring(1));
        for(String s: recAns){
            char ch1 = (char)('a'+ch-'1');
            myAns.add(ch1+s);
        }
    }
}
```

```
if(ques.length()>1 && ques.charAt(1)<'7')
```

because second character should

int ~~String~~ kooku = Integer.parseInt (ques.substring(0,2));

if (ques.length()>1)

char ch1=ques.charAt(1);

int num = (ch-'0') + ch + (ch1-'0');

if (num<27).

{

AL<S> recAns = Encoding(ques.substring(2));

{for String s: recAns)

char ch1 = (char) ('a' + num-1);

myans.add (ch1+s);

}

}

return myans;

Sudoku.

Att atoi stoi

centeroid decomposition.

dsposition decomposition.

	0	1	2
0			
1			
2			

$\frac{5}{3}, \frac{5}{3} \rightarrow (1,1)$

$\frac{5}{3} = 1 \times 3 = 3$ gives correct position.

$\left(\frac{5}{3}\right) \times 3$.

board, int i, int j, int

int
String nooku - Integer.

if (ques.length() > 1)

char ch1 = ques.charAt(

int num = (ch - '0') + c

if (num < 27).

{

AL<S> recAns = succ

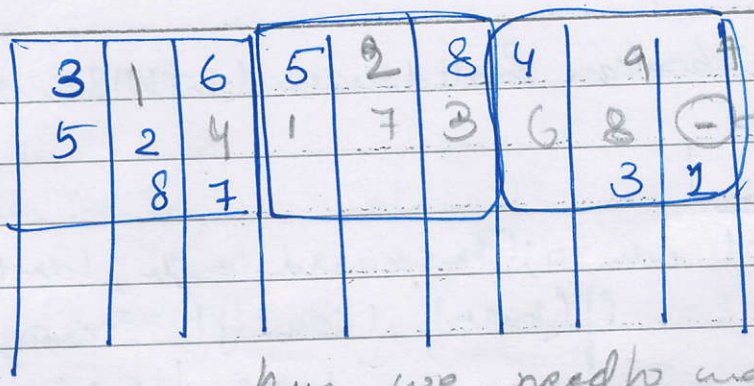
{ for (String s : recAns

char ch1 = (c

myans.add (

}

}



thus we need to make sure backtracking takes place and it will fall to the previous position.

```
public static sudoku(int[][] board)
{
    for(int i=0; i<board.length; i++)
    {
        for(int j=0; j<board.[0].length; j++)
        {
            for(int num=1; num<=9; num++)
            {
                if(isValidSudoku(board, i, j, num){
```

main.

virtual index

boolean.

```
public static void int sudoku (int [][] board, int vidx ).
{.
  if (vidx == board.length * board[0].length)
  {
      ~~return 1;~~
      ① for(int[] ar : board)
        {    for(int ele : ar)
             {
                 sypo( ele+" ");
             }
             sypo()}
      ② return 1;   }          return true
  }

  int r = vidx/9;
  int c = vidx%9;
  int count =0;                 boolean res = false;
  if (board [r][c] != 0)).      res = res || sudoku (board, vidx+1)
  {
          return sudoku( board, vidx+1);
  }
  else{
      for(int num =1; num<= 9 ; num++)
      {
          if (isval (board, r, c, num).
          {
              board [r][c]= ~~too~~ num;
res= res || sudoku(board, vidx+1) count += sudoku ( board ,vidx+1);
              board [r][c] = 0;
          }
      }
  }

  return count;     |    return res
}
```

This will print all possibility

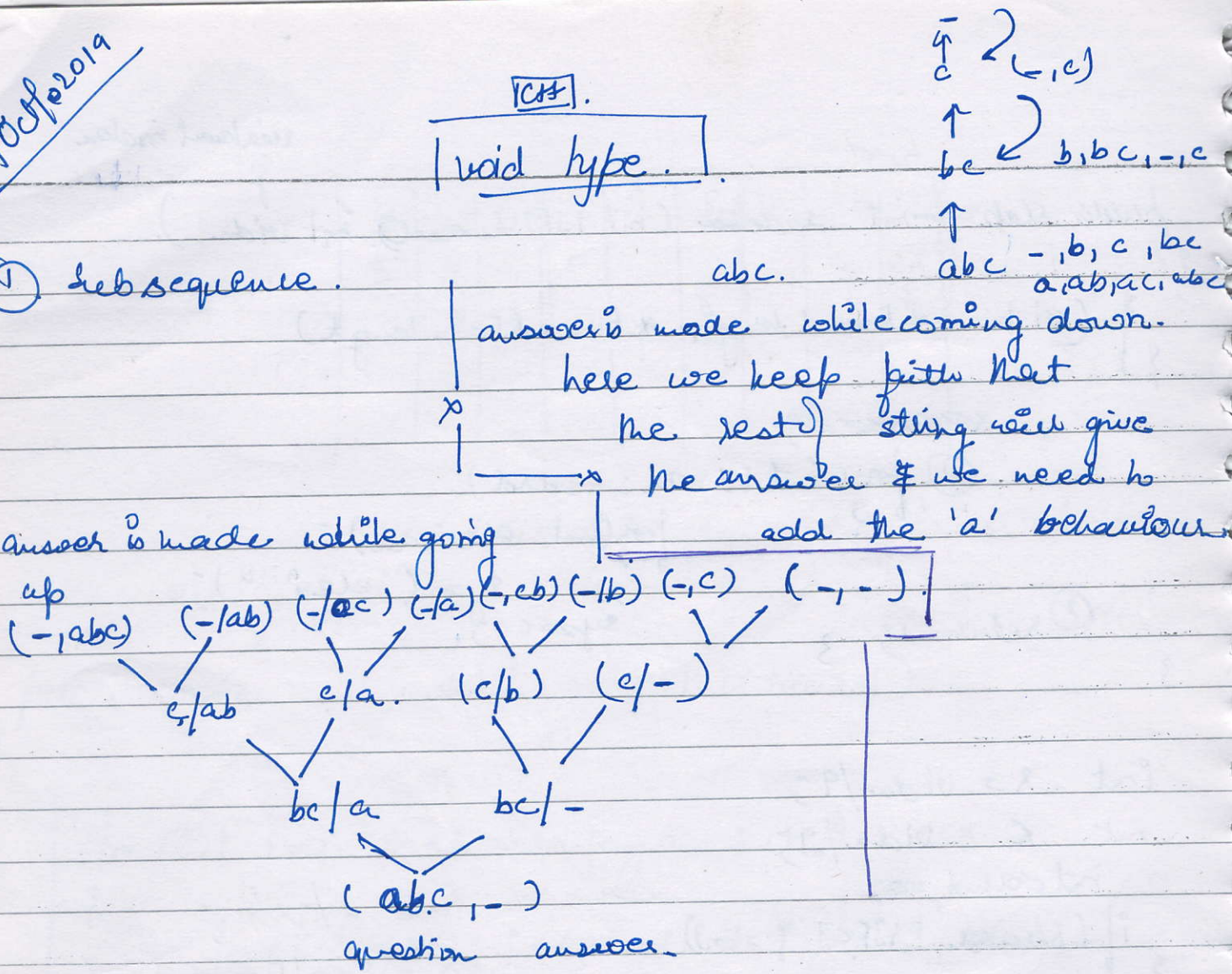
This will print only 1

3rd Oct 2019

C++.

void type.

① Subsequence.



Sabse pehle element ko bola ki I am following these step rest of the string is supposed to follow the same.

```
#include <climits>.

void subseq (string ques, string ans).
{
    if (ques.length() == 50)
    {
        cout << ans << endl;
        return;
    }

    char ch = ques[0];
    string roq = ques.substr(1);
```

Java string input

String str = scn.nextLine();

OR

string str = scn.next();

(2) Remove hi

```
void removehi(string ques, string ans)
{
    if (ques.length() == 0)
    {
        cout << ans
        return;
    }

    if (ques.length() >= 2 && ques[0] == 'h' && ques[1] == 'i')
    {
        removehi(ques.substr(2), ans);
    }
    else { removehi(ques.substr(1); ans + ques[0]);
    }
}
```

(3) Remove duplicates.

```
void removedupli(string ques, string ans).
{
    if (ques.length() == 0)
    {
        cout << ans;
        return;
    }

    if (ques[0] == ans[ans.length() - 1])
    {removedupli(ques.substr(1), ans);
    }

    else
    {
        removedupli(ques.substr(1), ans+ques);
    }
}
```

(2) method

(aabbcc, -)

always include first character.

and.

match if the ques[0] == last char of ans

donot add it

else add it.

(4) Compression

aaaaa bbbb dl.

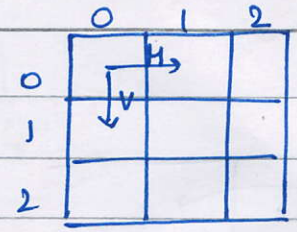
a5b4 dl.

```
void compression (string ques, string ans int count)
{
    if (ques.length() == 0)
    {
        cout << ans;
        return;
    }
    char ch = ques[0];
    string roq = ques.substr(1);
    if (roq.length() != 0)
    {
        if (ch == roq[0])
        {
            compression(roq, ans, count+1);
        }
        else
        {
            compression(roq, ans + ch + to_string(count), 1);
        }
    }
    else
    {
        compression(roq, ans + ch + to_string(count), 1);
    }
}
```

```
if (ques.length() > 1 && ques[0] == ques[1])
{
    compression(ques, ans, count+1);
}
else
{
    ans += ques[0] + to_string(count);
    count = 1;
    compression(ques.substr(1), ans, count);
}
```

this is when rest of string is blank (roq = "")

6. Maze path




```
void mazepath(int sr, int sc,
        int er, int ec, string ans)
{
    if (sr == er && sc == ec)
    {
        cout << ans;
        return;
    }

    if (sr+1 <= er)
        mazepath(sr+1, sc, er, ec, ans + "V");

    if (sc+1 <= ec)
        mazepath(sr, sc+1, er, ec, ans + "H");
}
```

if we want to return count i.e total no. of path

```
int mazepath(                    )
{
    if (sr == er && sc == ec)
    {
        cout << ans;
        return 1;
    }
int count = 0;
    if (sr+1 <= er )
    {
        count += mazepath(                    );
    }
    if (sc+1 <= ec)
    {
        count += mazepath(                    );
    }
return count;
```

* To return No. of path count.

2nd/Nov/2019.

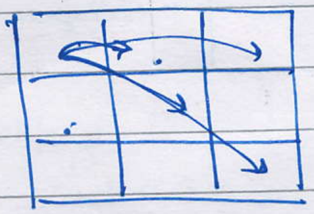
Data Structure

Prerna Singh following

Gurgaon

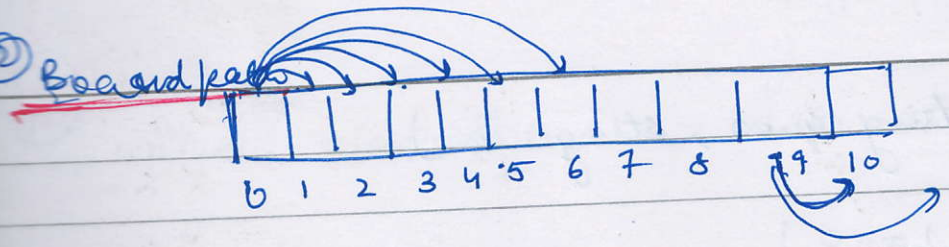
Kolkata

1. maze path multimove
2. board path
3. permutation aaa]x6.
4. without permut repetition aaa → aaa



```
void mazepath(int sr, int sc, int er, int ec, string ans).
{
    if (sr==er && sc==ec)
    {
        cout<<ans<<endl;
        return;                              | return 1;
    }
    int count=0;
    for( int i=1; i+sr <=er; i++)
    {
count +=    mazepath( i+sr, sc, er, ec, ans+"v"+to_string(i));
    }
    for( int i=1; i+sc<=ec; i++)
    {
count +=    mazepath( sr, i+sc, er, ec, ans+"h"+to_string(i));
    }
    for( int i=1; i+sc<=ec && i+sr<=er ; i++)
    {
count +=    mazepath (i+sr, i+sc, er, ec, ans + "d" +to_string(i))
    }

    return count;
}
```

for count of paths



abc →

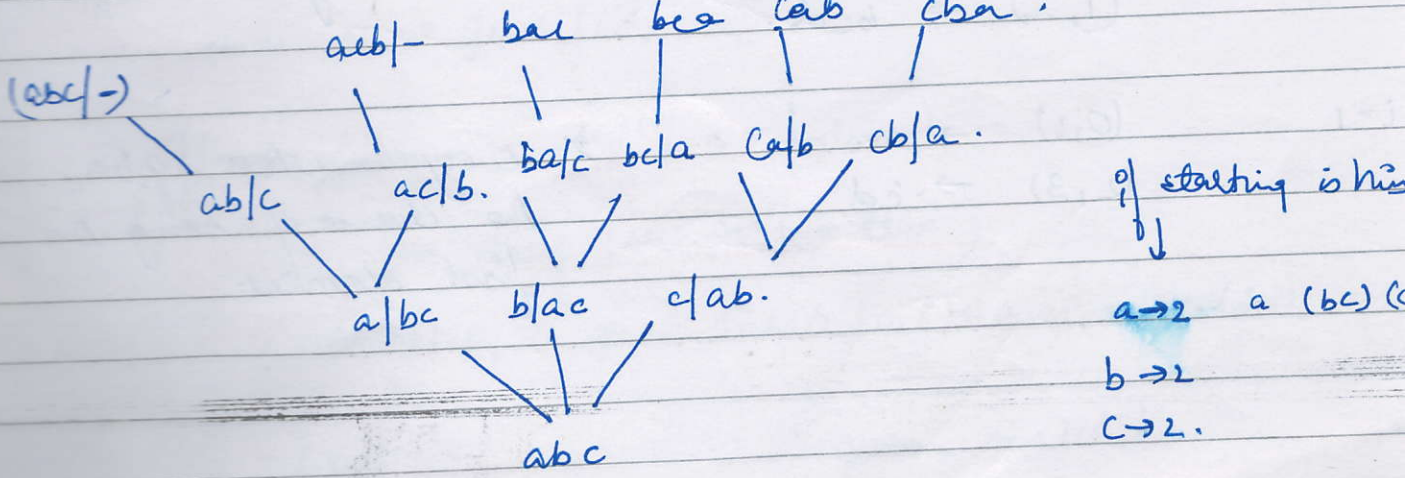
c b
↑ ↗
bc c
↑ ↗
a bc

jump of 6 or less.

