YAGYA MODI

GENESIS

modiyagya@gmail.com

HW How do vector expand & all Dynamic allocation.

9-04-2020

Array questions solving technique

⊙ int arr[5]

→ Homogenous distribution
→ static size is fixed
→ contigous memory allocation

Dynamic Arrays ⇒ vector
(in java: array list)
size grows as & when needed

⊙ void func(int arr[], int N)
int *p
{

}

input is passed in 2 ways—
1. pass by value
2. pass by reference → just pass the address. Quick also.
(bcoz new array isn't created.
unless in pass by val.)

main()
{

func(arr, N)
↳ base address → &arr[0]

Q1 Given Arr[N] arr[i] $\leq 10^5$

int p, q, r $\leq 10^5$

Maximise: $p*arr[i] + q*arr[j] + r*arr[k]$, $i<j<k$

Sample Input

Arr: { 1 2 3 4 5 }

p: 1 q: 2 r = 3

SO: - 26

$1\times3 + 2\times4 + 3\times5 = 26$.

→ p, q, r can be negative; arr[i] can be -ve too.

$i<j<k$ → IMP thing.

Algo →

M1 Brute force.

3 ~~loops~~ Nested loops

```
int ans = INT_MIN;   //initialize most min value.
for(i=0; i<n; ++i){
    for(j=i+1; j<n; ++j){
        for(k=j+1; k<n; ++k){

}
}
}
```

i=0, j=1

Time complexity: (N-2) + ... - - -

$O(N^3)$ - note exactly n^3. little less than that.

~~XX~~ ① p·a[i] $\leq 10^{18}$

can't store it in int.

Take long / long long int

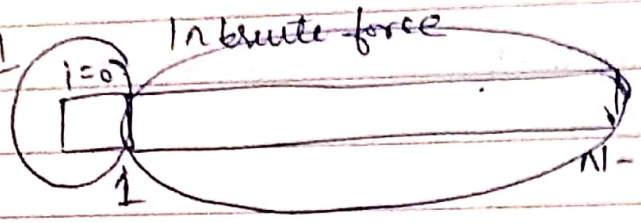
NX2 Sort the array
indexes change

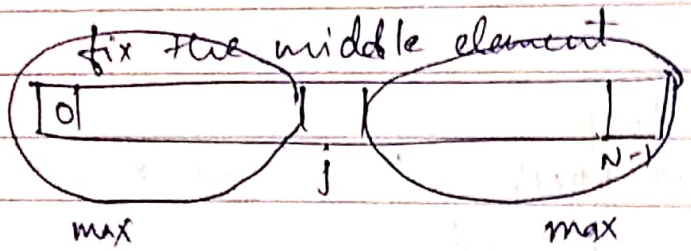
Q1 a) Max arr[i]·1 arr[j]·1 arr[k] , i<j<k

↳ Ago choose max 3 elements, no matter what their indexes are

[7 10 5]
arr[i] arr[j] arr[k]

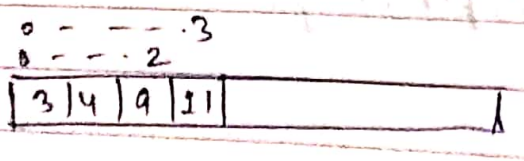
M In brute force





Max[0 - - j-1]
prefixMax[j-1]

Max[j+1 - - - N-1]
suffixmax[j+1]

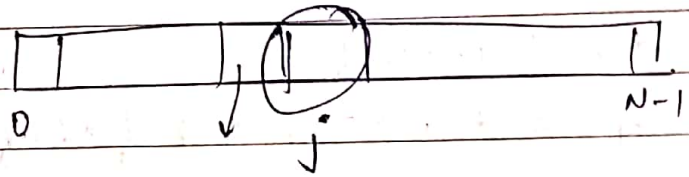


prefixMax[2] = 9
prefixmax[3] = max(prefixMax[2], arr[3])

prefMax[j] = max(prefMax[j-1], arr[j])
int prefMax[N];
prefMax[0] = arr[0];
for(i=1; i<n; i++)
prefMax[i] = max(prefMax[i-1], arr[i]);
Time: O(n)

Similarly, calculate suffixMax[]..

```
int suffixMax[N];
SM[n-1] = arr[n-1];
for(j = n-2; j >= 0; --j)
    suffixMax[j] = Max[SM[j+1], arr[j]);
```



Corner cases: j = 0, N-1

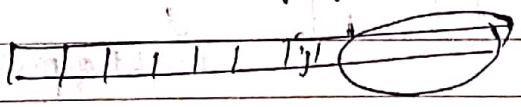
```
for(j = 1 to N-2)
{
    x = prefMax[j-1] + suffMax[j+1] + a[j];
    ans = max(x, ans);
}
```

Time: O(n)

Space: O2n

We can reduce time complexity by-

initialise maxm var for prefixMax. keep suffixMax[]
No need of pMax[]



When i'm at j, i only need max till 0 - j-1, that can be stored in maxm var. = max(arr[j-1], max)

for q1 (original)

M

In Prefix array store max (p*arr[i])

Suffix max (r*arr[i])

prefixMax[j-1] = max(p·a[0], p·a[1], ... p·a[j-1])

→ p, q, r were given just to confuse you.

Actual concept was to pre compute prefix & suffix.

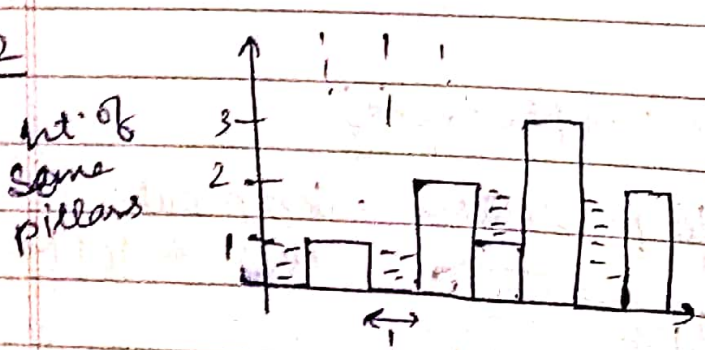
HW → Here constraint was i<j<k.

if constraint was i≤j≤k.

Change: same element can repeat also.

Compute for it

Q2



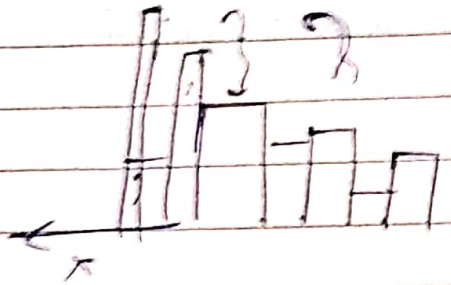
Water dropped from above

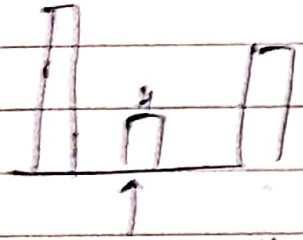
Ans = {0, 1, 0, 2, 1, 3, 0, 2}

↓

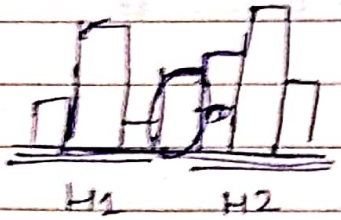
height of pillars.

If there's a pillar at to the right of it there's no pillar whose height is greater than this. No water will get trapped over this. Why? Because





Water will get trapped over this only if there's a pillar in left whose $ht_1 > H$ & there's atleast a pillar in right also, whose $ht_2 > H$.



find $\max(H1)$. if $H1_{max} < H$ || $H2_{max} < h$; Not today possible over this.

water filled above this is $\min(H1, H2) - H$

Common sense

1. When will the answer dont exist. If you figure out this, you can
2. automatically get when will the ans exist.

Q. Arr[N] = { } p
all are unique elements
permutation of {0, 1, ... N-1}

What is max no. of chunks, sorting which will sort the entire array

chunk - any contigous segment

→ split into multiple chunk
- Sort chunk
- Concatenate the chunk in proper order

S1: Arr[5] = [1, 0, 2, 3, 4]

[1, 0, 2] [3, 4]
0, 1, 2 3, 4 chunk = 2
I want max. no. of chunks.

[1, 0] [2] [3] [4]
c1 c2 c3 c4 4 chunks.

if asked min. no. of chunk = 1 take complete arr, sort it

S2 [1 2 0 4 3 5]
all elements of the form [i...j]p max = 3 chunks

[1 2 0 4 3 5]

if (idx == max) then ans++;

max: 1 2 2 4 4 5

idx=3 idx=4 idx=5

idx==max

ans++;

ans++ ans++ ... 5 chunks.

→ What if array can be anything

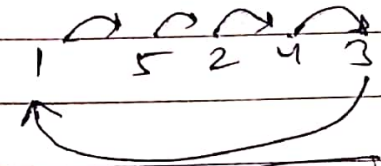
Q4 Arr[N]: 1 5 2 4 3 $\xrightarrow{+3}$ 2 4 3 1 5

int k >= 0

Right rotate the array by k units

If by 1 unit.

k = 1



3 1 5 2 4 → ~~O(n)~~ O(n)

→ T(O(n*k)) <= n^2

k = k % n for k > n.

no need to rotate again

M2 Reduce time, may use extra space.

Arr: 1 5 2 4 3

Tmp: 2 4 3 1 5

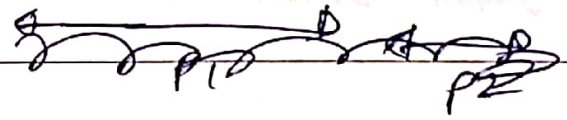
arr[i] → Tmp[(i+k)%n]

M3 Remember

Time O(n)

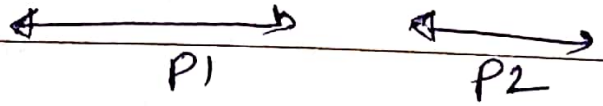
Space O(1)

3 4 2 5 1



Reverse array

1 5 2 4 3



1 5

Q5 Arr[N]

1 5 2 4 3

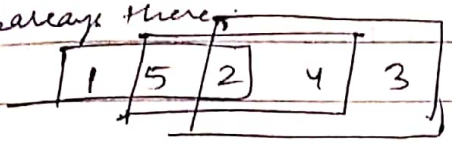
print the sum of every k sized subarray.

↳ contigous

subsequense → Need not be contigous.