```
int boardpath (int s, int d, string ans)
{
    if (s==d)
        cout << ans ;
        return 1;

    int count=0;

    for( int dice=1 ; dice<=6 && s+dice<=end; dice++ ).
    {
        count+= boardpath(.
    }
    return count;
}
```

);

(3) Permutation



```
int permutation (string ques, string ans).
{
    if (ques.length()==0)
    {
        cout<<ans<<endl;
        return 1;
    }
    int count=0;
    for(int i=0; i<ques.length(); i++)
    {
        char ch = ques[i];
        string roq = ques.substr(0,i) + ques.substr(i+1);

        count += permutation (roq, ans+ch);
    }
    return count;
}
```

abcd
0 1 2 3

bcd acd abd abc.
0 1 2 3
abcd.
0 1 2 3

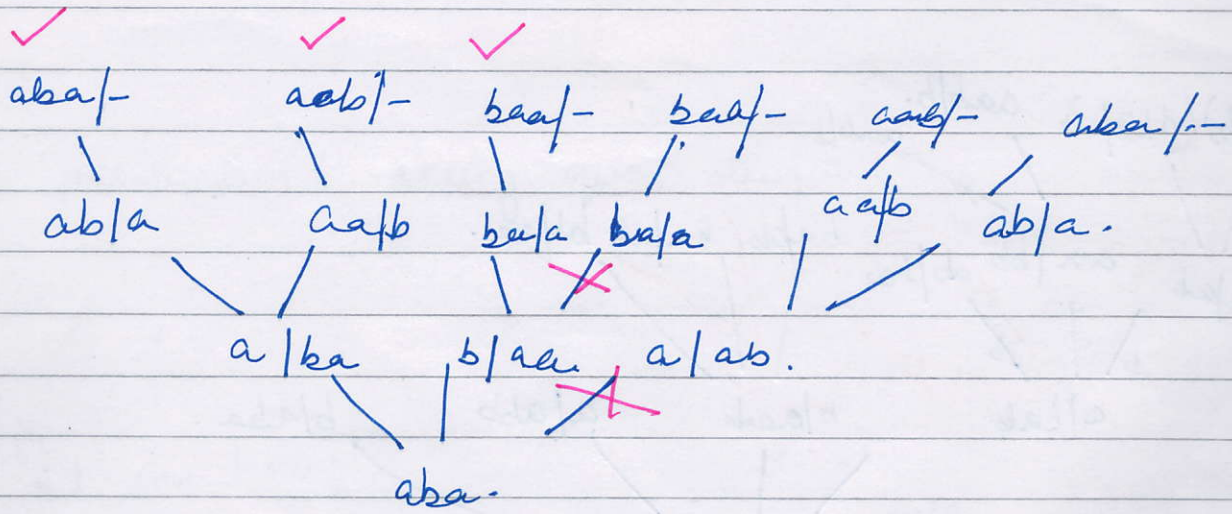
i=0 (0,0) → "" → zero se zero length ki string
(1,--) → bcd.

i=1 (0,1) → a.
(2,3) → cd

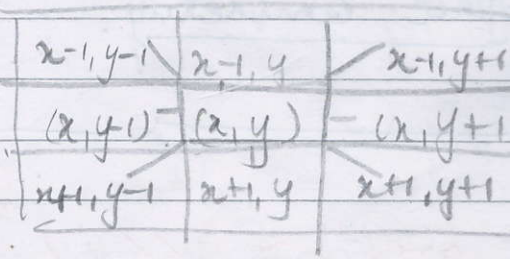
* every character is given the chance of being the first element

aabb → 4!/2!×2!

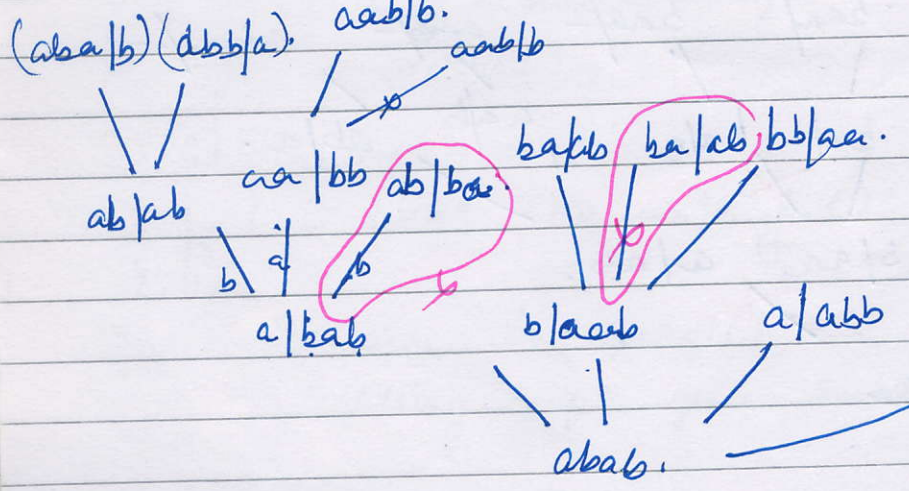
(4) without repetition - Permutation




```
int noRepition ( string ques, string ans).
{
    if(ques.length()==0)
    {

    }
    int count=0;
    vector<bool> mapping(26,false);
    for(int i=0; i< ques.length ; i++)
    {
        char ch = ques[i];
        if(mapping[ch-'a']==false)
        {
            mapping[ch-'a'] = true;
            string roq = ques.substr(0,i)+ques.substr(i+1);
            count+= noRepition(roq, ans+ch);
        }
    }
}
```



```
  10 1110 1110
&      1000
-------------
  00 0000 1000
```



space complexity. ⟶ (26 × str.length()) × 1 (size of boolean array)

* arr[i] ≃ (1<<i) {using bit mapping for reduce the space complexity}

(arr & (1<<i) != 0)

↳ if show '0', that means bit of ith position is 0 & vice-versa.

arr	bit
① * arr[i]	* (1<<i)
② * if (arr[i] == true)	* (arr & (1<<i)) != 0)
③ * arr = true.	* arr | = (1<<i) ↳ or on the bit of

if you know bit masking you will use 4 byte. space

(4) arr = false;

;

;

* or ~(1<<i)
↳ off the bit of the ith position

int permutation(string ques, string ans)

0 ——— 0000

memory 32 bit

int count=0;

int mapping=0; → mapping ka number ka array of size 4.

for(i=0; i<ques.length(); i++)

{

char ch = ques[i];

if((mapping & (1<<(ch-'a'))) == 0)

{

mapping = mapping | (1<<(ch-'a'));

string roq = ques.substr(0,i) + ques.substr(i+1);

count += permutation(roq, ans+ch);

}

}

(32 x sizeof string) bits

2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4.

(7)

1 << 7

10000000

| 10000000

1 0 0 0 0 0 0 0

1 << 4

(4)

4 ⇒

(7)

(1 0)

0 ⇒ 1

2, 2, 3, 3, (4)

(2n) + 1

Unique occurrence in k times repeated element.

2 → 7
3 → 7
10 → 7
14 → 1
5 → 7

8 4 2 1
0 0 1 0
0 0 1 1
1 0 1 0

0 1 0 1

| 1 | 1 | 3 | 2 | x7

adding bits

| 7 | 7 | 21 | 14 |

1 1 1 0

add 4 bit

| 8 | 8 | 22 | 14 |

take modulus

| 1 | 1 | 1 | 0 | → no. is 14.

→ repeated only once

a | b | c | d | e

(111010) (1001) (110) (10) (111111)

111010 10

111010 10

111010 30.

33303 0

if we modulus addition with k it will make it 0

((a+b+c+d)xk +e) % k.

make 32 bit size array.

2	3	3	4	4	3	2

2 010

3 011

4 100

make 32 size array

																												1	2	1	
31																															0

8. find x/y without / * operator

```
int uniqueInt(vector<int>& arr, int k).
{

    vector<int> bits(32,0);

    for(int ele : arr)
    {
        for(int i=0; i<32; i++)
        {
            int mask = (1<<i);

            if((ele & mask)!=0)      // check if bit is set
            {
                bits[i]++;
            }
        }
    }

    int ans=0;

    for( int i=0; i<32 ; i++)
    {
        if( bits[i]%k != 0).
        {
            ans |=(1<<i );      ] setting the bits
        }
    }
```

```
int uniquelink(
        int ans=0;
for(int i=0; i<32; i++)
{       int count=0;
    for (int ele:. arr).
    {
        int mask=(1<<i);
        if((ele & mask)!=0)
        {
            count++;
        }
    }
{   if(count%k!=0).
    {
        ans |=(1<<i);
    }
}
}
```

```
0 0 0 1 0  → 3.
0 0 0 1 1  → 3
0 0 1 0 0  → 3
0 0 1 1 1  → 1
0 1 0 1 0  → 3
```

(1) In vector array we were adding all the elements bits at that position then in code taking modules over it and creating no.

(2) Here we are having 32 loop first of this will help us in getting the last position of each element.

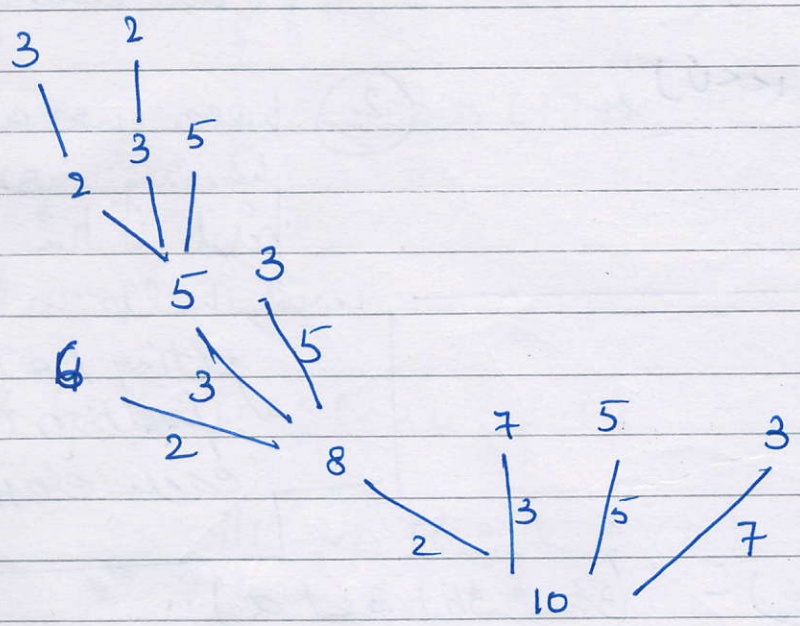
a, b, c, d, e.

$$\frac{3*(a+b+c+d+e) - (3a+3b+3c+a)}{2}$$

maths

3rd/NOV/2019.

coins | 2 | 3 | 5 | 7 | tar = 10.



```
int coinchangepermut(vector<int>& arr, int idx, int target, string ans
{
    if (target == 0)
    {
        cout << ans << endl;
        return 1;
    }
    int count = 0;
    for (int ele : arr)
    {
        if (target - ele >= 0)
        {
            count += coinchange(arr, idx, target - ele, ans + to_string
        }
    }
    return count;
}

int coinchangecomb(vector<int>& arr, int idx, int target, string ans
{
    if (target == 0)
    {
        cout << ans << endl;
        return 1;
    }
    int count = 0;
    for (int i = idx; i < arr.size(); i++)
    {
        if (target - arr[i] >= 0)
        {
            count += coinchange(arr, i, tar - arr[i], ans + to_string(
        }
    }
```

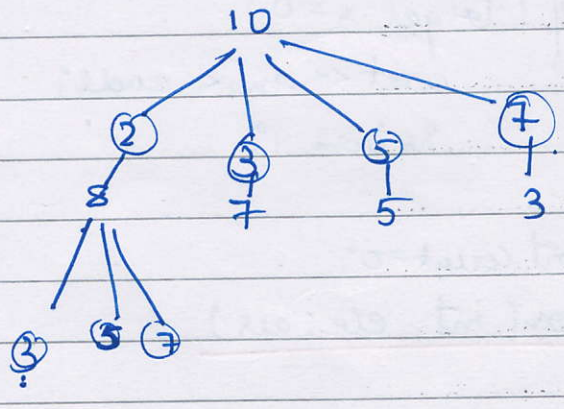
22222.
2233

22222 2233
235
37
55

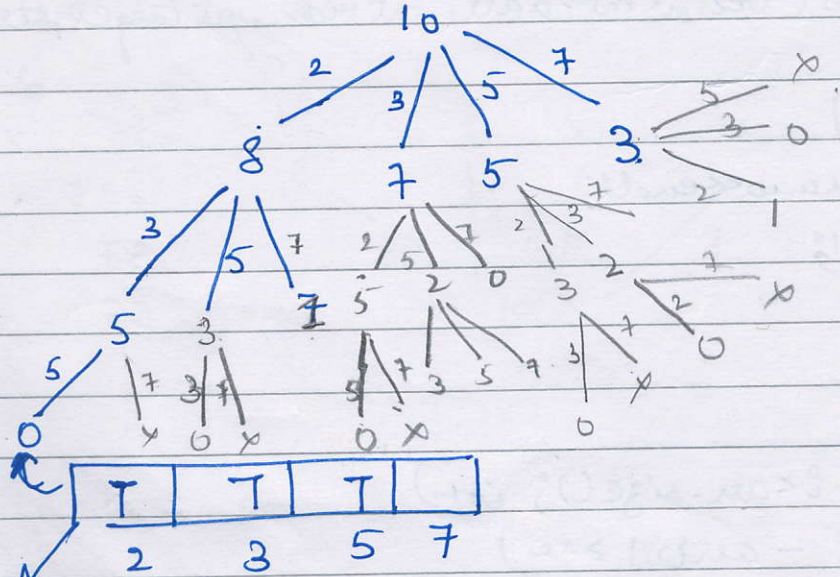
Combination
no repeated element.

235
37

combcomb02.




```
for (int i=index; i< arrsize.(); i++)
{
    if (tar - arr[i] >=0)
    {
        combcomb02(arr, i+1, target-arr[i], ans+arr[i]);
    }
}
```



for this call in order to make sure that no element visits it again.

3! +2!
6 +2 =8

235
253
352
325
523
532
37
73.

permutation without repetition of element

permutation

```
int coinchange (vector<int> & coin, vector<bool> & isDone, int target
{
    if (target == 0)
        cout << ans
        return 1;

    int count = 0;
    vector
    for(int i = 0; i < coin ; i++)
    {
        int ele = arr[i];

        if (target - ele >= 0 && ! isDone [i]).
        {
            isDone [i] = true;
            count += combination(arr, isDone, target-ele,
                                 ans + to_string(arr ele));
            isDone [i] = false;
        }
    }
    return count;
}.
```

boolean array is being marked.

2 3 5
2 5 3
3 5 2
3 2 5
5 2 3
5 3 2
3 7
7 3

permutation
No coin repeated.

Target sum equal set.

10	20	30	40	50	60	70	80

set1 set2.

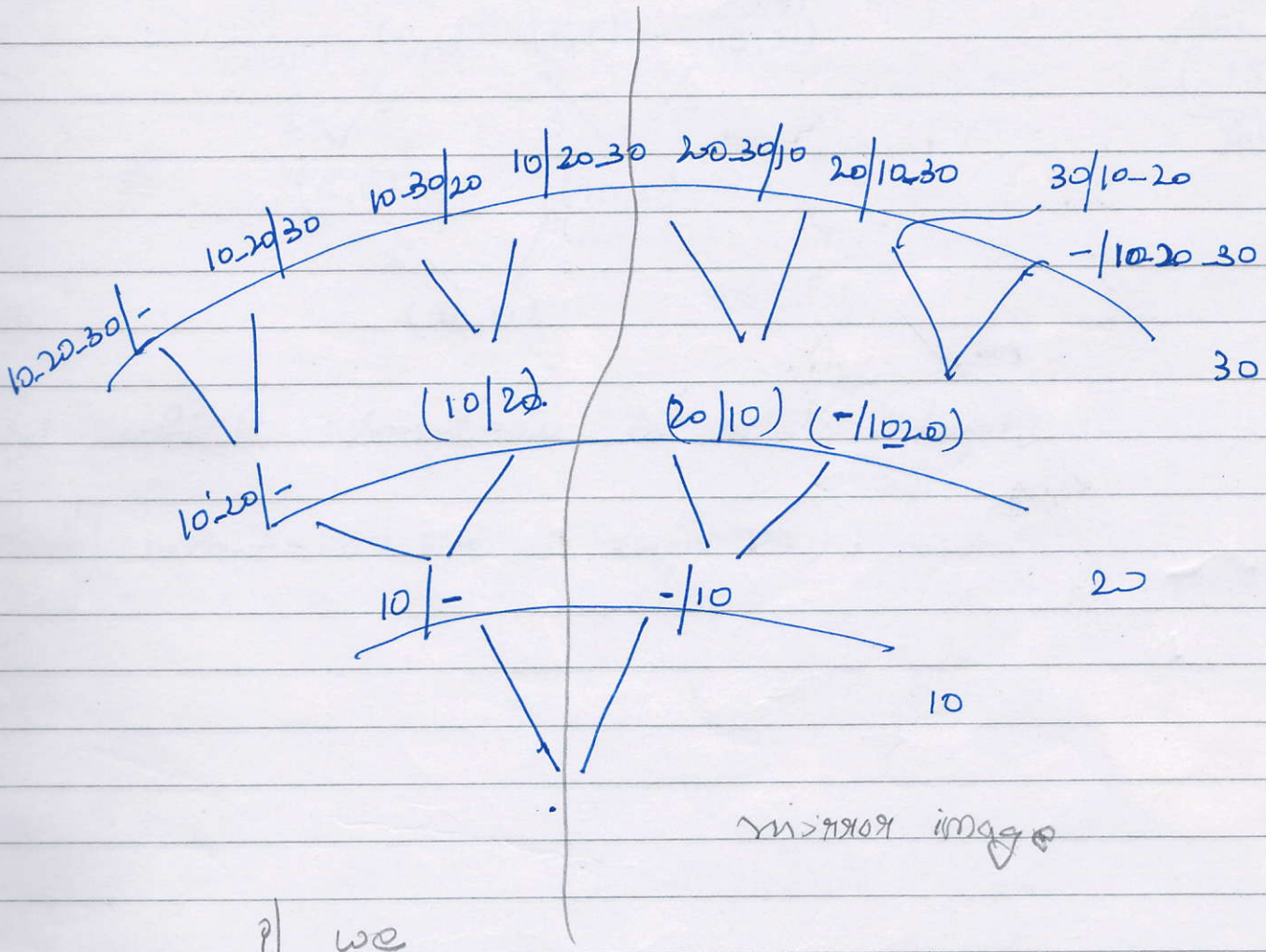
because we want combination because once we have used 10 we won't be using 10 again.

```
int equalset( vector<int> & arr, int set1, int set2, string ans1, string ans2)
                int vidx.
{
    if( vidx == arr.length()-1)
    {
        if(set1 == set2)
        {
            cout << ans1 << "=" << ans2 << endl;
            return 1;
        }
        return 0;
    }

    int count = 0;

    count += equiset(arr, vidx+1, set1+arr[vidx], set2
                ans1 + to_string(arr[vidx]), ans2 );

    count += equiset(arr, vidx+1, set1, set2+arr[vidx],
                ans1, ans2 + to_string(arr[vidx]));

    return count;
}
```

```
if(vidx == arr.length-1 || set1 == set2)
{
    return 1.
}
```

10 20 - 30

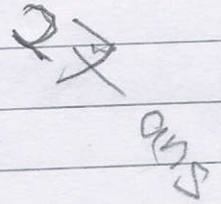
This will print two answers.



If we define the set for first element it will make sure that it will present mirror image.
and index should be 1.

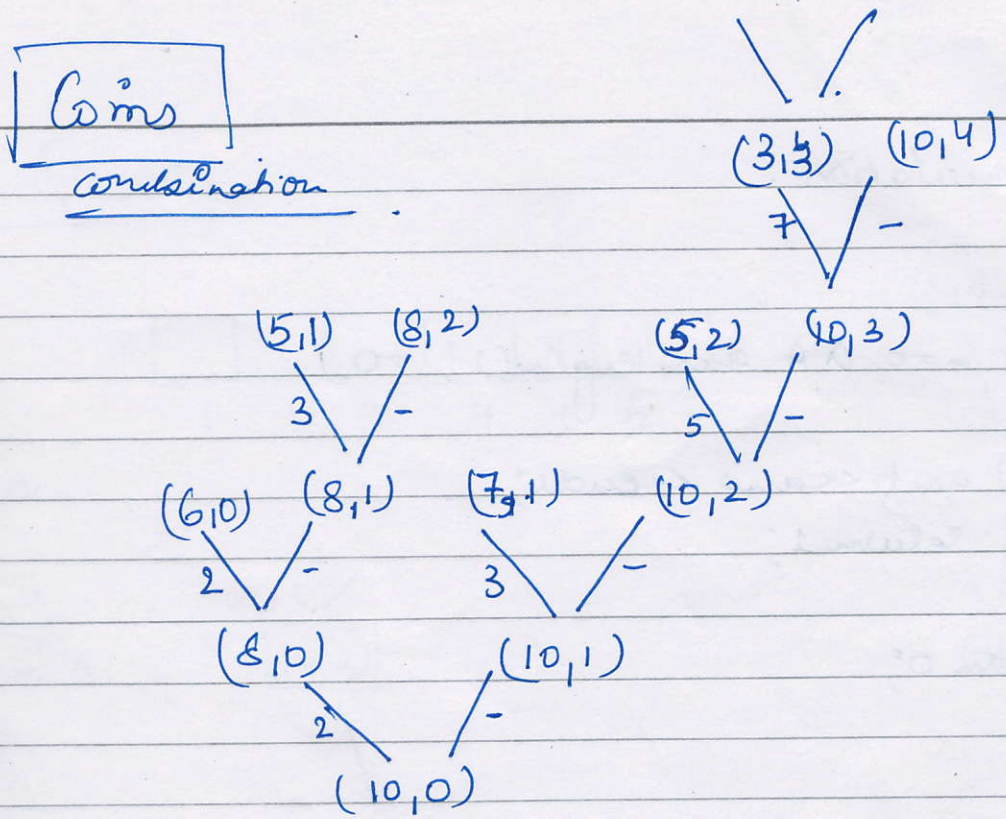
give 3 option.

element can go to set1 or set2 or no set



Coins

combination.



152
160

int coin (vector<int> arr, int vidx ; int target,

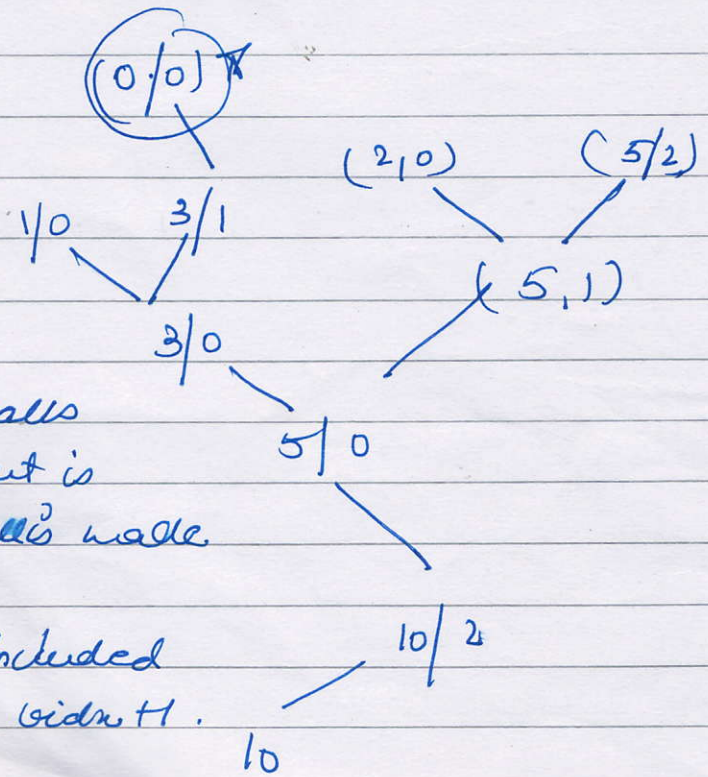
{ if(vidx == arr.size() || target == 0)

19/

10/X

Com permutation.

```
if (target == 0 && ans.length() != 0).
{
        cout << ans << endl;
        return 1;
}
    return 0;
}

int count = 0;
if ( target - arr[vidx] >= 0)
{
    count += combinationSum(arr, 0, target - arr[vidx], ans + to_string
                                                        (arr[vidx]));
}
```

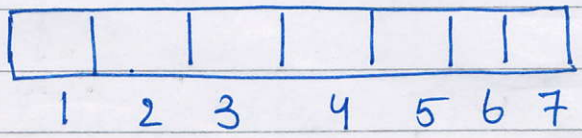


523.

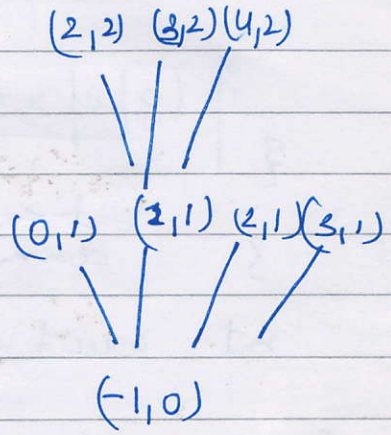
here make 2 calls one if element is accepted call's made from "0" if coin is not included calls from vidx+1.

$\frac{5!}{2!}$ = $\frac{5}{(n-2)!}$ $\frac{5!}{3!2!}$ $\frac{5\times4\times3!}{3!\times2!}$ = 10 $\frac{5!}{3!}$

N Queen.



Queen.



queen 3. (4,3)

queen 2 (3,2) (4,2)

level = queen 1 (1,1) (2,1) (3,1) (4,1) → queen 1.

→ box position

(0,0)

```
int nqueencomb( int boxes, int tnq, int qloc, int qfes, string ans).
{
    if( qfes >= tnq || qloc >= boxes.)
    {
        if(qfes == tnq)
        {
            cout << ans;
            return 1;
        }
        return 0;
    }
    int count = 0;
    for( int i = qloc+1; i <= boxes; i++)
    {
        count += queencomb( boxes, tnq, i, qfes+1, ans + "b" + to_string(qloc) + to_string(qfes));
    }
    return count;
}
```

bon queen
→ 7, 3*

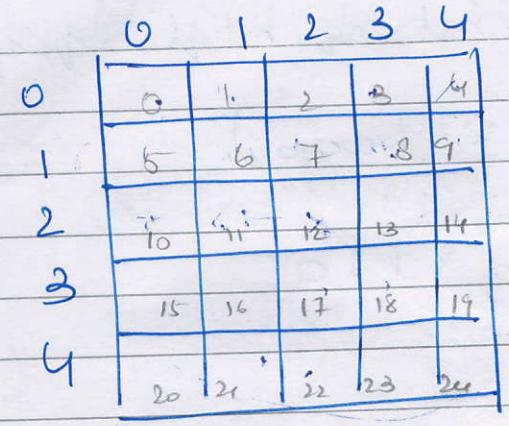
ans 336

```
nqueen(int bones, int tnq, vector<bool> loc, int qpsf,
                                    string ans)
{
    if (qpsf == tnq)
    {
        cout << ans << endl;
        return
    }
    int count = 0;

    for(int i=0; i<=bones; i++)
    {
        if (!loc[i])
        {
            loc[i] = true;
            count += nqueen(bones, tnq, loc, qpsf+1,
                            ans + "b" + to_string(i));
            loc[i] = false;
        }
    }

    return count;

}
```

On level we are keeping boxes

```
int nqueen combi(int boxes, int tnq, int qloc, qpsf, string ans)
{
    if(qpsf == tnq || qloc >= boxes)
        if(qpsf == tnq) { count++; ans; return 1; }
        return qpsf == tnq ? 1 : 0;
}

int count = 0;

return nqueen(boxes, tnq, qloc+1, qpsf+1,
              ans + 'b' + to_string)
       ||
       nqueen(boxes, tnq, qloc+1, qpsf, ans);

from main pass -1 as index of qloc.
}

int vidu.
vector<bool> qloc.