Eg

→ 3 subarrays there.



(N-k+1) subarrays. index ← 0 to (N-k)

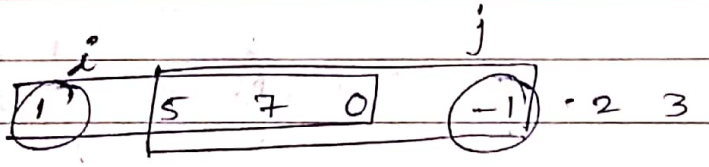
↳ (N-k)+(k-1)=N-1

M1 rude

calculate for every subarray calculate it. O(k) for each subarray

O((N-k+1)k)

M2



let go of one element, & you welcome another element

S + ans of window 1., Next window ans can be calculated in just O(1) time

s - a[i] + a[j]

Sliding window can give you answer only if next window can be.

→ print max element for every subarray of k size.

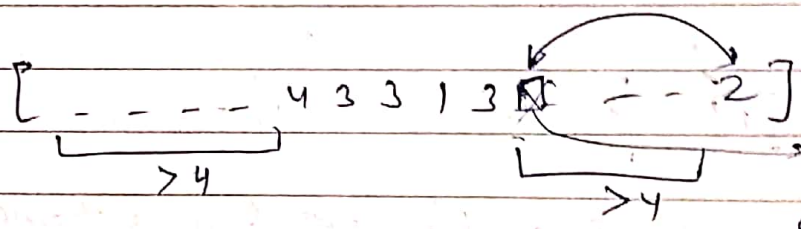
Q int arr[N]
int k;

What is the min no. of swaps reqd to bring all the elements $\leq k$ together.

Arr: { 2 1 5 6 3 }

2 4 5

First find total ele $< k$ say x.
then find sliding window of size x with max ele $\leq k$

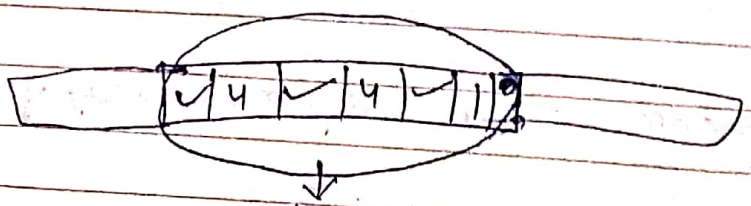


It's better to swap 2 with 4 instead of rather to swap all other element to near 2.

count = CNT of elem $\leq k$

Check all subarrays of size count, that have max legal elements
↳ elements $<=k$.

let k=4 count = 6 for it



here (let this is that sliding window
I wont want to swap 4,4,) 1
I would want to swap) with other

Legal elements out there - (Not swap 4, 4, 1 among themselves)

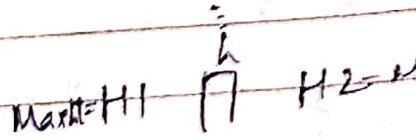
here l.c. = 1 (legal count = l.c.)

for each slide of window
- if leaving element is legal count → l.c --
- if coming element is legal count → l.c ++
  - if a[j] <= k. (if (a[i] <= k)) ↳ O(1)

Total : (N - k) more windows.

⇒ So O(n) time.

87 Trapping rain water



(H1)>=h<=(H2)=>min(H1, H2) - h >= 0)

Amul
Shivani

1. perseverence
1. Do not accumulate a lot of backlog
Midsems, college work, bhai marriage
Unless you finish things on time. (otherwise ulti doubt will come)
Tue, wed, thursday → take responsibility that finish it off in this time

Come what may →

2. Do not skip classes at all.
~~Ability~~ Attend live classes as much as possible.
Put it in your brain — there are no recordings.

3. Make Notes
After every class, make good notes.

4. Whenever you have issues, reach out emotionally, physically, or

5. What do
1. Consistent Hardwork
2. Confidence in yourself
3. Conviction in yourself (what do you feel about yourself.

6. Minimise your screen time
take good care of yourself. Yoga. keep your energy levels high.

Q4 Arr[n] - positive integers
Find min. no. of swaps required to bring ∀ no. ≤ k. together.

2 1 5 6 3, k = 3

2 7 9 5 8 7 4, k = 6

2 swap

~~Q~~
Q5 1 2 3 4 5 6 7, k = 3
P1 P2 rotate right by k units.

O/P: 5 6 7 1 2 3 4

Reverse

7 6 5 4 3 2 1

P2rev P1rev.

5 6 7 1 2 3 4 = O/P.

(P2rev)rev (P1rev)rev

= P2

6-2-1

0 1 2 3 4 5 k = 2.

5 4 3 2 1 0

20-04-2020 Techinique

1. Sliding window
2. prefix calculations
3. How to handle reverse/rotation in O(n) time without any extra space

Q Given arr[N] = 1 3 6 9 14

Find max difference b/w any 2 consecutive elements in the sorted form of array

$arr_i \leq 10^{18}$, $n \leq 10^6$

SO: 3

1 3 6 9
2 3 3

M Brute

Sort the array & check it

Time: nlogn

M2 : I do not want to sort the array

1. Max element = 9
2. Declare array of size = 10

B: | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
0 1 2 3 4 5 6 7 8 9

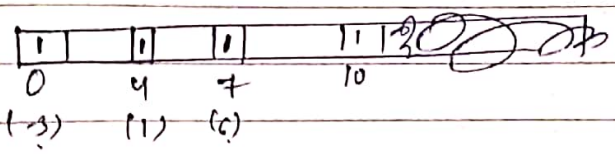
B[a[i]] = 1;

3. traverse B array. max($idx_2 - idx_1$) for B[idx_1] = 1 & B[idx_2] = 1

Cons:

✗ Negative Number ? No. Geeks (let say -3 there) Store most Negative No. (-3 here) at idx=0, & shift all other element (let 9) to idx=9+x;

{ -3 6 9 1 }



★ Time: ~~O(n)~~ ? XX

Space Comp.: O(Max - Min)

↳ Range of element ⇒ Cons: if Range ∈ [0, 10^8] Huge.

Time: O(max-min)

→ ~~Not~~ good approach when elements are not very diverse.

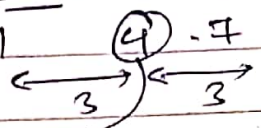
M3

Technique Bucketing

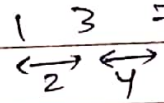
Q Given Arr[N], N=3, min=1, max=7

SI: { 1 _ 7 } What should be the middle element, s.t (Max. consecutive ~~possible~~ gap) between ~~consecutive~~ elements is Minimum possible.

SO: 4 → Middle element of 1 & 7

→ 

1) if Output < 4, let 3.



⇒ ~~Not~~ Max consecutive gap = 4

2) Same for output < 4

~~if not~~

→ Choice for middle element s.t max gap b/w consecutive elements is min. possible will be s.t. distance from ends will be equal.

SI2 N = 6, min = 10, max = 20, tell atleast 4 elements for per question

10 12 15 18 20

10 12 14 16 18 20
+2 +2 +2 +2 +2

I know for sure what can be the min possible value for any.

Min gap = $\frac{\text{Max-min}}{(N-1)}$

10 ---- 20
(N-1) slots

→ If this isn't exactly divisible, then ceil of this =?

$\frac{11}{5} = 2+3$

10 12 14 16 18 21
+2 +2 +2 +3

Ans = $\frac{\text{Max-min}}{n-1}$ + (max-min) % (n-1) != 0 ? 1 : 0

→ HW ceil explore.

★ How is this going to help us.?

Anything b/w [min ... (min+gap-1)] will never be

Eg 10 20 30 40 50 60 70 , N=6.

gap = $\frac{70-20}{6-1} = 10$

This is like a bucket responsible for my ans. s.t. no 2 elements from this bucket will ever be responsible for my ans

[20, 21)

[20 - - - 29]

If I pick any 2 elements b/w 20 to 29, Max possible gap = 9 < Max gap (10)

So any 2 elements from [20, - - 29] is useless to my ans since my ans >= 10 (Max gap).

B0: [min - - - (min + gap - 1)]

B1: [min + gap - - - - - (min + 2*gap - 1)]

B2: [min + 2*gap - 1, - - - - (min + 3*gap - 1)]

B3: [- - - - -]

Diff b/w Min of B1 & Max of B0 , i.e.

Min of B(i+1) & Max of B(i) is of my concern

→ (20) 25 27 33 65 (70) , k = 6

min Max

Initialise bucket		
Min	Max	Buckets
+∞	-∞	[20 - - - 29] 0
+∞	-∞	[30 - - - 39] 1
+∞	-∞	[40 - - - 49] 2
+∞	-∞	[50 - - - 59] 3
+∞	-∞	[60 - - - 69] 4
		[70 - - - 79] 5

Bucket will be → O(N) of size (N-1)/gaps.

For every bucket I'm tracking: min & maximum.

57 → which bucket?

$\frac{57 - min}{Max\ gap} = \frac{57 - 20}{10} = 3$ → bucket

$$bucket\_no = \frac{arr[i] - min}{gap}$$

20 23 25

20 → which bucket does 20 belong? Bucket 0.
what it changes Min (+∞) or Max (-∞)? Min & Max.

	Min	Max
0	+∞	-∞ → 20 → 20 → 20 → 23 → 20 → 25
1	+∞	-∞
2	+∞	-∞
3	+∞	-∞
4	-∞	-∞

Same for 23: Which bucket? 0
Change Min? No.
" Max? Yes. 23

Code:
N = No. of gaps. Calculate gap. O(1) time
Iterate over array. that which bucket element belongs to.
MIN[N]
MAX[N]
Once you get this you can easily update min[x], max[x] of x bucket.

Time complexity: O(n)

Warning: Aware
1) if gap==0, pls dont use this formula.
(Eg: if Max = Min → all elements same)
2) $gap = \frac{max-min}{n-1} \Rightarrow n \geq 2$

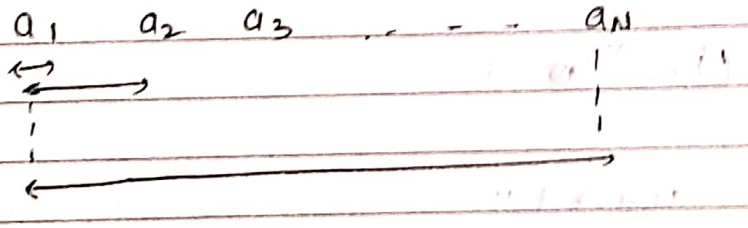
3) In bucket if min = +∞, then max can never be +∞. (will always be +∞)
(can not be anything like 20, 30 etc)

4) It can never happen that only 1 bucket is non-empty & all other buckets are empty.