int nqueen permut(int boxes, int tnq, int qloc, int qpsf, string ans)
{
    if(qpsf == tnq || qloc >= boxes).

    int count = 0;

    pass 0 at vidu.
    if -1 check for qloc+1

    if(!loc[qloc])
    {
        loc[qloc+1] = true;
        count += queen(boxes, tnq, 0, loc, qpsf+1, ans + 'b' + to_string
        loc[qloc+1] = false;
    }
    count += queen(boxes, tnq, qloc+1, qpsf, ans.

    return count;
}
```

N queen

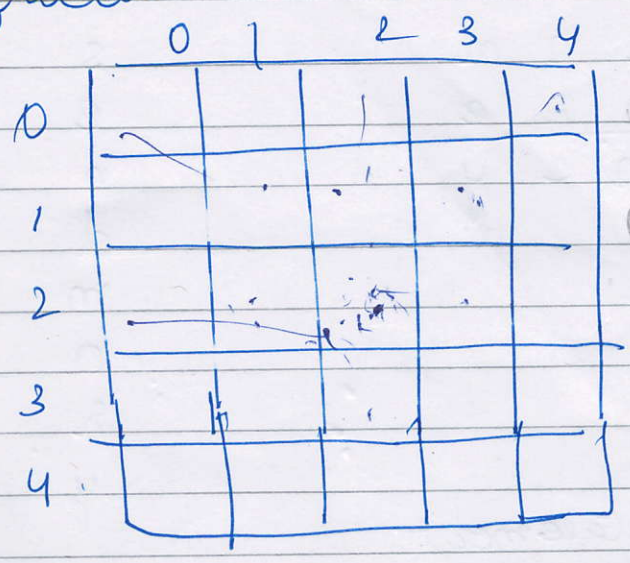


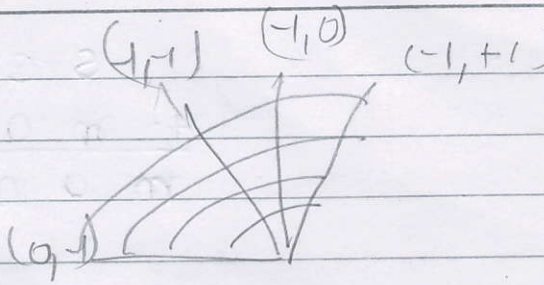
queen combination & permutation

```
int nQueen(vector<vector<bool>>& boxes, int tnq, int qloc,
           int qpsf, string ans)
{
    if(qpsf == tnq)
    {
        cout << ans;
    }
    return 1;

    for(int i=qloc+1; i<boxes.size()*boxes[0].size(); i++)
    {
        int x = i/boxes.size();
        int y = i%boxes.size();

        if(isSafe(boxes,x,y))
        {
            boxes[x][y] = true;
            count += nQueen(boxes, tnq, i, qpsf+1, ans+"("+to_string(x)+"-"+to_string(y)+")");
            boxes[x][y] = false;
        }
    }
    return count;
}
```

we won't use bsafe we will get combi answer as 1820.

N Queen.





bool isvalid (vector<vector<bool>> & boxes, ~~int rq, int cq.~~
~~int qfest, string ans~~
int r, int y.)

int dir [4][2] = { {0,-1}, {-1,-1}, {-1,0}, {-1,1} };

```
{ for(int d=0; d<4; d++)
  { for(int rad=1; rad<boxes[0].size(); rad++)
    { int r = x + dir[d][0] * rad;
      int c = y + dir[d][1] * rad;
```

this multiplication with rad will help in entire traversal previously filled only.

```
      if( r>=0 || c>=0
      || r< boxes.size()
      || c < boxes[0].size())
          && boxes[r][c] == true)
      {
          return false;
      }
  }
}
return true;
```

14/11/2019

```
  s e n d
+ m o r e
---------
m o n e y.
```

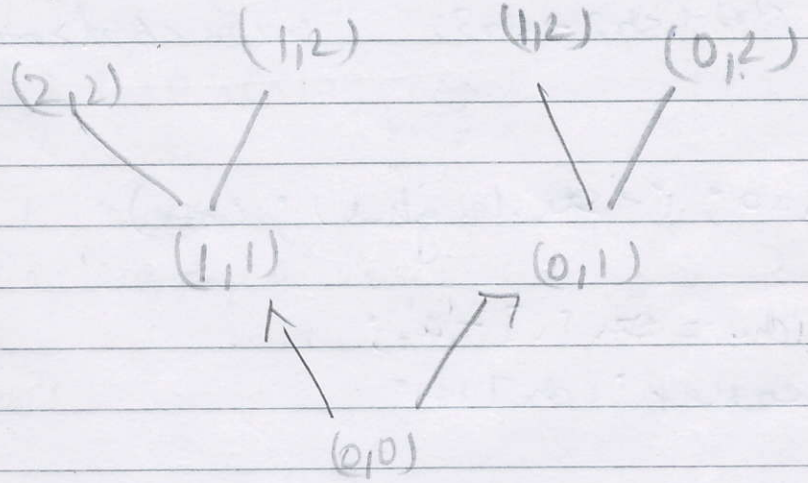
```
s
e
n
d
m
o
r
y.
```

Subsequence nqueen placement.

```
int queensub(vector<vector<bool>>& boxes, int tnq, int qpsf,
    if(vidx == boxes.size() * boxes[0].size())
        { if(qpsf == tnq) return 1; } return 0;
    int x = vidx / boxes[0].size();
    int y = vidx % boxes[0].size();

    if(isSafeToplace(boxes, x, y))
    {
        boxes[x][y] = true;

    //call. count += nQueen(boxes, tnq, qpsf+1, vidx+1, ans+"-")
        boxes[x][y] = false;
    }
    count += nQueen(boxes, tnq, qpsf, vidx+1, ans);
//call.
```

// queen is placed. we increment the boxes

// queen is not placed. we increment the box and




```
  s e n d
  m o r e
---------
m o n e y
```

s
e
n
d
m
o
r
y.

8

each character is given option for 0,1,2,3,4,5,6,7,8,9

each puzzle is of two type either permutation & combination.

Crypto

Take a 26 size array
make unique string → sorted string.

permutation

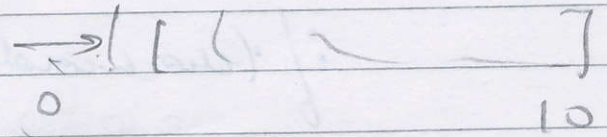
```
void crypto() {
    string str = str1 + str2 + str3;

    for(int i=0; i<str.length(); i++)
    {
        int idx = str[i] - 'a';
        freeMap[idx]++;
    }

    string ans = "";

    for(int i=0; i<26; i++)
    {
        if(freeMap[i] != 0)
        {
            str += (char)(i + 'a');
        }
    }
}
```

```
string str1 = "send";
       str2 = "more";
       str3 = "money";
vector<int> maping(26,0);
vector<bool> mark(10,-1);
```

16th/Nov/2013

(1)

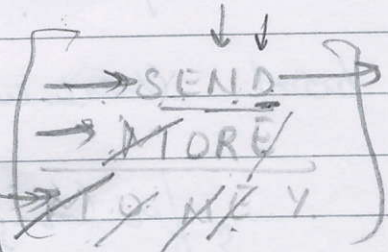
```
int cypho (String str, int idx)
{
    if (idx == str.length())
    {
        if (strNum1 + strNum2 == strNum3)
        {
            return 1;
        }
        return 0;
    }
    for(int i = 0; i < 10; i++)
    {
        // used.
        // 1 numused
        // 2 making

        // call

        // mapping
        // numused
```



map [26]

String +=

[ ]



for (0 <= 9 ; i++)

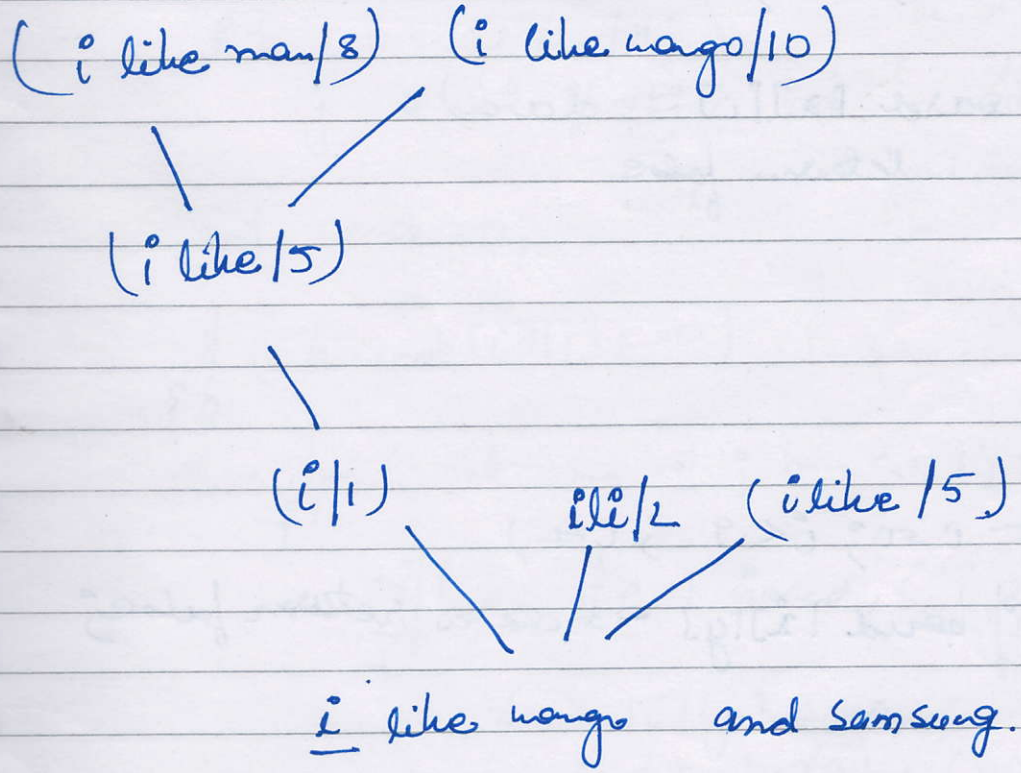
SENDORY

if ( )

map = i

(2)

```
int wordbreak (string word, string ans)
{
    if (word.length() > 50)
    {
        cout << ans << endl;
        return 1;
    }

    string temp = "";
    int count = 0;

    for (int i = 0; i < word.length(); i++)
    {
        temp += word[i];

        if (checkwordinDictionary(temp))
        {
            count += wordbreak(word.substring(i), ans + temp + " ");
        }
    }

    return count;
}

vector<string> dict = {};

bool checkword(string word)
{
    for (string s : dict)
    {
        if (s.compare(word) == 0)
        {
            return true;
        }
    }
    return false;
}
```


(i like mango/8)
(i like mango/10)
(i like/5)
(i/1)
i like/2
(i like/5)
i like mango and samsung.

Sudoku.

(3)

```
bool isSafeToPlace(vector<int>>board, int x, int y, int data)

    // Row.

    for(int i=0; i<9; i++)
    {
        if(board[x][i] == data)
            return false
    }

    // col.

    for(int i=0; i<9; i++)
    {
        if board[i][y] == data return false;
    }

// 3x3 check.
    int x = (x/3) * 3;
        y = (y/3) * 3;

    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++)
        {
            if (board[x+i][y+j] == data)
                return false;
        }
    }
return true.
```

if we backtrack on 2D array it becomes difficult because the one position will be backtrack & there is nothing behind.

To overcome this we will represent it as 1D.

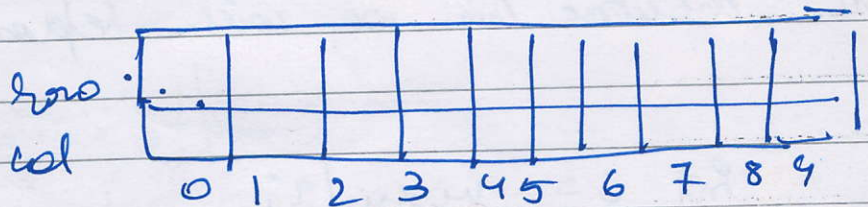
```
if(idx == 9*9)
    find board;
for(vector<int> vec: board)
    for(int de: vec)
        cout<<de;
    cout<<endl;
```

```
int i = idx/9;
    j = idx%9;

int count = 0;

if ( board[i][j] != 0 )        1D se 2D
{
    for( int num = 1 ; num <= 9; num++)
    {
        if ( isSafe ( board, i, j, num)
        {
            board[i][j] = num;
            count += sudoku( idx+1 );
            board [i][j] = 0;
        }
    }
}
else
{
        count += sudoku (idx+1);
}

return count;
```

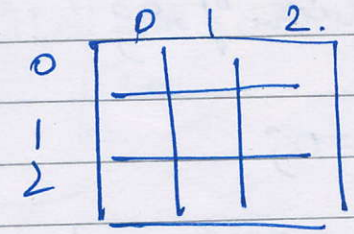
(12:9)

10.
11.

vector<int> rows(9,0);
vector<int> col(9,0);
vector<vector<int>> mat(3,vector<int>(3,0));

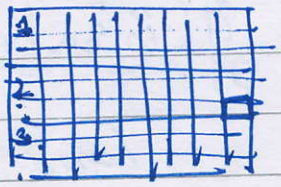


int sudoku



111

with help of bits we can check this quickly.



int mask = 1<<num;

if(((rows[i] & mask)==0 && col[i] & mask)==0
&& (mat[i/3][j/3] & mask ==0))
{

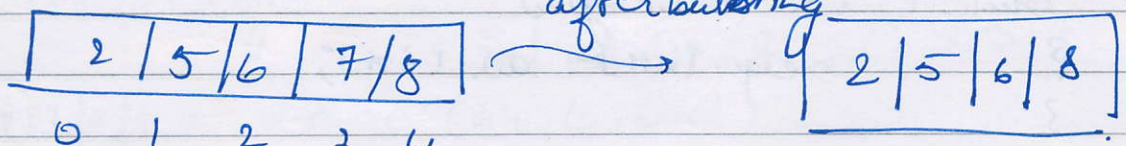
board[i][j] = num;
rows[i] |= mask;
col[j] |= mask;
mat[i/3][j/3] |= mask;

count += sudoku(board, i+1, j+1);
board[i][j] = 0;
rows[i] ^= mask;
col[j] ^= mask;

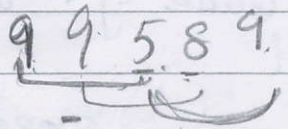
populating the row, col, mat

```
{ for(int i=0 ; i<9; i++)
  { for(int j=0; j<9 ; j++)
    { if(board[i][j]!=0)
      { int mask = 1<<board[i][j];
        row[i] |= mask;
        col[j] |= mask;
        mat[i/3][j/3] |= mask;
      }
    }
}
```

balloons.



if you burst balloon.



arr[3]

cost = arr[2] * arr[3] * arr[4].

i-1 i i+1

if i==0 || i==length

arr[i-1] = 1 length += 1

Order in which balloon is burst.

17th/NOV/2014.

Merge two array.

arr1 | 2 | 8 | 12 | 14 | 18 | 36 |

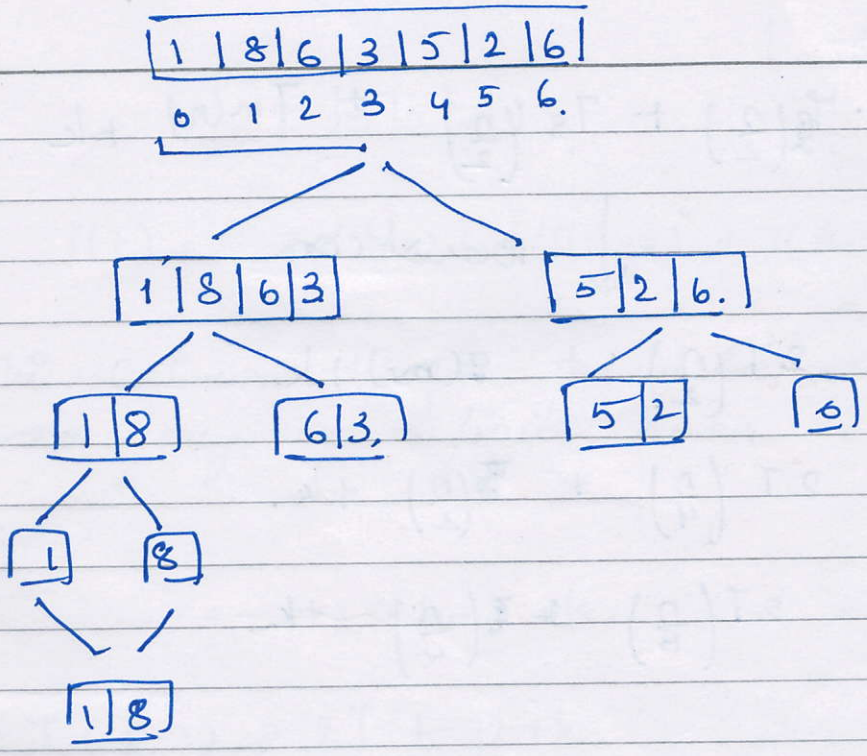
arr2 | 4 | 6 | 12 | 17 | 29 | 54 | 94 |

```
int[] merge = new int [ arr1.length + arr2.length].
  while( i< arr1.length. && j < arr2.length)
{
        if ( arr1 [ i] < arr1[j].
       {
              merge [k++] = arr1[i++] ;
       }
       else.
              merge [k++] = arr2[j++];
       }
}
  while ( i< arr1.length)
  {       merge [k++]= arr1 [i++];
  }
  while (j < arr2.length)
  {
        merge [k++]= arr2 [j++];
  }
```

1/ one single array is given and we assume it to be containing two array

| 1 | 3 | 7 | 9 | 15 | 2 | 4 | 6 | 11 | 13 | 14 |

for this we will be needing indexes.




```
if (li => ri)
{
    int[] base = new int[1];
    base[0] = arr[li];
    return base;
}
int mid = (li+ri) >>>1;
```

Time complexity.

```
int[] left = merge (arr, lo, mid);
int[] right =      (_, mid+1, hi);

int[] merge = mergeSort(left, right);
return merge;
```

time complexity depends on those 3 calls.

T(n) =

$$T(n) = T_1\left(\frac{n}{2}\right) + T_2\left(\frac{n}{2}\right) + T_m(n) + k.$$

↓ worst com

$$T(n) = 2T\left(\frac{n}{2}\right) + g(n) + k.$$

$$2T\left(\frac{n}{2}\right) = 2^2T\left(\frac{n}{4}\right) + 2g\left(\frac{n}{2}\right) + 2k.$$

$$2^2T\left(\frac{n}{4}\right) = 2^3T\left(\frac{n}{8}\right) + 2^2g\left(\frac{n}{4}\right) + 2^2k.$$

⋮

we are multiplying so that we can cut out the common terms.

$$T(n) = 2^kT\left(\frac{n}{2^k}\right) + (n+p)k.$$

t target

coin 2 | 3 | 5 | 7.

$T(t) = T(t-2) + T(t-3) + T(t-5) + T(t-7) + k.$

[If minimum height would be when we will choose only 2 infinite times

worst case:

$= 2T(t-2) + k.$

$T(t-2) = 2T(t-4) + k.$

$T(t-4) = 2T(t-6) + k.$

!

$T(t) = 1^{\left(\frac{t}{2}\right)} + (1 + 1^2 + \dots)k.$

height of 1 is target smallest no.

sudoku.

$T(n) = \sum_{i=1}^{9} n_i + n^2$

$= n^2 \sum_{i=1}^{9} n_i$

Q1

q^{n^2}

$T(n) =$ $\cdot T$.

$q\,T(n) \simeq q^2 T(n-1) + \dots$

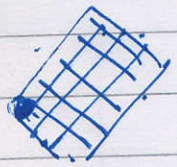
$q^{n^2} + q^{n^2} \cdot n^2$.

Sudoku. $(q^{n^2})(n^2)$ [7]

with bit q^{n^2}

q

$(1 \dots q)^{n \times n}$



jitni call nikal kari kai ho use maximum weight jaha tak ja sakte ho.

Sort 0,1

0	1	1	1	0	0	1	0	1	1	0	1	0	1	1	0

```
public static void sort01(int[] arr.
    int pt=0
    int itr=0;
    while (itr < arr.length)
    {       if (arr[itr]==0)
            {   swap (arr,ptr,itr);
                pt++;
            }
            itr++;
    }
```

```
public static void pivot ( int [] arr, int li, int ri, int pivot
{
        int  pt = li;
        int itr = li;
```

11 | 2 | 3 | 26 | 1 | 4 | 1 | 4 | 6 | 3 | 22 | 1 | 11 | 2 | 44 | 62

0 | 0 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 1 |

```
public static void sort0_1_2 (       )
        int pt = 0;          0
        int pt1 = 0; arr.length - 1
        int pt2 = 0;         1

                      pt1
    while ( itr < arr.length )

        if ( arr[itr] == 0 )
        {         swap(arr, pt, itr);
              pt++;
        }
        else if ( arr [ itr == 2 ]
        {
              swap(arr, pt1, itr);
               pt --;
               continue;
        }
        itr ++;
}
```

itr ++ when we get 0 or 1.

[0, pt) → 0

[pt, pt2) → 1

[pt, li] → 2

Anagram

① Subsequence

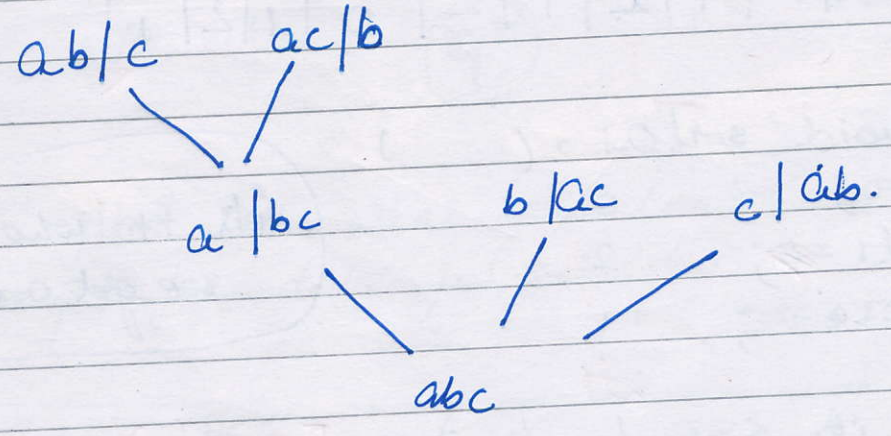
② keypad

xyz (abc) def.

	abc	def
0	1	2
ghi	jkl	mno
3	4	5
6	7	8
10	9	11

3. Permutation.

abc acb.



duplicacy.

```
arr
vector<i> arr(32,0);
int idx=k;
arr[k]= false;
      = true;

if( arr[k] ) // true.
{

}
```

```
bit.
int arr=0;
int mask= (1<<k);
arr |= mask;
arr &= (~mask);

if((arr & mask) != 0)
{

}
```

```
int permutp( string ques, string ans)
{
    if( ques.length()==0)
    {
        cout<<ans<<endl;
        return 1;
    }
    int count=0;
    int vis=0;

    for( int i=0; i< ques.length(); i++)
    {
        int mask = 1<<(ques[i]-'a');
        if ( (vis&mask) ==0)
        {
            vis |= mask;
            string str =
            count+=permutp(
        }
    }
    return count;
}
```

5th Jan 2020.

Coin — subsequence

1. permutation
   combination -

2. single coin

Subsequence

1. Combination.

target is having indices.

(0,8) (1,10) (0,10)

target is accepting coin or not

Combination - Repetition

```
(1)
if( tar >=0 || idx == all.length ).
{       if(tar >=0)
        {
        }
}
```

Combination - No Repetition

(2) combination without repetition here as (idx+1).

```
if(tar - arr[idx] >=0)
{       comb( arr, idx, tar - arr[idx], ans + str[idx]);
}
        comb( arr, idx+1, tar, ans );
```

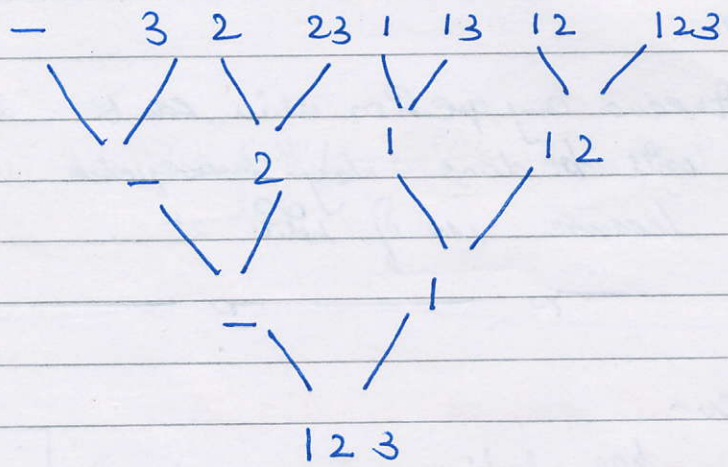
(3) permutation put "0" for permutation without repetition.

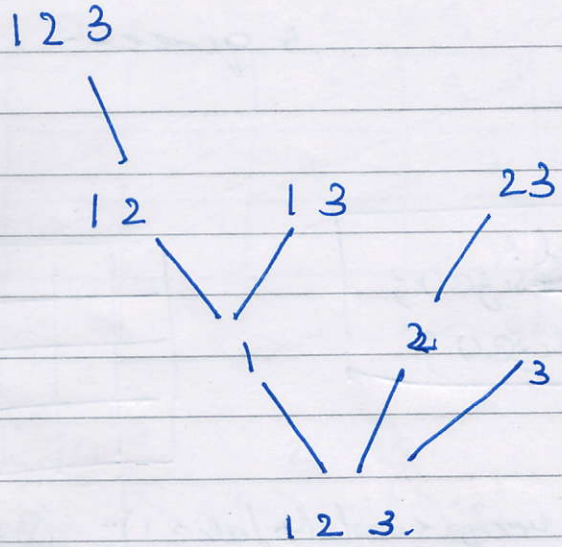
Permutation
Repetition.

(4) Permutation - No Repetition - use visited.

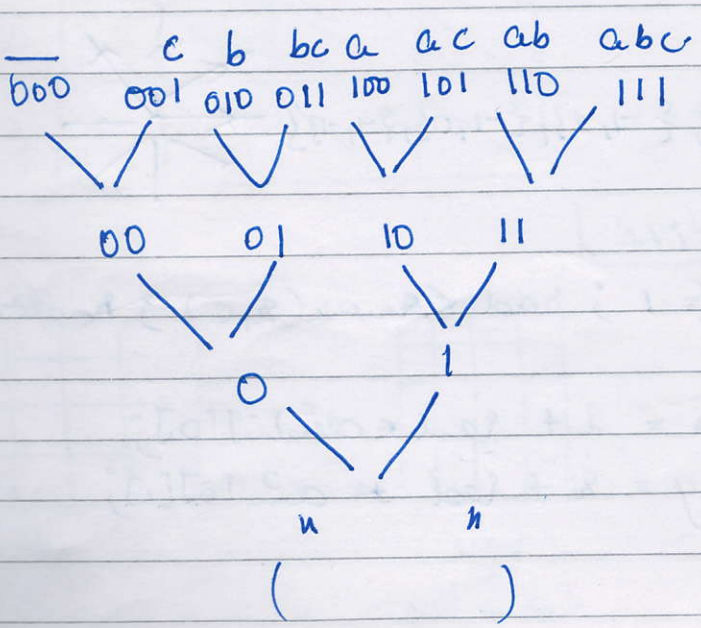
String.

[ ],
[1],
[1, 2],
[1, 2, 3],
[1, 3],
[2],
[2, 3],
[3],





Combination.
pre order
print.



Bit Tree
Same as
subsequence tree

kya the hai yani kata ke ? ?

$(x+1)^n =$

put $x=1$

2^n

It means any question which can be done by combination will be done by subsequence also & by bit because no. of bit is 2^n.

Queen

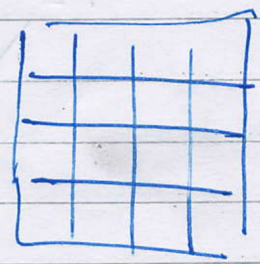
permutation

comb

3 queen

6/Jan/2020.

x = idx / col size();
y = idx % col size();

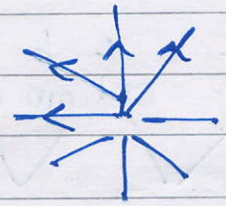


vector<vector<bool>> board (4, vector<bool>(4, false)); Q = 4

P. / C.

7/Jan/2020.