Q. ~~a0 a1~~ int Arr[N]

{1, 2, 3}

sum of all subarray



No of subarrays starting from a_1 = N

starting from a_2 = N-1

= N + (N-1) + . . . 1

Total = $\frac{N(N+1)}{2}$ subarrays

~~2 + (-1) + (3) +~~

2 + 1 + 4 + -1 + 2 + 3

= 11

(1) Ulta socho

Kitne arrays me a_i aayega.

a_i will come $(i+1) \times (n-1)$

$\frac{n(n+1)}{2}$ $\frac{n(n-1)}{2}$

Total

You do not do what is being asked for. You start reverse

start [0 - - - i] = (i+1) elements

ending idx [i - - - N-1] = (N-1) elements

total subarrays in which a_i will come = (i+1)*(N-1)

Contribution of a_i in ans is ans += (i+1)*(n-1)*a_i

$$(a+b)\%m = (a\%m + b\%m)\%m$$

$(i+1)^*(n-i)$

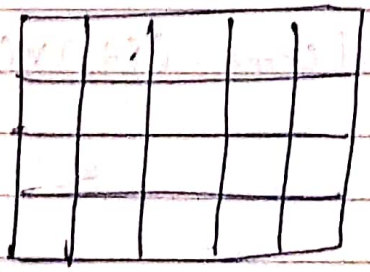
4 + 8 + 4

Q

n^2

M brute

any cell can act as top left of submatrix.

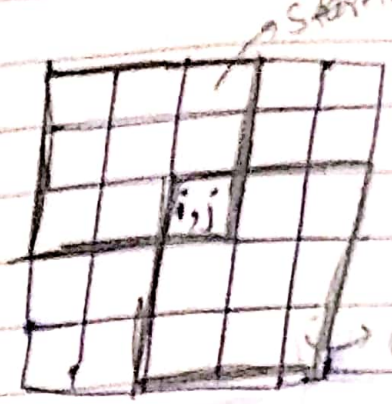


```
i1 ← 0 to N-1
  j1 ← 0 to N-1
    i2 ← i1 to N-1
      j2 ← j1 to N-1
        i ← i1 to i2
        j ← j1 to j2
```

Time = O(n^6)

3x3 x 3x1

Top left and Right bottom needed to locate a submatrix. For any element a_{ij}



chairs = $(i+1)*(j+1)$ elements

chairs for $(N-i)*(N-j)$

$$ans += a_{ij} * (i+1)*(j+1)*(N-i)*(N-j)$$

1) Overflow chances very HEAVY Modulo ing

Q Given 2D ~~array~~ Matrix of mxn.

'Q' queries given. [TL, BR]

Every query denotes top left & bottom right of submatrix.

[i1, j1, i2, j2]

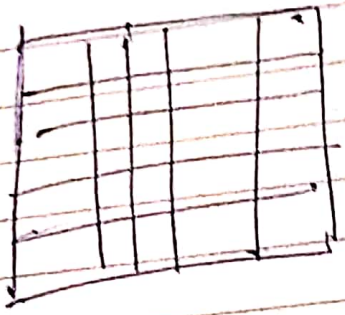
O/p:- ~~sum of that matrix~~ qi query.

SI:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 2 & 0 & 1 \end{bmatrix}$$

Q = [0, 1, 1, 2]

O/p = 16



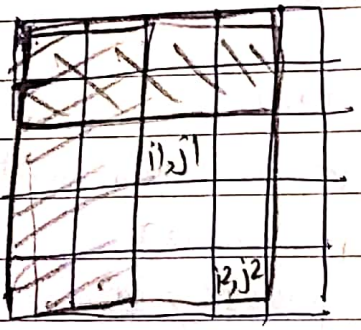
M2 precompute prefix sum.

arr [1 3 -1 5 2]

(prefixsum) PS: 0 1 4 3 8 14

Q: [i1, j1]

Time = O(n + q) ← O(1) time to replace



$$Sum = Arr_{i2,j2} - Arr_{i1-1,j2} - Arr_{i1,j2-1} + Arr_{i1-1,j1-1}$$

Calculate prefix sum as

a	b	c
d	e	f
g	h	i

Psum row →

a	a+b	a+b+c
d	d+e	d+e+f
g	g+h	g+h+i

PS col. ↓

a	a+b	a+b+c
a+d	a+b+d+e	a+b+c+d+e+f
a+d+g		

Warning: When i1, j2 - 1 go out of index error, corner cases.

Approach: Take sum of all submatrices.

M3 precompute prefixSum

↳ $O(n^4)$ → for 4 loops.

Innermost 2 loops, no need.

(We have better approach $O(n^2)$)

Q Given a matrix $M \times N$

→ every row is sorted

→ " col is sorted.

e.g.

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

int target = 29

M1 binary search on m logn.
for m rows.

M2

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

start

$a_{03} > 29$ ←

$a_{02} > 29$ ←

$a_{01} < 29$ ↓

$a_{11} < 29$ ↓

$a_{21} == 29$. I found it

Termination when ($i < 0$ || $j > n-1$)

Time: $O(m+n)$

I can start from either Top Right corner or Bottom left corner.

(Cannot start from left top & or Right bottom)

Q Permutation in string

M1

If there is permutation of one string into S2.
just sort the 2 strings & compare if they're equal.

// strings S1, S2 => same length
// S1 is a palindrome of S2
// Anagrams.

// S1 = "abcdba"
// S2 = "cda abb"

TC: O(nlogn)
SC: O(1)

M2 Hashing - Storing frequency

TC: O(N) + O(K)
↳ no. of unique characters (max k = 26)

SC: O(K)

// freq = {a:-1, b:2, c:0, d:0}
cnt = 2

↳ using this for original question

Time →

Code {

M1

sort(s1.begin(), s1.end()); // nlogn
int n = s1.size(), m = s2.size();
for(int i=0; i<=m-n; i++) { // m
string tem = s2.substr(i, n);
sort(temp.begin(), temp.end()); // nlogn
if (temp == s1) return true
} // mnlogn
return false;
}

M2 ⊖ frequency approach
O(mn)

M3 sliding window approach

// "caa"
// {'c':1, a:2}, b:0}
// "aba cabab"
// {a:2, b:1, c:0}

ASCII value	'a' = 97
	'A' = 65

```
{
    int freq1[30];
    int freq2[30];
    int n = s1.size(), m = s2.size();
    if (n>m) return false;
    for(int i=0; i<26;++i)   freq1[i] = freq2[i] = 0;
    for(int i=0; i< n; i++) {
        freq1[ s1[i]-'a'] ++;
    }
    for (int i=0 ; i<m; i++) {
        freq2[ s2[i]-'a']++;
    }
    int i=0;
    for(i=0; i<26; i++) {
        if (freq1[i] != freq2[i]) break;
    }
    if (i==26)   return true;
```

```
for(int i=n; i<m; i++){
    freq[s2[i-n] - 'a']--;
    freq[s2[i] - 'a']++;
    int j=0;
    for(j=0; j<26; ++j){
        if(freq1[j] != freq2[j]) break;
    }
    if(j==26) return true;
}
return false
}
```

HW MY Divya optimisation

Q Always better to ~~text~~ replace last character of matched substring in S1.

Eg // sirsirsirsi
// sirs

↳ // #irsirsirsi (First letter #tag)
↳ this matches again

~~// ##sirsirsirsi~~

↳ sirsir#sirsi (last letter #tag)
↳ no match No.

TC: O(m+n)

HW string.find() function to compare.

Note → whenever you have ~~prefix sum~~ subarrays sum=k, ⊕
JUST THINK PREFIXSUM.
→ product from l to r = product till r / product till (l-1)
- used for XOR (l--r)

25 April Lecture 4

idx:	0	1	2	3
arr:	1	3	0	2
tmp:	2	0	3	1

M1

```
int tmp[N];
for i=0 to N-1{
   tmp[arr[i]] = i;
}
```

copy tmp to arr

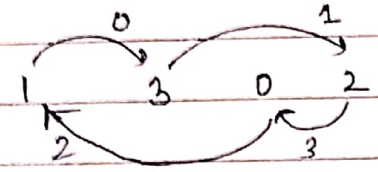
TC: O(n)

SC: O(n)

M2 Reduce SC.

~~X~~ update directly

~~as~~ Since value lost.



2	0	3	1

before overriding value.
you should save the
value in tmp. then
~~further~~ place the ~~value~~ of tmp idx
at its correct value

→ Problem: Disconnected graph

Sol^n: Maintain bool vector

bool ~~arr~~ visited[N]

Eg 2 0 1 3 5 4 → 2 parts.

Cons: SC still O(N) then?

~~but~~ M1 is much simpler
than

M3 when you're doing

~~arr~~[arr[i]] = i

We can instead change it to

arr[arr[i]] = -i

So if (val < 0) we won't alter that

Problem: -(0) = 0. How will
we know for idx = 0?

Modify: So replace it with

arr[arr[i]] = -1*(i+1)

(or we can replace it with

arr[arr[i]] = ~~o~~ i+N.

~~for i=~~

CODE

```
for(i=0; i<N; i++)
{
   if(arr[i] < 0) continue;
   else{
      int idx = arr[i], val = i;
      while(idx != i)
      {
         int tmp = arr[idx];
         arr[idx] = -1*(val+1)
         val = idx;
         idx = tmp;
      }
      arr[idx] = ~~val~~ -1*(val+1);
   }
}
```

TC: O(n)

SC: O(1)

M

i=6

[0, 2]

I want to store both new value & old value

New-val old-val

transformation N* New val + old_val

x%N = ? old_val

x/N = New_val

- Can we use this-

N * old_val + New_val ?

XX NO

Because let say.

idx = 2 n=7

arr[idx]=3

& this is not being visited.

Then I use

n* old + new formula

My old ~~val~~ val = $\frac{x}{7} = \frac{3}{7} = 0$

Not true.

So.

Cons: 10^{22} overflow situation of int need it to store it.

TC: O(n) SC: O(1)

Pro: Unutilised bits of array are being utilised.

32 bit integer was storing small value 0 0 0 3 0

Now some of those unutilised value are being used.

→ Because of this method value overflow. $10^6 * 10^6 = 10^{12}$

previous method is better than this.