```
int[][] dir = { {0,-1}, {-1,-1}, {-1,0}, {-1,-1} }

for( int i=0; i< 4 ; i++ )
{
    for( int rad = 1 ; rad < max(r,c) ; rad++ )
    {
        int x = r + rad * dir[i][0];
        int y = r + rad * dir[i][1];
```



For permutation

take all 8 dir

for i=0

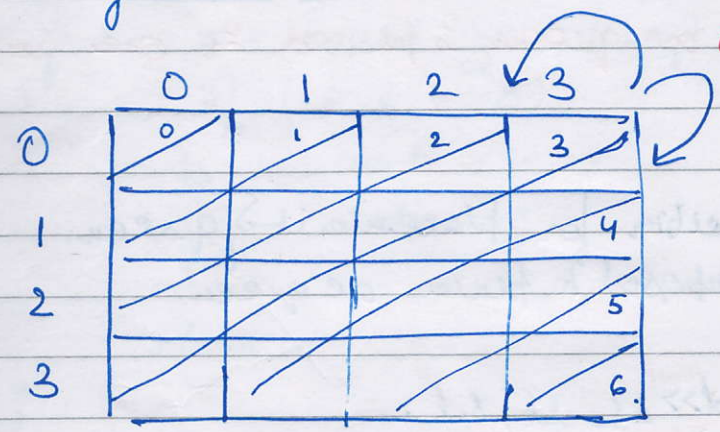
if

(! board[x][y] && isSafeToplace(, x, y))

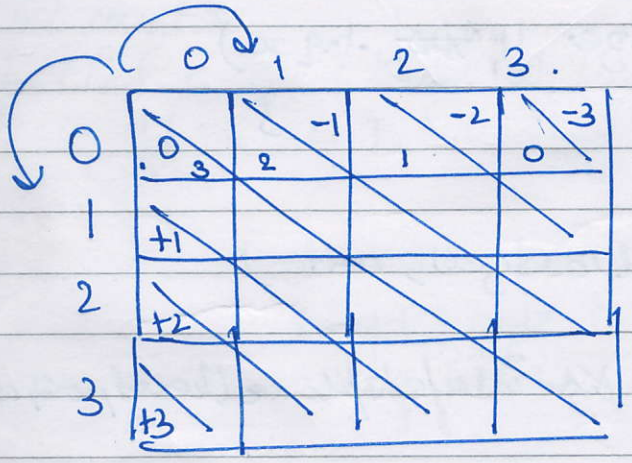
Taking shadows...

In order to update or iterate h
along we do addition x+y
for diagonal.

size of array [row+col -



0,2 | 1,3
1,1 | 2,2
2,0. | 3,1

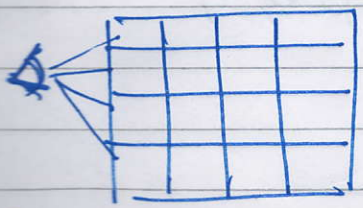


1,0 | 0,1
2,1 | 1,2
3,2 | 2,3

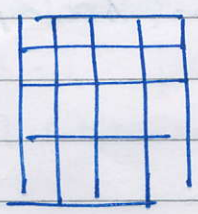
add size of col-1

[x - y + col. length - 1 .

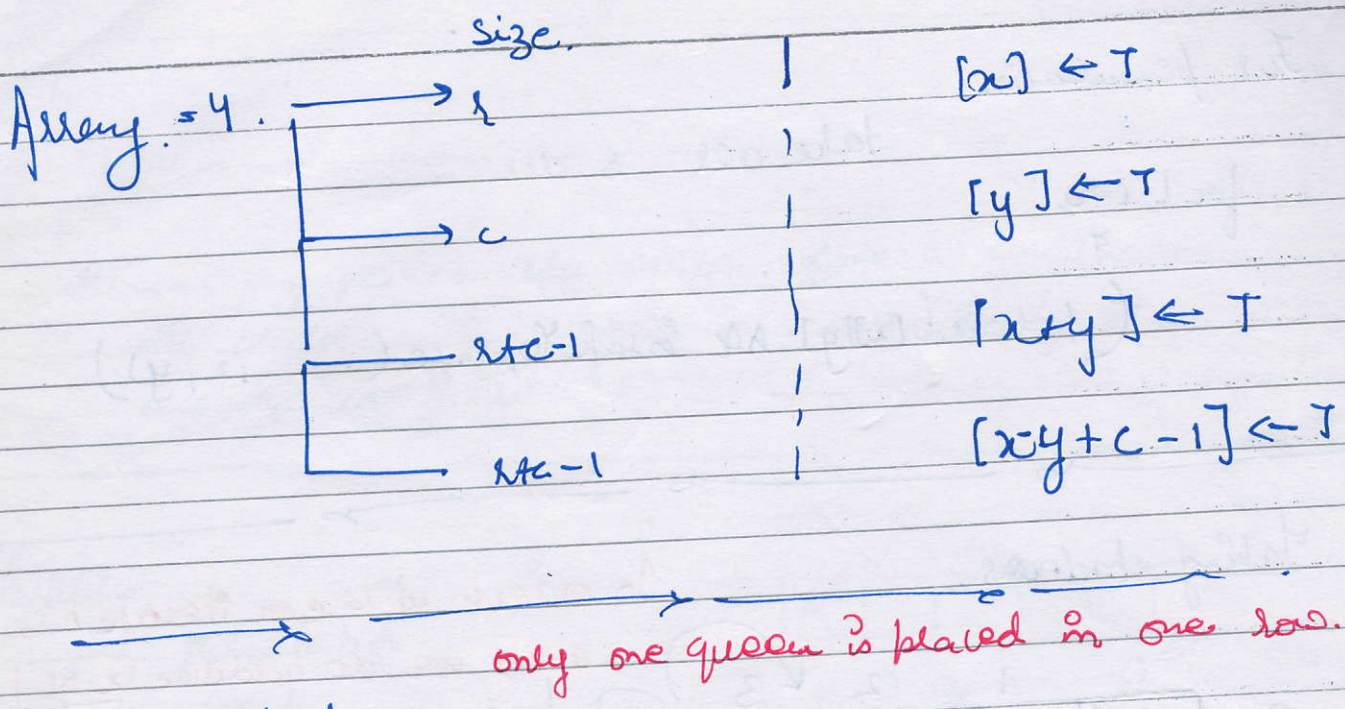
to mark whether
queen is present or



row mask.



col mask.

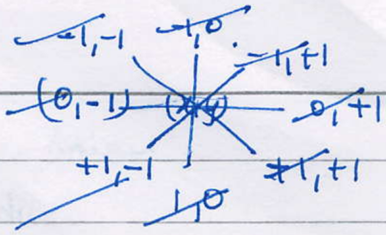


Another method.

Instead of sending next position for placement of queen. in each row. we are supposed to place one queen.

```
int queen(vector<vector<bool>> &board, int
    if( row == board.size || q == 0)
    {
        if(q == 0){
        }
    }
for(int col = 0; col < board[0].size(); col++)
    if( !board[x][y], && isSafe(board, row, col)
    {
        board[row][col] = true;
        count += queen(board, row+1, q-1,
```

11th Jan 2020




```
# define vi vector <int> vi;
# define vvi vector <vi> vvi;
# vb vector <bool> vb;
# vvb vector <vb> vvb;
```

not required

```
int queen ( int row, int tnq, vvb &board, vb &col,
                       vb &diag, vb &adiag, string ans)
int n = col.size();
  if( row == n || tnq == 0)
  {      if ( tnq == 0).
      {     cout <<ans
            return 1;
      }
      return 0;
  }
```

instead of this we pass n over here

we can do this on col.size.

```
int count = 0
for (int c = 0; c < n; c++)
     if( !col[c] && !diag[r+c] && !adiag[r-c+n-1]
     {
          col[C] = true;
          diag[r+c] = true;
          adiag[r-c+n-1] = true;

        count += queen(row+1, tnq-1, board, col, diag, adiag, ans + row + col.)
          col[c] = false;
          diag[r+c] = false;
          adiag[r-c+n-1] = false;
     }
}
return count;
```

main.

```
vb col (1, false)
vb diag (2*n, false)
vb adiag (2*n-1, false);
int n;
int queen(int r, int mq, int &col, int &d
```

```
int count = 0;
for(int c=0; c<n; c++)
    int w = 1<<c;
    x = 1<<(r+c);
    y = 1<< (r-c+n-1);

if(!(col&w) && !(diag&x) && !(adiag&y))
{
    col |= w;
    diag |= x;
    adiag |= y;

    count += queen(r+1, mq, col, diag, adiag,
                   ans+(r*n+c));

    col |= w;
    diag |= x;
    adiag |= y;
}

}
```

int col = 0;
int diag = 0
adiag = 0;

} for bits operations

Sudoku.

instead of making width call we call above one

positions where 0 is place

```
if (board[x][y] == 0){
  for (int num = 1; <= 9; num++)
    if (is safe to place)
      place
      call
      unplace
  }
} else
    call  idx+1
    count += sudoku(board, width);

void sudoku.
{
  ii calls;
  for( int i=0; i<9 ; i++)
    for(int j=0; j<9 ; j++)
    {
      calls.push back( i*9+j );
    }
}
```

board[x][y] = num

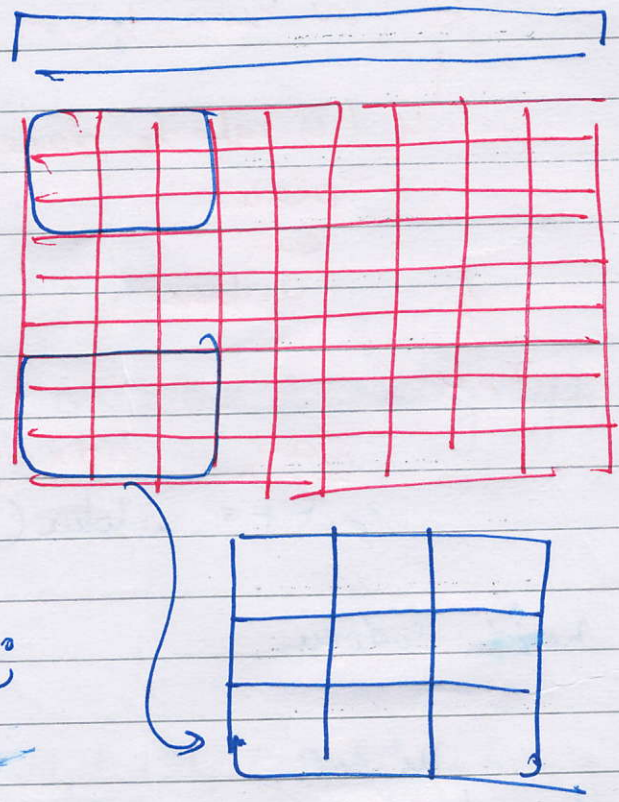
board[x][y] = 0.

increasing order mein vector ka share
iden print kar do.

```
vii V board, vii< calls, int vidx)
if( vidx == calls.size() )
{   display(board);
    return 1;
}
```

```
vi row(9,0)
vi col(9,0);
vi mat(3, vi(3,0));
global.
```

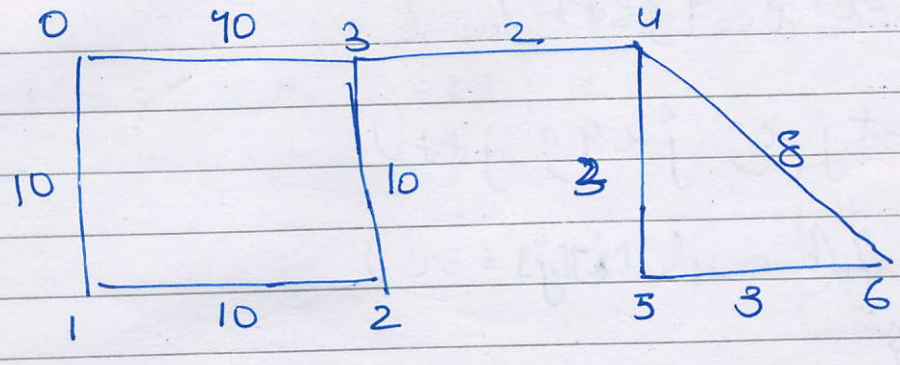
```
int x = calls[idx] / 9;
int y = calls[idx] % 9;
int count=0;

for( int num=1 ; num<=9; num++)
{   int mask = 1<< num;
    if( isSafePlace.

    if( !(row[x]&mask)
    && !(col[x]&mask)
    && !(mat.
    )

        board[x][y] = num;
        row[x] ^= mask;
        col[y] ^= mask;
        mat[x/3][y/3] ^= mask;

    count += queen(

        board[x][y] = 0;
        row[x] ^= mask;
        col[y] ^= mask;
        mat[x/3][y/3] ^= mask;
}
```