→ But this method is very very useful at some other places.

Q Compute GCD (Greatest common divisor)

gcd ≤ min(a, b)

M1

6 = 1 2 3 6

14 = 1 2 7 14

X X

M2

a b

7) 25 (3

21

4) 7 (1

4

3) 4 (1

3

1) 3 (3

3

0

gcd.

rem ≤ b/2

Case 1 a > b/2

Case 2 a ≥ b/2

let a > b;

```
int gcd(int a, int b)
{
    if(a%b==0) return b;
    return gcd(b, a%b);
}
```

max(a,b)/2

Q LCM

6 : 6 12 18 24 30 36 ...

8 : 8 16 24 32

lcm(a,b) Min value ?

$$Max(a,b) \le lcm(a,b) \le a*b$$

Eg a, b = {100, 11417}

lcm > 11417

Eg2 2, 100

lcm = 100

M Brute → Loop

for(x ← Max(a,b) to a*b)

if(x % Max(a,b) == 0
&& x % (a*b) == 0)

return x;

}

Cons. Max(a,b) = 10^6
a*b = 10^{12} } Very large range to iterate

Ha $LCM * HCF = a*b$

HCF ⇒ TC : $O(\log_2 \max(a,b))$

then ⇒ $LCM = \frac{a*b}{HCF}$.

M2

N = 10

a = 2

b = 3

Tell how many no's in [1 ... N] are divisible either by a or by b.

[2, 3, 4, 6, 8, 9, 10]

⇒ 7 no's.

Ans No. of numbers divisible by a, b in [1 to N] are

$\frac{n}{a} + \frac{n}{b} - \frac{n}{a*b}$

No.

Eg a = 4

b = 6

4 6 8 12 16 18 20

= 7 no's.

$\frac{n}{a} + \frac{n}{b} - \frac{n}{a*b} = 5+3-0 = 8$

XX

Not correct

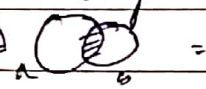
$$\frac{N}{a} + \frac{N}{b} - \frac{N}{lcm(a,b)}$$

4 6
8 12 *
→ 12 18
16 24
20
24

12 - lcm(4,6) is repeated in both the table.
so it is counted in $\frac{n}{a}$.
" $\frac{n}{b}$

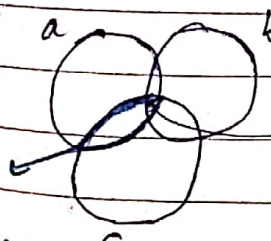
Counted twice.
⇒ decrement once. $\frac{n}{lcm(a,b)}$ no's divisible by both a, b

all other nos, 24, 36. -- will be multiples of 12 (lcm) only.



$= A + B - (A \cap B)$

Q How many no's are there in 1 to N. divisible by either a or b or c
3 no's given.



counted 3 time
how many such no's
$\frac{n}{lcm(a,b,c)}$

Counted twice
$\frac{n}{lcm(a,c)}$
subtract one time

$lcm(a,b,c) = lcm(a, lcm(b,c))$

Ans = $\frac{N}{a} + \frac{N}{b} + \frac{N}{c} - \frac{N}{lcm(a,b)} - \frac{N}{lcm(b,c)} - \frac{N}{lcm(c,a)} + \frac{N}{lcm(a,b,c)}$

Question

Find gcd of 3 numbers.
GCD(a, b, c) = ?
→ GCD can be computed in any order
eg - = GCD(a, GCD(b,c))
= GCD(b, GCD(a,c))

~~GCD can~~

~~HP~~

Question

Given an arr[N]

Function returns true if there is any subsequence that has gcd = 1.

Subsequence ⇒ can be discontinous.
{x, x - - , x}
⇒ 2^n subsequences

Constraint:
1. 0 array is not prime.
2. gcd of [4] = 4. (1 integer array)

Ans

-If gcd of arr is 1 then we can say that atleast one such subsequence exists for which gcd = 1.

Ans Coprime no's ??

No, because, may be gcd of no 2 numbers have gcd = 1, but a subsequence which may be of size 3 can have gcd = 1

CODE

```
if gcd1 = a[0]
for i ← 1 to N-1
    gcd1 = gcd(gcd1, a[i]);
    if(gcd1 == 1) return true;
```

Lecture 5

Q Factor a number N
all factor of 'N'
(assume no. all +ve)

→ Smallest factor = 1
Largest factor = N

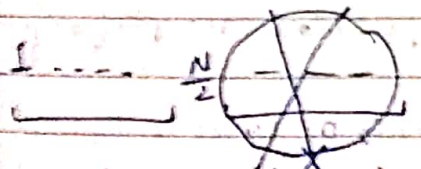
Naive

```
for (i = 1 to N)
{
    if (num % i == 0)
        i is divisible by N
}
```

TC: O(N)

Eg: N = 100.

Fact { 100, 50, ... }
second highest factor = $\frac{N}{2}$



In this region no factor is going to come.

N2

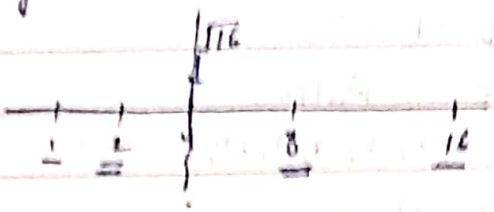
```
for (i = 1 to N/2)
{   if (num % i == 0)
        i, N/i both are factors
}
print 'N'  // N is also a factor
```

TC: O($\frac{N}{2}$)

Concept

Factors of a number n are Symmetric about sqrt(N)

Eg N = 16 :



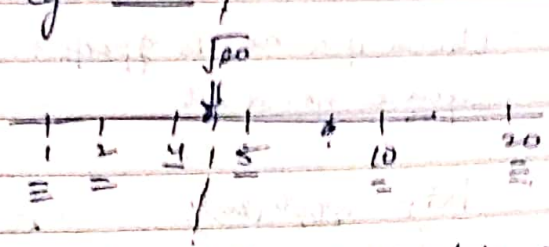
if there is a factor a.no i in N s.t. i is a factor then definetly there'll be a factor $\frac{N}{i}$ towards it.

→ It makes sense to iterate for

```
for (i ← 1 to √N) {
    if (n % i == 0)
        i, n/i both are factors
}
print 'N'
```

TC: O(√N)

Eg N = 20.



Factors will be symmetric about symmetric line.

Mathematical Proof:

If x ≤ √N & y ≥ √N
xy ≤ gN always
xy != N
Here x ≤ √N y ≥ √N

Q2 Given a number N, Check if the count of factors of N is odd. if it is count as True else false

I/p N = 20

O/p False.

1, 2, 4, 5, 10, 20. = 6.

I/p N = 16

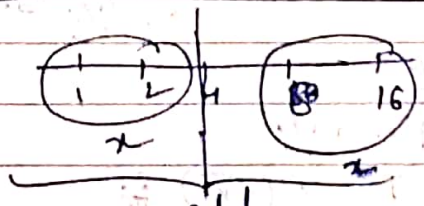
O/p True.

1, 2, 4, 8, 16 = 5

1

Concept –

For a perfect square no. – factors are symmetric about the perfect square.



Check if a no. is a perfect square no. or not.

int p = sqrt(20);

if (p * p == 20) it is perfect square.

else Not a perfect square.

→ same as finding square root of number n

TC: O(1) it is O(log n)

NB

using previous method simply count no. of factors

```
for(i ← 1 to sqrt(N)) {
    if(N%i == 0)
        print i, N/i;
}
print N.
```

TC: $O(\sqrt{N}) >> O(\log n)$

N1 is better

Q3 N doors.

all of the doors are initially closed.

C_1 C_2 C_3 C_4 C_5 C_6 C_7

'N' operations are performed.

1 →
2 → O_1 C_2 O_3 C_4 O_5 C_6 O_7
3 → O_1 C_2 C_3 C_4 O_5 O_6 O_7
4 → O_1 C_2 C_3 O_4 O_5 O_6 O_7 only 1 toggle
5 → O_1 C_2 C_3 O_4 C_5 O_6 O_7 "
6 → O_1 C_2 C_3 O_4 C_5 C_6 O_7 "
7 → O_1 C_2 C_3 O_4 C_5 C_6 C_7 "

toggle all the door no's s.t. the no. of that door is perfectly divisible by i, where i is ith operation

toggle means if it closed make it open if open make it closed

→ after 1st state,

1st → O_1 O_2 O_3 O_4 O_5 O_6 O_7

$2 \to O_1\ C_2\ O_3\ C_4\ O_5\ C_6\ O_7$

In the end, only 2 doors are open.

n and how many doors are open.

My sol"

- C_a will be opened if it is toggled odd no. of times
- $\Rightarrow C_a$ has odd no. factors in $[1, \ldots, a]$
- C_a has odd no. of factors if C_a is perfect square.
- ~~C_a is not.~~
- $\Rightarrow C_a$ is a perfect square.

MSit No. of a door is touched is count of factors of it.

For a closed door to eventually become open — odd no. of toggling should happen.

$O_1 \to 1$ factor.

$O_4 \to 1, 2, 4 \to 3$ factor

→ No. of perfect squares b/w 1 to N.

I/p N = 100

O/p: 10 → prfct square in $[1 \ldots 100]$

1 4 9 16 25 36 49 64 81 100

x = sqrt(100)

$1^2\ 2^2 \ldots \ldots \ldots x^2$

= x perfect squares.

STOP

First goto → After 3 pages

Sum of Reciprocal of prime Numbers

$$\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11}\right)^2 = \log(\log N)$$

Sum of reciprocals of natural numbers

$$\frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{N} = \log N$$

TC of seive is $O(N \log\log N)$

~~Arr[N+1]~~

prime[N+1] = {1} 0 [1 . . . N]

prime[i] → 1 is prime no.

→ 0 not a prime no.

2. ✓ prime[0] = 0

3. ✓ prime[1] = 0

1. ✓ Initialise all the values as 1 of prime.

4. ✓ for(i=2; i<=N; ++i)

{ // I should start cutting if ith no. is prime.

if(primes[i] == -1)

{

// cut all the multiples of primes[i]

~~for~~ for(j=2; i*j <= N; j++)

{

primes[i*j] = 0;

}

}

}

Optimisation 1

2 X 5 = 10 10 cut ho gya

Jab 5 ka turn ayega 5X2 = 10 to already cut ho chuka hai. Repetitive.