```
for( int i=0; i<9; i++)
    for(int j=0; j<9; j++)
        if(board[i][j]==0)
        {
            calls.push_back( i*9+j );
        }
        else{
            mask = 1 << (board[i][j]);
            row |= mask;
            col |= mask;
```

CROSS WORD

Graph

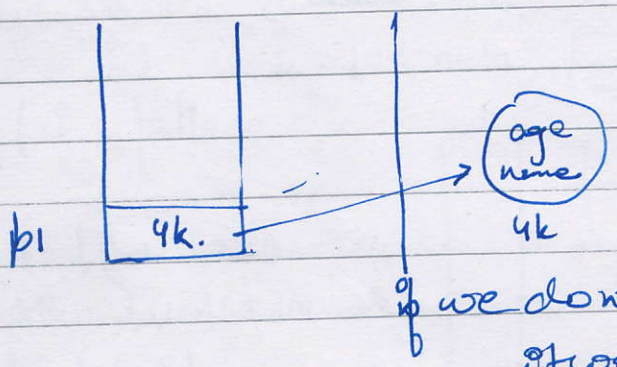


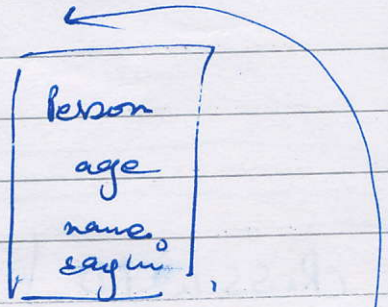
0-6 ka sabse chota rasta.

Dijkstra

Minimum spanning tree - Prims Algorithm.

Person p1 = new Person();





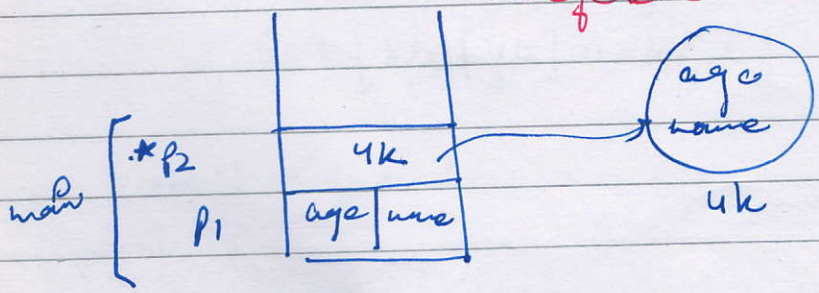
if we do not write new person, it would be of no use.

C++

Person p1; is the object

Person *p2 = new Person(); //Similar to java

↳ reference to the object.

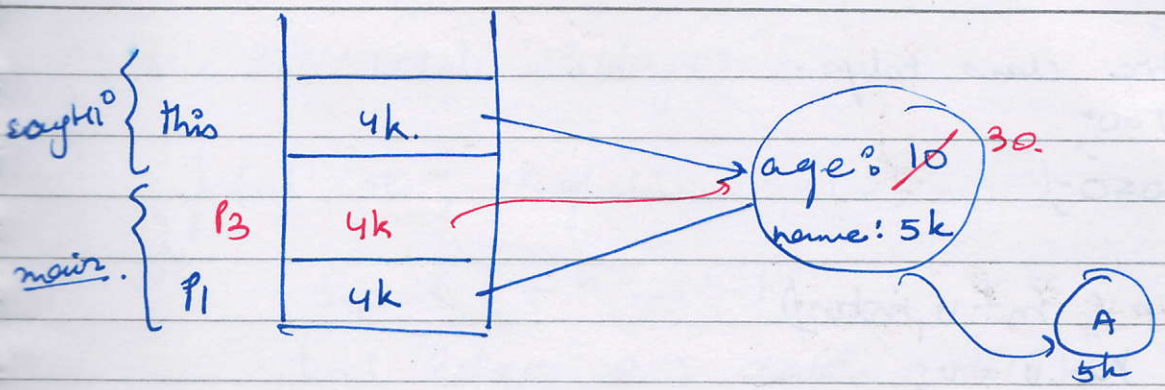


byte short int long

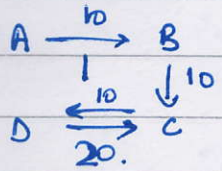
Decimal No.
NO.
bool ch.
character

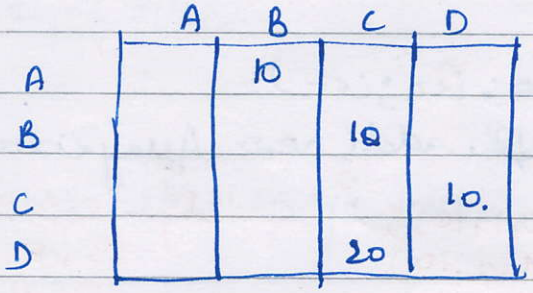
} primitive

Non primitive.



Person p3 = p1;
p3 mein whi address copy ho gya hai
now
p3.age = 30;
& we call p1.sayHi();
It will print 30 as age.





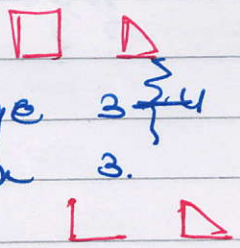
Ans

Vector<Vector<Pair<int,int>>>

```
import java.util.*;

public class Demo1
{
    public static Scanner scn = new Scanner(System.in);
```

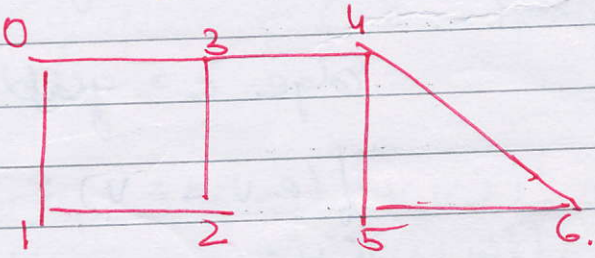
```
public static class Edge.
{       int v=0;
        int w=0;

        Edge ( int v, int w)
    {       this.v = v;
            this.w = w;
    }
}
public static ArrayList< ArrayList <Edge >> graph = new
                                        ArrayList<>();

public static void construction()
{
        int n=7;
        for( int i=0; i<n; i++)
        {       graph.add( new ArrayList<>() );
        }addEdge (0,1,10);   addEdge 2,3,40)  addEdge(4,5,2)
                 (1,2,10)        (3,0,10)        (4,6,3)
}                                (3,4,2).

public static void addEdge. (int u, int v, int w)
{
    if( u<0 || v<0 || u>= graph.size() || v >= graph.size())
    {       return;
    }

        graph.get(u).add( new. Edge ( v,w) );
        graph.get(v).add( new edge.(u, w));

}
```

```
public static void display()
{
    for(int i=0; i<graph.size(); i++)
    {
        sopo(i + " -> ");
        for(Edge e : graph.get(i))
        {
            sopo("(" + e.u + ", " + e.v + ")");
        }
        sopo();
    }
}

public static void main(String[] args)
{
    construction();
    removeEdge(3,4);
}

public static void removeEdge(int u, int v)
{
    int i=0;
    int j=0;
    while(i < graph.get(u).size())
    {
        Edge e = graph.get(u).get(i);
        if(e.v == v) { break; }
        i++;
    }
```

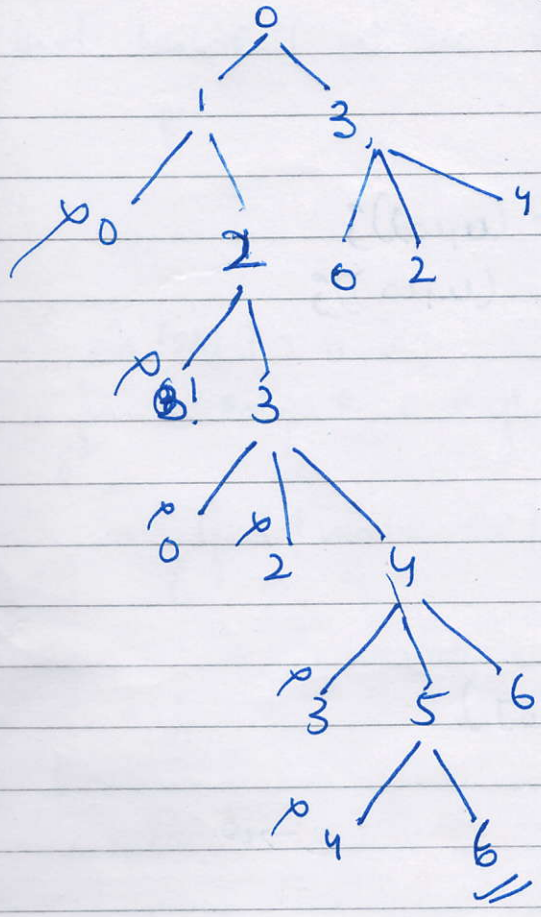


```
while ( j < graph.get(v).size() )
{
    Edge e = graph.get(v).get(j);
    if(e.v == u)  break;
      j++;
}
    graph.get(u).remove(v);
    graph.get(v).remove(j);
}
public static void removeVtx(int u)
{
    while ( graph.get(u).size() != 0 )
    {
        int v = graph.get(u).
        int lidx = graph.get(u).size() - 1;
        int v = graph.get(u).get(lidx).v;
        removeEdge(u,v);
    }
}
```

13th January 2020.



if only one path is to be discovered we need not make it false again.



① src == dst
return

② vis = true

③ for all unvisited neighbours.

④ dfs. (nbr);

```
#include <iostream>
#include
using.        ;

class Edge
{     public:
        int u=0;
        int w=0;
      Edge ( int u, int w)
      {  this->u = u;
         this->w = w;
      }
};
vector <vector <Edge *>> graph (n, vector <Edge *>());
```

void addedge.

graph[u].push_back(new Edge(u,v));
graph[v].push_back(new Edge(u,w));

display.

for(.

for(Edge* e : graph[i]).

cout e->v e->w.

~~arr.beg~~

arr.erase(arr.begin() + 3);

graph[u].erase(graph[u].begin() + i);
graph[v].erase(graph[v].begin() + j.);

void removeVtx.

while(graph[u].size() != 0)

{

Edge* e = graph[u][graph[u].size() - 1];

removeEdge(u, e->v)

}

1 1 1 3 3 3 4 4 5 5 6 6 v [0] [1] [2]

bool hasPath (int src, int dest, vector<bool> & vis, string psf)
{

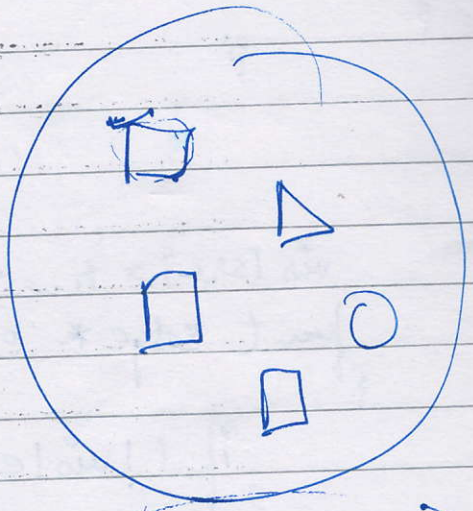
vis [src] = true;
for (Edge * e : graph[src])
{
if (!vis[e->v])
{
hasPath (e->v, dest, vis, psf + to_string (e->v)
}
}
// vis [src] = false for all path
}

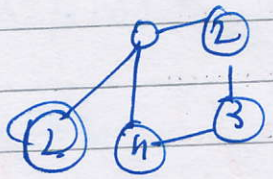
Q. smallest path in terms of weight.] connected path
Q. largest path
Q. Total No. of connected graphs.] No. of islands. the same

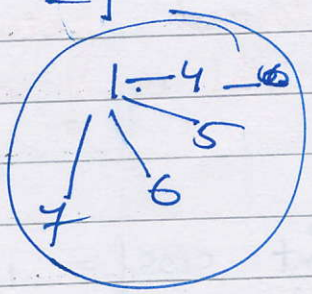
int smsf = 1e8; string spsf = "";
int lmsf = -1; string lpsf = "";

if (src == dest)
{
if (wsf < smsf)
{
smsf = wsf;
spsf = psf;
}
if (wsf > lmsf)
lmsf = wsf;
lpsf = psf;

```
void getconnectedcomponents()
{
    vector<bool> vis(n, false);
    int comp = 0;

    for(int i=0; i<n; i++)
    {
        if(!vis[i])
        {
            comp++;
            dfs(i, vis);
        }
    }
    cout << comp << endl;
}

void dfs(int src, vector<bool> &vis)
{
    vis[src] = true;
    for(Edge *e : graph[src])
    {
        if(!vis[e->v])
        {
            dfs(e->v, vis);
        }
    }
}
```



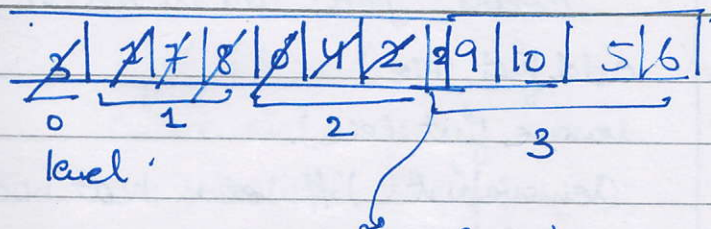




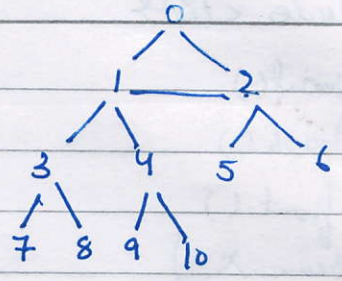
BFS.

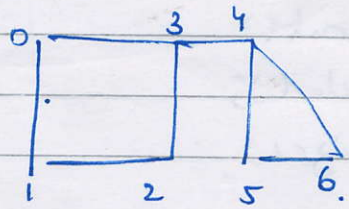
14 July Sunday 20

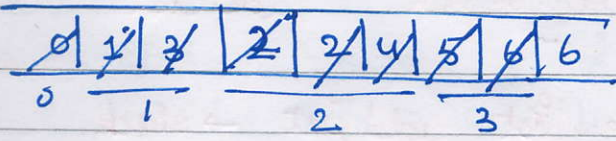
BFS is a good algo to find cycle wrt to a given source.



as 2 is coming twice it is implying that there is cycle.







∴ No. of edges to reach 6 from 0 is 3.

If any no which is already visited is encountered in graph we will say here is a cycle.

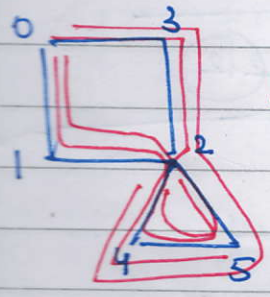
0 source ke respect mein we are always pushing unvisited element in queue. we are getting 2 cycle.

Shortest path in terms of length.

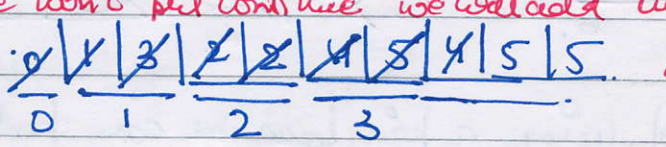
add continue in order to avoid this

```
for cycle & destination.
if (vis[ele]) cycle++;
                continue;
```

1. queue of int, vis
2. queue ← src.
3. while (queue.size() > 0)
   - 3.1 remove first of queue.
   - 3.2. vis ← ele.
   - 3.3 For all unvisited nbr's
     - 3.3.1 que ← nbr's



If we won't put continue we will add unvisited neighbours which will increment cycle count.



① ② ③ ④

If we won't put continue in bfs it will give all different paths to traverse in cycle.

Here here are 4 ways | 0 ke respect mein sabhi edges ko traverse kiya he aaya.

queue <int> que = new LinkedList <>(); Java.

C++