So start j from j = i, coz for

j < i for i

2 < 5 for 5 already included

optimisation 2

for (i loop) → i * i <= N

Q Prime Factorisation

Eg $24 = 2 \times 2 \times 2 \times 3$

- Given a number N, perform prime factorisation of this number.

I/p: 24

o/p: arr[2 2 2 3]

Code

```
arr = [ ] ; c ← 0, M = N
for i ← 2 to sqrt(N)
{
    while (N%i == 0)
    {
        arr[c++] = i;
        N = N/i;
    }
}
```

Would it ever happen

This loop will only run for those numbers which are prime. And it will not run for number which is not prime.

$404 = 4 \times 101$ → 101 is a prime number

$4 > \sqrt{404}$. $101 > \sqrt{404}$

- So in last check — it can either be 1 or 1 prime no which is greater than $\sqrt{M}$ so we need to include it.

Code:-

```
if (N != 1)
{
    arr[c++] = N;
}
```

TC: ?

$\sqrt{N}$

Inner while loop runs very less times. It will run just for prime no.'s.

Because the worst case would be that the no. that I have is

Eg 2^{32} → will run $2^{32} > 10^9$

If no. is run too many times for a prime no. x, it will run less no of times for other primes.

Because no. cannot exceed 64 bit.

⇒ TC: $O(\sqrt{N})$

M2

Is there a way to do that I just iterate for prime no's

2, 3, 5, 7 ...

instead of iterating for all numbers from 2 till $\sqrt{M}$.

⇒ store all primes in arr[ ] using Seive

⇒ TC: O(no. of primes btw 1 & sqrt(N)) + ...

Time of using seive algorithm.

→ $O(n \log(\log n))$

→ $O(\sqrt{N}) + O(n \log\log n)$

→ If only 1 query there.
Dont use seive. ⇒ TC: $O(\sqrt{N})$

→ If Q queries there; Q no(N)
Not using seive ⇒ TC: $O(Q\sqrt{N})$ too much $O(n\sqrt{n})$

⇒ use seive;
$O(n \log\log n + q)$
$= O(n \log\log n)$

M3 More optimisation.

In M2 some prime no's were useless also.

This Method-
Hey! I'll always do atleast do 1 useful work every time.

Eg 6060
↓2
3030
↓2
1515
↓3
505
↓5
101

* Whenever we reach a state, s.t, either it is 1 or a prime no. then this terminates

I want to get -
Smallest Prime factor of x.
SPF[x]

If SPF[x] can be got in $O(1)$ time.
then what is TC: ?
We know SPF[x] ≥ 2. (smallest prime no.)
⇒ after each iteration; number x is will atleast reduce to $x/2$.

for worst case: $2^{10} \to 2^5$,
⇒ all prime factors are 2 of x.

⇒ TC: $O(\log_2 N)$, for 1 query

Now how can we calculate this SPF[x] very efficiently?
Can we use seive?

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

generate SPF[] array. the no. with which 12 is cut for first time is it's SPF.

TC: $O(n \log\log n)$ + $O(Q \log_2 N)$
seive Q queries

if Q, N
TC: $O(n \log n)$ using SPF.
SC: $O(n)$ (SPF array)

SPF[] ← {-1}

Code

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

Q. Given number N
Count. no. of factors/divisors

I/P: 6

O/P: 4
1 2 3 6

6 = 2*3 O(log N) using SPF

* If 2 prime factorisations are appearing different to you, it can never ever happen that those 2 numbers are same

6 = $2^1 \times 3^1$

$0 \le x \le 1$
$0 \le y \le 1$

divisor = $2^x \times 3^y$

$\Rightarrow$

Number = $(p1)^{x1} \times (p2)^{x2} \times (p3)^{x3}$

divisor = $(p1)^i \times (p2)^j \times (p3)^k$

$0 \le i \le x1$
$0 \le j \le x2$
$0 \le k \le x3$

$(x1+1)(x2+1)(x3+1)$,
No. of Total factors

all these no's will be distinct. NO 2 are same bcoz (Read *)

O(log N) for 1 query to find out the prime factors of N.

— END —

RESUME

*1 lectures continued :- -

Prime Numbers

Defn
prime no. has just 2 factors - 1 & itself

Eg 7

1 is not considered prime no.

Q Check if n is prime.

Algo

```
for(i <- 2 to √N)
{
    if(N % i == 0) break;
}
if(i == √N+1) is prime.
```

TC : O(√n)

Q → Print all the prime no.s in 1 to N.

```
for[1 to N]
    O(√N).  for[j <- i to √i]
```

→ TC : O(N√N) = O(N^3/2)

If (N = 10^6) → 10^9 operations timeout.

Need better way

Concept

⇒ Sieve of Eratosthanes

Means filter

1 - - - N

N = 20.

How many no's are prime b/w 1 to N

X (2) (3) X (5) X (7) X X

X X (11) X (13) X X X (17)

X (19) X

[1 - - - N]

filter

allow only prime no's to pass out

2
3
5
7
11
13
17
19

It maintains 2 classes of no's.

1) X
2) Not X

⇒ First non crossed no. is always prime

1 is not prime. ⇒ X

2 - prime ⇒ non crossed no. 1st

all ~~primes~~ multiples of prime (N) are non-prime. (which is not N itself)

I'm done with 2 Now.

1st non crossed prime is 3.

3 is prime.

Cut all multiples of it

I'm done with 3.

1st non crossed prime is 5

Note → Whenever you're at number s.t that no. is cut ⇒ then no need to do anything

Note

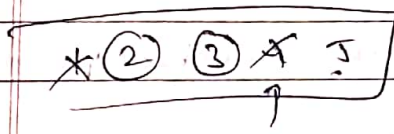
→ If that no. is not cut then it is a prime no. Cut all the multiples of this no.

→ Why 5 remained non-crossed?

Because - 5

for if it had a factor, that factor would have definitely cut 5.

factors $\leq \sqrt{N}$



I'm at N-1

$\sqrt{N} < N-1$

So yadi ye ab tak bacha hai, toh ye pakka prime no. hai

→ Whenever we arrive at no. that is already cut, we simply ignore & move ahead.

Why don't we cut 4 multiples - Bcoz - there would have already cut by that no. which cut 4.

TC: ?

cut 1.

at 2, cut all 2 multiples.

$\Rightarrow \frac{n}{2}$ operations

at 3, $\frac{n}{3}$ operations

at 4 ⇒ 0 operations coz already cut

at 5 ⇒ $n/5$

at 6 ⇒ 0 (Crossed no. move ahead)

N

$\Rightarrow \frac{N}{2} + \frac{N}{3} + \frac{N}{5} + \frac{N}{7} + \frac{N}{11} + \dots$

$N\left(1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} \dots\right)$

Sum of all reciprocals of prime no.

$1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \dots \frac{1}{N} = \log\log N$

TC of Seive algorithm $\Rightarrow (N \log\log N)$

Better

$N \times \log\log N < N \times \sqrt{N}$

Sum of reciprocal of natural no's.

$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots = \log N$

Store all values in boolean/int array.

- Continue from back.

Go Back

Lecture 6

Solve Algo

Conc: Space

10^9 space → V. Huge.
Not possible.

[1 - - - 10^6]
N

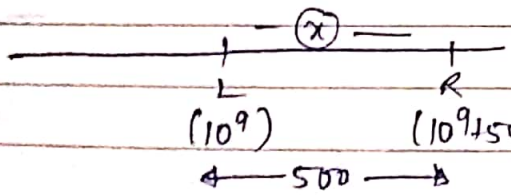
prime [10^6]
N

int L = 10^9; L ≤ K
int R = 10^9+500;

[L - - - R]

Storing 10^9+500 → int prime[n]
huge.

→ Declare –
arr [R-L+1]



Initially status of x was represented in primes[x]

Now, we make primes[R-L+1]
we store x at
primes[x-L]

Prob 2:

I want to cross non prime in [L - R] to know which all are primes in [L - - - R]

To do this -
I want to ~~good~~ strike off non primes.
I'll need to know prime nos $\le k$
$\therefore$ $p < 10^9$

- we need only the primes upto $\sqrt{n}$

Eg [90 - - - 100]

```
i=11
for(i= 2; i*i<=n; i++)
{
    cond^n
    for(j=i; i*j<=n; ++j)
    {
        j*i
    }
}
```

prev value = 11×11 = 121 > 100
not useful to me

→ all $i > \sqrt{100}$ will not be useful to me

→ for primes[1 to $\sqrt{N}$]
→ safe to compute primes[10^5] & store all primes in pval[k]

for primes = 2 3 5 7

I don't want to iterate for all j = 2, 3, for prime 2.

1st value useful to me => 45th iteration

bcoz 2 * 45 = 90.

$n = \sqrt{R}$

for prime = 7, first iteration will be 13th. 7 x 13 = 91

So - take ceil $(\frac{L}{x})$ till $\frac{R}{x}$.

Q TC: ?

$R - L <<< L, R$

$n = \sqrt{R}$

$(n \log\log n)$ → seive

$O(\sqrt{R} \log\log\sqrt{R}) +$ ⟶ negligible $R-L <<< R$

Q Modulo Arithmetic

(Arithmetic of Remainders)

a, b

Rem: a % b

1. Mod operator property

$x \% m = [0, \dots, m-1]$

let $m = 5$, $x \in R$

$Rem \in [0, \dots 4]$

iterating over x

x+m						
0:	0	5	10			
1:	1	6	11			
2:	2 (x)	7 (x+m)	12	17	22	---> family
3:	3	8	13			
4:	4	9	14			

m, 0, m

Time period is m.

Repetion of over m period

$x \% m$

2, 7, 12, 22 -- belong to same family

These no. are congruent to each other. Modular Congruence

$2 \pmod 5 \equiv 7 \pmod 5 \equiv 12 \pmod 5$

① $(a+b) \% m = ?$

$\in [0, \dots (m-1)]$

$(a+b) * c = a*c + b*c$

because * is a distributive operator we can do this

Can

$(a+b) \% m$ written as $a \% m + b \% m$ $\times\times$

$\in [0 -- m-1]$ $[0 \dots m-1]$ $[0 \dots m-1]$

$= [0 -- 2m-2] \geq m$

$$[a+b) \% m = (a \% m + b \% m) \% m$$

Remainder of Sum = (Sum of remainder) % m.

Proof HW

	2	m = 2
	m - 1	m - 2
	a/m	b/m

$[m^{m+1} - 2m-2^{(2m-3)}]$

Observation based approach

2. $$[(a-b) \% m = (a \% m - b \% m + m) \% m$$

$[0 - \dots (m-1)]$

$a \% m - b \% m = [-m+1 \dots m-1]$

Some compiler error for -ve no. ka mod. So simply add m

$a \% m - b \% m + m = [0, \dots 2m-1]$

$(a \% m - b \% m + m) \% m = [0 \dots m-1]$

3. $(a*b) \% m = (a \% m * b \% m) \% m$

4. $(a/b) \% m = ?$

* Not distributed easily over division *

Inverse modulo done.

Q int arr[N] Non-ve int
int k
count total no. of pairs (arr[i], arr[j]) s.t. their sum is divisible by k.
i ≠ j

(arr[i] + arr[j]) % k == 0

I/P: arr[2 2 1 7 5 3]
k = 4

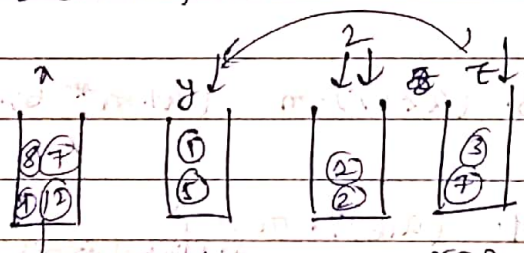
O/P: 5
[2, 2]
[1, 3]
[7, 1]
[5, 3]
[7, 5]

(a % m + b % m) % m = 0

[0 - 2m - 2]

only 2 multiples of m — 0, m

buckets from 0 to m-1



$^{2}C_{2}$
1
two ways 2 pick two diff. nos from same bucket

If no. of bucket
if r = 2, I don't want to take same no.
$^{n}C_{2}$ for r = 2.

If total bucket is odd, this case won't arise

for even no. of buckets take care of total middle element

Code

```
int cnt[k] = {0};

for i ← 0 to (N-1)
{
    cnt[arr[i] % k]++;
}

ans += cnt[i] * cnt[k-i]
handle cnt[0] case diff;
```

TC : O(n + k/2)

If k is v.v. large opt for $O(n^2)$ approach

Q Sum of triplets to be divisible by m

Ans: 2 loops
fix 2 remainders a, b
I can comment on what c should be

Q int arr[N] Non-ve int
int k
count total no of pairs (arr[i], arr[j]) s.t. their sum is divisible by k.
i ≠ j

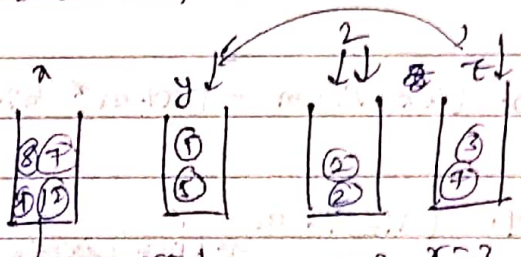
if: (arr[i] + arr[j]) % k == 0

I/P: arr[2 2 1 7 5 3]
k = 4

O/P: 5
[2, 2]
[1, 3]
[7, 1]
[5, 3]
[7, 5]

(a+b) % m = 0
(a%m + b%m) % m = 0
[0 ... m-1]
only 2 multiples of m = 0, m

buckets from 0 to m-1



$^{2}C_{2}$
1
two ways 2 pick two diff nos from same bucket

If no of bucket
if r = 2, I don't want to take same no.
$^{n}C_{2}$ for r = 2.

If, total bucket is odd, this case won't arise
for even no of buckets there can of total middle element

Code
int cnt[k] = {0's};
for i = 0 to (N-1)
{ cnt[arr[i]%k]++; }
ans += cnt[i] * cnt[k-i]
handle cnt[0] case diff's

TC: ~~O(n)~~ O(n+k/2)

If k is v.v. large opt for N) $O(n^2)$ approach

Q Sum of triplets to be divisible by k

Hint: 2 loops
fix 2 remainders a, b
I can compute in what c should be

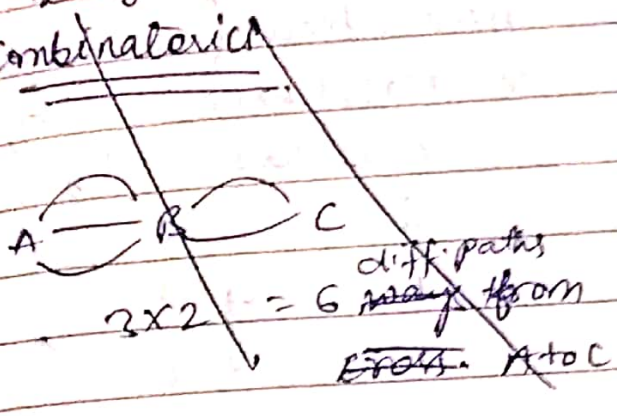
HW Basics of STL

Many edge cases –

r1 r2 r3

x y z

Combinatorics



1. r1, r2, r3 - diff.
   Contribution: x x y x z
2. r1, r2 - same, r3 - diff ${}^{x}C_{2} \times y$
3. r1, r2, r3 - all same ${}^{x}C_{3}$

r1 ← 0 to n-1

r2 ← ?

coz Repetitions should not be there

r2 ← r1+1 to n-1

Run your loop in a ways s.t. no repetitions happen.

Combinatorics

Q Bitstring of len N.

N = 4 - - - -

Find no of diff. bitstrings with exactly 'k' ones.

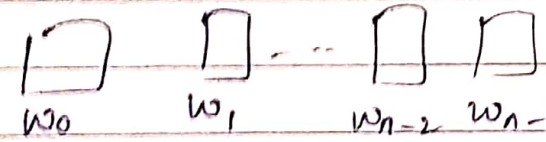
k = 2

O/p: $^4C_2$ (coz order doesn't matter)

^nC_k

If let say order matters then ans = $^nC_k \cdot k!$

Q 'N' walls.



3 colours (R, G, B)

diff possible arrangements can be there if i need to pick & color 3 walls.

$\frac{^nC_3 \cdot 3!}{}$

Pick 3 walls from n

w_i	w_j	w_k
R	G	B
G	R	B
B		

Q 'N' boys, 'M' girls.

Conduct a play.

I need a team of t members & Must have ≥ 4 boys

Must have ≥ 1 girls

G	B
1	t-1
2	t-2
	⋮
t-4	4

$^mC_1 \cdot {}^nC_{t-1} + {}^mC_2 \cdot {}^nC_{t-2} + \cdots {}^mC_{t-4} \cdot {}^nC_4$

$^{n+m}C_T$

$^{60}C_{10}$ → V. high!

$^{n+m}C_T - {}^mC_0 \, {}^nC_t - {}^mC_1 \, {}^nC_0 \, {}^mC_{t-1} \, {}^nC_1 - {}^mC_{t-2} \, {}^nC_2 - {}^mC_{t-3} \, {}^nC_3$

Pascal triangle

$^1C_0$

$^2C_0$ $^2C_1$ $^2C_2$

$^3C_0$ $^3C_1$ $^3C_2$ $^3C_3$

$^4C_0$ $^4C_1$ $^4C_2$ $^4C_3$ $^4C_4$

1 5 10 10 5 1

$$^NC_r = {}^NC_{n-r}$$

$$^NC_r = {}^{N-1}C_r + {}^{N-1}C_{r-1}$$

No way to compute ^NC_r with value overflow

2, 1 R

20000

30-04-2020

THURsday Homework discussion

Q1 Rotate Image Leetcode

rotate matrix 90°

without extra space

```
[[1,2,3],          [[7,4,1],
 [4,5,6],    →      [8,5,2],
 [7,8,9],           [9,6,3]]
]                  ]
   A                  Ans
```

- 1 row converted to last column
  ith row → ith column from end
- Make rows into columns → TRANPOSE

```
// 1 2 3          // 1 4 7
// 4 5 6    →     // 2 5 8
// 7 8 9          // 3 6 9
   A                 AT
```

Reversing rows is same as swapping columns; C_i & C_{n-i}

```
// Reverse rows (swap col's)
// 1st ⇒ nth column
// 2nd col ⇒ (n-1)th column
```

Transpose means - swap(a[i][j], a[j][i])
↓ do it for half of matrix.

(if done for full, $a_{ij} \leftrightarrow a_{ji}$; then $a_{ji} \leftrightarrow a_{ij}$, swapped twice, effect is null)

Q2 Pascal's Triangle Leetcode

```
      1
    1   1
  1   2   1
1   3   3   1
1 4   6   4   1
```

Write it in this form -

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

This way, → observe

$$a_{ij} = a_{(i-1)j} + a_{(i-1)(j-1)}$$

Every row has elements equal to row no.

- ith row has i elements
- out of which 1st and last element is always 1, & remaining can be calculated.

- a[i][0] = a[i][i] = 1

```
for(int i=2; i<numRows; ++i){
    Vector<int> tmp;
    tmp.push_back(1);
    int j=1;
    while(j<i-1){
        tmp.push_back(ans[i-1][j-1] + ans[i-1][j]);
        j++;
    }
    tmp.push_back(1);
```

ans.push_back(temp);

}

TC: $O(n^2)$
SC: $O(n^2)$

q) let question be calculate value of ith row, jth column
a_{ij} th col-
instead of using pascal triangle use

$$^{n}C_{r} = \frac{n!}{r!(n-r)!}$$

→ can create factorial array. and use this formula

wow! ☆ ⇒ SC: O(n)

Q3 A Star Sky Problem Codeforces.

Stars with a initial brightness. Brightness keep on increasing until it reach c, after reaching c, it go back to 0.

// n*n matrix

c ⇒ ~~matrix~~ Max brightness of a star = 3

Star ⇒ brightness 1 ⇒ t = 0
t = 1 ⇒ 2
t = 2 ⇒ 3
t = 3 ⇒ 0

given time t l_1, r_1, l_2, r_2 ⇒ sum of brightness of stars.

Technique -
prefix sum on a matrix

arr[l2,r2] - arr[l1-1,r2] - arr[l2,r1-1] + arr[l1,

arr(l2,r2) - arr(

t=0. a_{33}

after c time, whole matrix would convert to ~~same as~~ original matrix.

$t = c \quad a'_{33} = a_{33}$

c=3

1 - -
- - -
0 -

→

2 - -
- - -
- 1 -

→

3 - -
- - -
- 2 -
↓
0 - -
- - -
3 -

after this Matrix same as 1st one

∝ c ≤ 10

⇒ at max 11 different matrices.