```
#include <list>
push_front()
push_back()
pop_front()
(return x)
front()
back();
size()
```

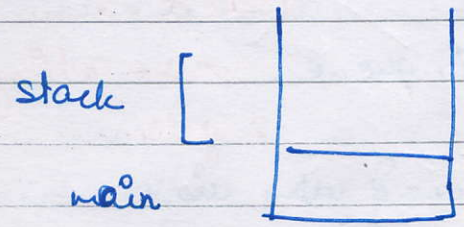
Java

```
import java.util.LinkedList;
addFirst(ele)
removeFirst(ele).
removeFirst(); // return that node.
getFirst()
getLast()
.size()
```

removeFirst addLast to make queue.

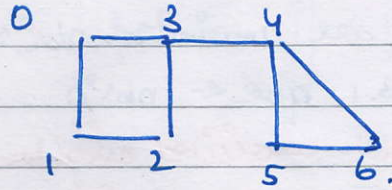
removeFirst addFirst → stack.

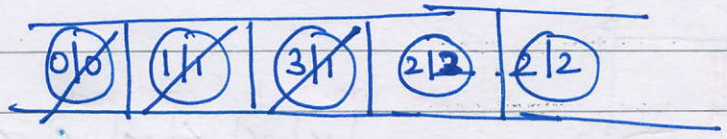
* Array de stack banta hai queue nhi banta.



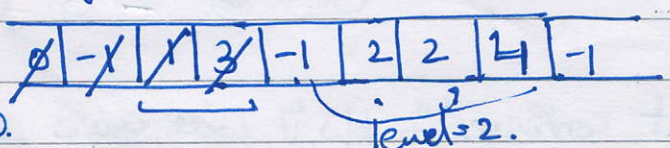
Importing library file in program leads to firing of copy constructor. which is costly.

Heap tak pahunchne mein aur stack tak pahunchne mein stack is more efficient. ∴ if we make array on stack - it is efficient.





Instead of using a pair class we can put null

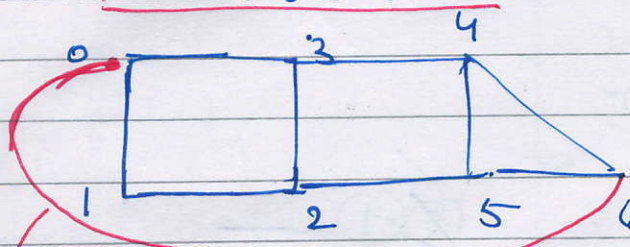


```
void bfs( int src)
{       queue<int> que;
        vector<int> vis(n, false);

        que.push( src);
        que.push(-1);
        int cycle = 0;

        while( que.size() != 1).
{       int curr = que.front();
            que.pop();
        if( curr == dest){ cout<<level. }.
        if( vis[curr]) { cout << "cycle" << cycle++ <<endl;
                         continue; }.

        vis[curr] = true;
        for( Edge* e : graph[curr])
{           if( ! vis[e->v])
            {   que.push(e->v);
            }
        }

    if( que.front() == -1)
    {       level++;
            que.pop();
            que.push(-1);

    }

}
```

18th January

Total no. of vtx − 1 — If we need minimum no. of edges to connect graph without creating cycle.

Hamiltonian Path

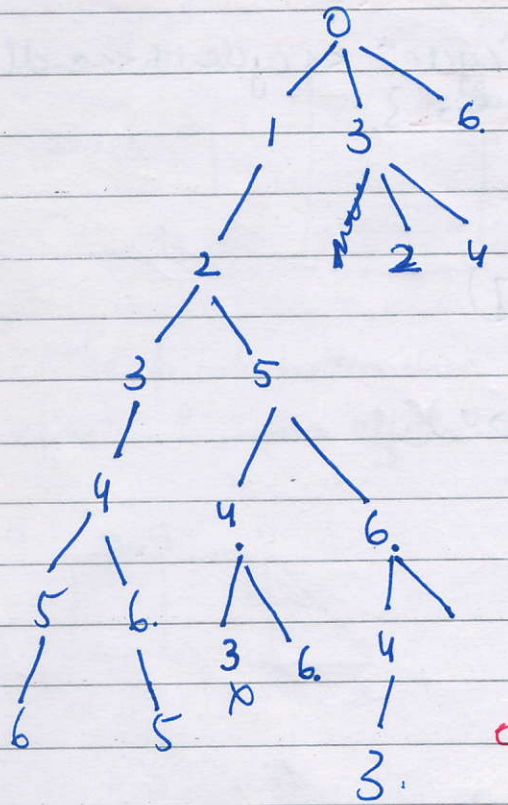


visiting each vertices exactly once.

0 1 2 3 4 5 6
0 1 2 3 4 6 5
2 1 0 3 4 5 6
2 1 0 3 4 6 5

Hamiltonian path & Hamiltonian cycle.

If there is a direct edge between source & destination after all edges are traversed is called hamiltonian cycle.



base case when total no. of count = one less than size(v) − 1.

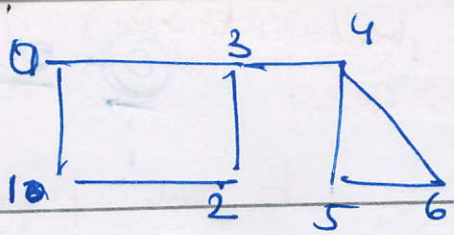
```
int hamiltonianpath( int vtx, vector<bool>&vis)
                     int vtxcount, int osource
{ (1) if( vtxcount == graph.size()-1)
  { (2) for( Edge e: graph[vtx])
        { (3) if( e->v == osource)
              cycle.
          int count = 0;
          vis[vtx] = true;
          for( Edge *e : graph[vtx]) flag = true;
          { if(!vis[e->v])
            {
              count += hamiltonianpath(e->v, vis,
                       ans + to_string(vtx)
                       vtxcount+1);
                       osource.
            }
          }
  }
  vis[vtx] = false;
}
```

(2.2)

```
if(!flag)
{
  cout << ans + to_string(vtx) << endl;
}
return 1;
}
```

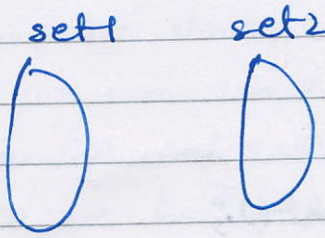
path: 0 1 2 3 4 6 5

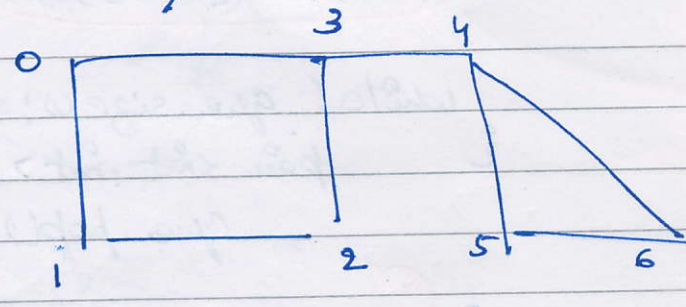
cycle:
0 1 2 3 4 5 6
0 1 2 5 6 4 3
0 3 4 6 5 2 1
0 6 5 4 3 2 1

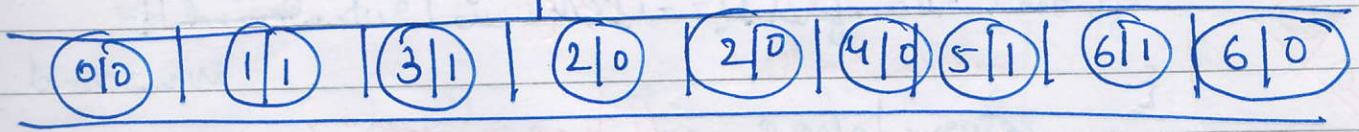


Bipartite.

Graph is to be divided in two sets such that No two adjacent vertex are in same set.







queue.

add 0 with 0.

unvisited neighbour $(0+1)\%2$

val associated with node update in vis array.

```
0 → -1
1 → -1
2 → -1
3 → -1
4 → -1
5 → -1
6 → -1
```

1. vis<int>. default -1
2. que ← (src|0)
3. while(!que.size())
   - 3.1 remove vtx
   - 3.2 check for contradiction.
   - 3.3 marks remove vtx to vis
   - 3.4 for all unvisited vtx
   - 3.4.1 que ← nbr.

* graph may be of type

for this que is to be done first

```
bool isBipartite(){

    vector<int> vis(graph.size()-1);
    queue<pair<int,int>> que;

    for(int i=0; i<graph.size(); i++)
    {
        if(!vis[i])
        {
```

```
bool bipartitehelper( int i ,vector<int> &vis)
    que<pair<int,int>> que;
    bool flag = true;
    que.push({i,0});

    while( que.size()!=0)
    {   pair<int,int> curr = que.front();
        que.pop();

        if( vis[ curr.first]!= -1)
        {   cout << "conflict"
            return false;
            continue;
        }
        vis[ curr.first] = curr.second;

        for( edge* e : graph[curr.first])
        {   if( !vis[e->v])
            {   que.push( {e->v , (curr.second+1)%2} );
            }
        }
        //cout << count << " " << (boolalpha)<< flag <<endl;
    }
    return flag;
}
void bipartite ()
    vector<int> vis (graph.size()-1) ;

{   for(int i=0; i< graph.size(); i++ )
    {   if( vis[i] == -1)
        {   cout << count << (boolalpha)<< bipartitehelper(
            count++;
        }
```

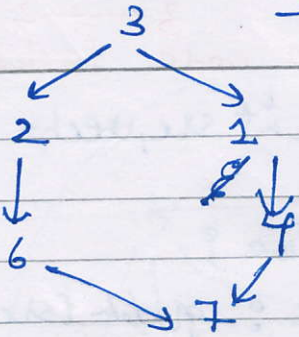
Directional Graph.

Leetcode

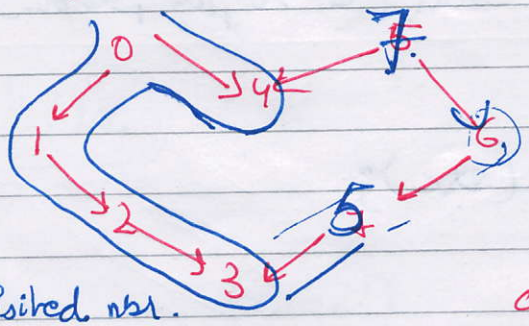
Course schedule

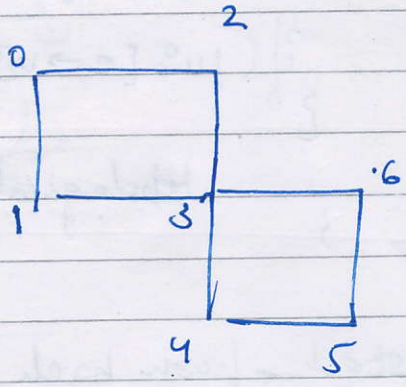
Course schedule II

If we want to study courses but we can read it when the dependent courses are being read.



here we will make stack using array





DFS

1. vis
2. for del unvisited nbr.
   add nbr

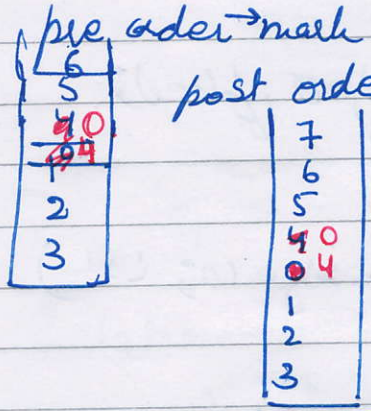
u → v jaana chahte ho ie we should always get forward edge.

0 –
1 –
2 –
3 –
4 –
5 –
6 –
7 –

pre order → mark time

post order → add in stack

put in stack

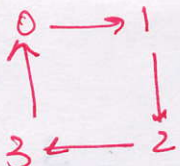


Reverse order of stack

Ans 7 6 5 0 4 1 2 3.

1. mark, src.
2. for all unvisited nbr
   DFS (nbr);
3. ans.stack.push(src)

(src, vis, ansstack)



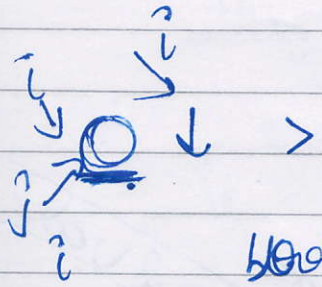
This code lacks ability to detect cycle.

```
void topologicalSort(int src, vector<bool>& vis, vector<int>& stack)
{
    vis[src] = true;
    for(Edge *e : graph[src])
    {
        if(!vis[e->v])
        {
            topological_sort(e->v, vis, stack);
        }
    }
    stack.push_back(src);
}

void topological()
{
    vector<bool> vis(graph.size(), false);
    vector<int> stack;

    for(int i=0; i<graph.size(); i++)
    {
        if(!vis[i])
        {
            topological sort(i, vis, stack);
        }
    }

    while(!st.isEmpty())

    while(st.size() != 0)
    {
        cout << stack.back() <<
        stack.pop_back();
    }
}
```

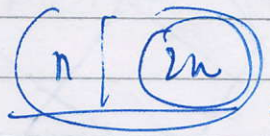
.bool()
.beach()

Namdeo

Indian classical.



bool flag



S1 $n\log n$

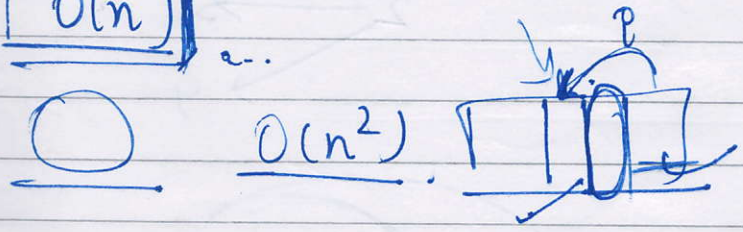
S2 $O(n^2)$

S3

S4 $S_1 <= S_2 >= S_3 <= S_4$

O(n)

Relationship.



$O(n^2)$

$S_1 <= S_2 >= S_3 <= S_4$.