$b_0, b_1, \ldots b_{10}$
bright at t=0, t=1, t=10

0, 1 - - - - - 10 11. $b_{11} = 0, \ldots$

Time period = 11

at time t, t = 10, 10 % 4 = 2 nd matrix
c(t+1)

3 - -
- - -
- 2 -

to handle multiple star at a point-
Create -
a[i][j][k] => frequency of stars with brightness k at (i,j)

[1,3], [2,4] => what is count at position 2?

[1, 1, 0, -1, -1] => [1, 2, 2, 1, 0]
idx: 1 2 3 4 5

HW

Q1 Pasha and string

Q a) H.NO // 1, 2, 3, 4, - - - Destination

No. of people in a house // 3, 4, 1, - - -

Everyone comes out & moves towards the destination.
How many people pass through HOUSE 3.
Ans is prefix sum

Q b) people from houses settling at other house in the way

H.No: // 1, 2, 3, 4, 5

// 3 people => H.NO 1 → 3

// 4 people => H.NO 2 → 4

Gold Mars Trickery

Gold Mars Trickery

// [3, 4, 0, -3, -4]

prefix sum => [3, 7, 7, 4]

+1, -1 approach

[l1, r1], [l2, r2] - - -

add 1 at a[l1], decrement 1 at a[r1+1]

Q) s = "abcdef"

even no. of reversal of same string
even => same, odd => reversed

reversal 1 => reversal 2 => fbcdea
reversal 2 => reversal 1 => fbcdea
. order of reversal doesn't matter

Calculate no. of times ~~each~~ each index is being toggled.
→ Can be done using Gold Mass Trickery

Q) Recording

So

→ M TC: O(n)

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

A, B → V

if my vote A => count++;
& for every every vote for someone else, I am cutting the vote of A. count-

at the end if vote A > 0 => A is in majority

★ Moore's Voting Algorithm

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

Fix 1st element as majority

Vote for 2 => 1 → 2 → 1 → 0
Means nothing out yet

vote for 1 now element changes
vote for 1 = 1 → 0 nullify two guys

vote for 1st element
=> vote for 2 = 1
// Majority is 2.

Moore voting algorithm

This algorithm is designed to give majority as answer if majority exists. ($no > n/2$)

If majority doesn't exist, it will give a random answer.

// [4,3,3,3,4,4,4,2,2,4,4]

this gets nullify

vote[4]=1, till idx 2. vote[4]=0 "gets nullify"

vote[3], for idx 3 to 6, gets nullify

vote[4], idx 7 to 8

vote[2], idx 9 to 10

vote[4] & idx 11 ⇒ 4 is majority

Q2 Majority ($n/3$)

~~my~~ M +2 −1

there can be two majorities here.

// leader1, leader2 ($n/3$ votes)

So leader1, leader2 won't deduct from the votes of each other

If 4 and 3 are contenders of majority

// 4 => $n/3$

// 3.

4 needs to prove majority in remaining array

↳ array may or may not contain 2nd contender (3)

Similarly

3 needs to prove majority in remaining array (except 4)

↳ [3,3,3,2,2]

[

// 4 ⇒ 4 → 4 → 2 → 4

// 3 3 ? → 3

Now both can or cannot be a majority.

So we need to iterate the loop again for value 4 & 3.

if cnt[4] > $n/3$ 4 is majority element

cnt[3] > $n/3$ 3 is also majority element

It may happen that there's only 1 majority.

if cnt[x] > $2n/3$ no second majority won't exist

→ Moore voting algo is used in only these kind of problems.

[4,3,2,3,2,3,4,4,4,3,4]

//4 → 1 → 0
//3 → 1 → 0 ← both reaches 0
idx 1 to 3 nullify.
Pick next 2 elements. Vote for them.

//3 → 1 → 2 → 1
//2 → 1 → 0 ← Reaches 0.
idx 4 to 7 nullify
Pick next 4 element as contender

//3 → 1 → 2 (already)
//4 → 1 → 2 → 3

Check in loop for 3 & 4

Q- First Missing Positive

My M

TC : O(n) SC : O(n)
want to O(1)

Approach - Rearrangement

[3, 4, -1, 1]

[4, 1, -1, 3] → 2 is not there.

Rearranging is actually putting a[i] = i.

So whenever a[i] ≠ i, that is answer.

So 2 is my ans

[5, 4, 0, 1, 3, 4, 2]
[4, 1, 0, 3, 4, 5, 2]
[0, 1, 4, 3, 4, 5, 2]
[0, 1, 2, 3, 4, 5, 4] ← idx = 6 != a[6]

So ans = 6

Note
⇒ There would be 1 corner case that would be stucked here
↳ eg [3, 1, 2] [1, 3, 2]
on Rearranging ⇒ [3, 1, 2]
⇒ I would say

~~[3, 1, 2] → [3,~~
[1, 3, 2] → [3, 1, 2]
on rearranging
we would say 1st index got missing as 3. by checking index will be 3. but 3 is already present.

So better -
instead of starting rearranging from as putting a[val] = idx

do as a[0] = 1 a[idx+1] = idx
a[1] = 2

Start with 1 value at idx 0 itself than 2.

[1, 2, 3]

that way 3 would be at right location in array

idx = 1 a[i] > n ✗
idx = 2 (do it for made -ve values also bcoz vo index bhi to -ve krni hai

→ [-3, INT_MAX, -1, 4]

idx = 3 [-3, INT MAX, -1, -4]

first positive Value found. Iska idx will be my ans

→ ans idx = 1

(actual 2, satisfied from 1)

M2 Ankur

First Make all val <= 0 & val > n to INT_MAX

Eg [3, -1, 1, 4], n = 4

↓

[3, INT_MAX, 1, 4]

then

```
for(int idx=0; idx<n; ++i){
    if(a[idx] < n){
        make value at idx2 = abs(a[idx]) negative
        a[abs(a[idx])] = -abs(a[abs(a[idx])])
    }
}
```

idx = 0 → [3, INT_MAX, -1, 4]

Lecture 7

Balanced Parantheses

(()()) → balanced
())(→ Not balanced

Q1 N pair of small braces
Tell count of no. of distinct bal. parantheses.

	Valid arrangements	
N=1	()	1
N=2 (2 openings & 2 closing braces)	()() (())	2
N=3	()()() (())() ()(()) ((())) (()())	5

I/p: N.
O/p: cnt of all valid braces

M1 Recursion
Exponential. V. High
I only need count

M2 ⇒ N '('
N ')'

will it always be balanced
No.

Eg N=2))((XX

M2. (: +1
) : −1

(+1 → ((→ (() ...] larger no. of elements & duplicate
((+2 → (((
(((+3

We can start with 1 opening brace / 2 opening brace / 3 ...
Observation is correct. But lot duplicates

N sol:-

Starting braces ⇒ '('
For any prefix
No. of opening braces >= closing braces

('N' pairs)
Bal:

(
↓
it will have its closing counter part somewhere in right.

(()())
0 1 2 3 4 5

(()) ()
0 1 2 3 4 5

Once you've put 1 opening brace & 1 close brace
( )

1 pair is gone ✗

Now (N-1) pairs.

possibilities -

( )

(i) 0 inside, (N-1) outside.

cnt[0] cnt[N-1]

( ) (( ))

( ) ( ) ( )

N-1 braces outside

(ii) 1 inside, (N-2) outside
cnt[1] cnt[N-2]

(iii) 2 inside, (N-3) outside

(N-1) braces inside, 0 outside

All categories different from each other

(ii) (( )) ( ) 1 inside, 1 out

(iii) (( ) ( )) 2 inside, 0 out

((( )))

Let's say I know -

Count of balanced parenthesis for 2 braces = C1

Count of balanced parenthesis for 1 braces = C2

for category (iii) 2 inside, N-3 outside

C1 x C2

for N pairs, - 1 pair taken as reference

cnt[N] = Count of distinct balanced parenthesis of N pairs.

If I know -

cnt[0] cnt[1] - - - - - [N-1]

known

then I can easily find out cnt[N]

cnt[N] = cnt[0] * cnt[N-1]

+ cnt[1] x cnt[N-2]

+

.

.

+ cnt[N-1] x cnt[0]

Catlan's Numbers

$$cnt[N] = \sum_{i=0}^{N-1} cnt[i] \times cnt[N-1-i]$$

⇒ cnt[0] = ?

0 pairs.

Cant consider ans = 0, coz that is a empty string basically. ⇒ Null parenthesis. It's also balanced.

So cnt[0] = 1

as . cnt[0] x cnt[N-1] ≠ 0

N = 0 1 " "

N = 1 1 ( )

N = 2 2 ( )( )

(( ))

N = 3.

#1 0 I, 2 out

( ) ( ) ( ) ( )

(( )) ( ) ( )

#2 1 I, 1 out

( ) → (( )) ( )

#3 2 I, 0 out

(( )) (( ))

(( )) → ((( )))

~~Lecture~~

Its just like fibonacci no. if you know $f(n-1)$ $f(n-2)$ u can know $f(n)$

TC: ?

cnt[i] →

1 + 2 + 3 + 4 + ... + N =

for finding cnt[i] — for finding cnt[i] — for cnt[N]

$= O(N^2)$

Catalan Numbers ★★

$$cnt[N] = \sum_{i=0}^{N-1} cnt(i) \times cnt[N-1-i]$$

Many tough combinatorics question.

Code:

```
int N;
int C[N+1];
C[0] = C[1] = 1;
for i ← 2 to N {
    C[i] = 0;
    for j ← 0 to i-1 {  // coz finding cnt[i].
        cnt[i] += cnt[j] x cnt[i-j-1]
    }
}
```

Generic Catalan Expression → $C_n = \frac{1}{n+1} {}^{2n}C_n$

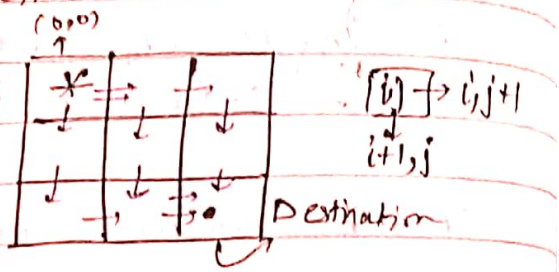
↓ faster to find n catalan no. Coz it is $O(n)$

This formula is just derivation of prev. one.

HW Derive on ur own.

Remember Logical Expression & its Logic

Q2 Given an n x n Grid



distinct ways [(0,0) → (n-1, n-1))

Q) bit string of n. Exactly k bits are 1. How many ways.

Ans- ${}^{n}C_k$

(0,0) → (3,3)

4 moves in every path.

⇒ No. of moves that I'm going to take is same = 4 = (2n-2)

2 → & (n-1) horizontal moves

& → (n-1) vertical moves.

S: VVHH

S: HVVH

any path is represented as string of H V.

Size of string = 2n-2

How many ~~exactly~~ string there where exactly (N-1) are V.

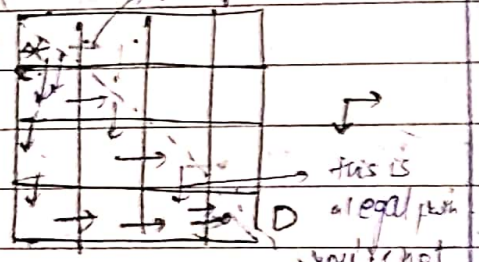
$^{2n-2}C_{(n-1)}$

for mxn matrix

$^{m-1+n-1}C_{m-1}$

$^{m+n-2}C_{m-1}$

Q3 Given a nxn sq. matrix



~~How many~~ Count of distinct paths that help me reach (n-1, n-1) s.t. none of the paths cross the major diagonal.

⇒ Whatever path I choose it can chose lower triangle cells only

Ans is just like Catalan Numbers.

nxn matrix ⇒ (n-1) Vertical moves

⇓

Find (n-1)th catalan number

= cat[N-1]

= $^{2n-2}C_{n-1}$

M See it as Balanced Parenthesis.?

legal path: VVVVHHHH ((()))

VHVHVH ()()()

VHHHVV ⇒ ()))((

no legal path

It can never happen

cnt(H) > cnt(V)

↳ property of lower triangle is i≤j

Remember Catalan for such property

Concept

Q Is -

⇒ Create a string of size (2n-2).

cnt(V) ≥ cnt(H).

& 1st move should always be V move.

(N-1) pair of (k,N) there for
N×N matrix

Q Permutation

Given a string 's'
all characters are
unique.

S = "acb".
How many permutations of S?

Ans → N!

1 abc
2 acb
3 bac
4 bca
5 cab
6 cba

Sorted in lexicographic word.

Q simply tell lexicographic rank of given string

Rank ["acb"] = 2

Rank ["rqp"] = ? 6

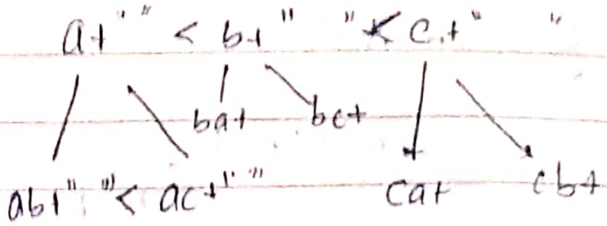
3! permutations.
pqr
prq
qpr
qrp
rpq
rqp

M1

Generate all permutations, then check for S?
TC: O(N!) to generate all

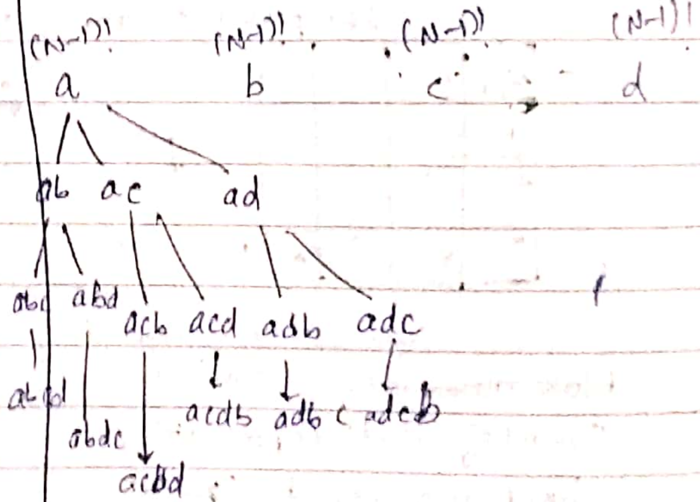
M2

~~all p string~~
lexicographic Rank -



String : dacb

Choice : a, b, c, d.



Distinct permutations start

Start from a's $(n-1)!$

$\underbrace{\quad\quad}_{(n-1)!}$

Rank of $3 \times 3(n-1)!$

rank = k

d c a b

rank+1 = $3(n-1)!$

d

$(n-2)!$ start — db — dc

$(n-2)!$ — dca — dcb

rank+1 = $2(n-2)!$

rank+1 = 0

Eg adcb

rank = 0

rank+1 = 0

Once you

a — — —

Choices - b, c, d

Subtree start from b.

* → Once you've used a. Then all the elements > 'a' their ranks need to be decreased by 1

'a' = 0

'b' = 1-1

'c' = 2-1

'd' = 3-1

rank 'a' = 0

'b' = 1

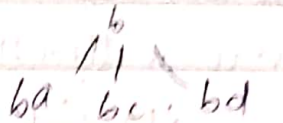
'c' = 2

'd' = 3

rank+1 = $2 \times (n-2)!$

Rank need to decreased for all c > a

Eg bcad



bc — — → rank of b

rank+1 = $1 \times (N-1)!$ — Rank c - -, rank d - -

rank+1 = $\frac{1 \times (N-2)!}{\text{rank of c}}$ — Rank d - -

rank+1 = $\frac{0 \times (N-3)!}{\text{rank of a}}$ — Rank d - -

rank+1 = 1.

• Compile fact[N+1] & fact[1 to N] earlier.

Sort string to get initial ranks.

$(n-1)!$ → $O(26 \times 1)$ upper bound $O(1)$

$(n-2)!$ → "

$(n-3)!$ → "

b

TC = $O(N \times 26)$

Precomputation: Sorting $O(N \log N)$

Core logic: $O(N \times 26)$

TC: $O(n\log n) + O(26n)$

Q Given an int N, N>0

Count total no. digit that are 1 in all the ints from 1 to N

I/p: N = 13

O/p: ? 6

1 2 3 4 5 6 7 8 9

10 11 12 13

M Brute.

Iterate i:1 to N:

check if i has 1:

```
while(i!=0){
    if(i%10==1) cnt++;
    i=i/10;
}
}
```

TC: ?

$O(N \times \log_{10} N)$

no. reduce to 0 in $\log_{10} N$ time

for N = 10^{18}

Timeout.

M

Count no of 1's in place value, ones value.

Reverse Lookup Approach.

Eg Q) Sum of all subarray.

we found out contribution of each element, i.e., no. of subgroups containing a[i].

sum += i x a[i]

Here -

= cnt of 1s (ones) 2

+ cnt of 1s (tens) 4

+ ... + cnt of 1s (1000) 0

Eg. N = 5149

(i) ones: [1], [11], [21] [31] [41] .. [5141]

1 x 10, 1 x 10

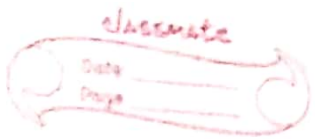
Smallest no. having 1 at ones pos

largest no. ≤ 5149 having 1 at ones pos

tens: [10 11 12 13 ... 19] [110 111 112 113 ... 119] [210 211 212 213 ... 219] ... 5110 5111 ... 5119

10 x 10, 10 x 10

tens



100 [100 101 102 ... 199]

[1100 1101 1102 ... 1199] 2100 2101 2102 ... 2199

$100 $101 ... $149 $149

$5149

Hundred

Careful about last chunk.

1000 1001 1002 ... 1999 11000 $5149

I + I know gap. Can I count no of chunks

$\left(\frac{N}{10^{i+1}}\right) \times$ (in each chunk i value same)

$= \frac{N}{10^{i+1}} \times i$

Only last chunk be careful.

$5149 - $100

Assignment

I/p: N

$\sum_{i=0}^{N-1} cat(i) \times cat(N-1-i)$

Make cat[n+1]

cat[0] = cat[1] = 1

cat[2]

lecture 7

Asymptotatic ANALYSIS

```
for(i=0;i<N;i++)
{
    for(j=0;j<=i;j++){
        cout<<i<<j;        O(1)
    }
}
```

No. of operations = ?

i=0 i=1 i=2 ... i=N-1

$1 + 2 + 3 \ldots N = \frac{N(N+1)}{2}$

$= \frac{N^2}{2} + \frac{N}{2}$

↳ for N = large

$TC = O(N^2)$

★ Recurrence Equation

tells to express run time of code on large I/p' O(N) to the runtime of same code on some small I/p. n

$T_N = a \times T_n + b$.

100ms. 10ms

Eg $T_N = T_{N-1}$

In prev. ques:-

I/p: $N \rightarrow (N+1)$

In code:

$i = N$ $(N+1)$ more operation

No. of operation = No. of oper $(N+1)$ + $(N+1)$ (N)

$\Rightarrow \boxed{T_N = T_{N-1} + O(N)}$

I/p size ↑ ⇒ TC: O(N)↑

$T_{N-1} = T_{N-2} + O(N-1)$

$T_N = T_{N-2} + O(n-1) + O(n)$

$= T_{n-3} + O(n-2) + O(n-1) + O(n)$

$= T_0 + O(1) + \dots + O(n)$

$= \frac{N(N+1)}{2}$

Space Complexity

① SC of your code?
② Extra space of your code?

Q1 & Q2 are slightly different.

SC means overall memory that code has utilised.

* Extra memory excludes the size of input.

Q) Tell SC?
Need to tell extra space. ②

Eg
```
void sort(int arr[], int N)
{
    ...
}
```

SC: O(1)

Eg
```
void func(int arr[], int N)
{
    int P[N][N];
}
```

SC: $O(N^2)$

Q) I/P: int N

```
int func(int m)
{
    int i=0, j=0;
    for(; i<m && j<m; ){
        if(i==j){
            i++; j=0;
        }
        else{
            j++;
        }
    }
}
```

No. of operations = ?

My M

$O(1) + (1+O(1)) + (2+O(1)) \cdots (m-1+O(1))$

$\frac{m(m+1)}{2} \rightarrow \frac{m(m-1)}{2} + m$

My Sir

$i=0, j=0$ $O(1)$

$i=1, j=0$ $j=1$ $O(2)$

$i=2, j=0$ $j=1$ $j=2$ $O(3)$

$i=3, j=0$ $O(4)$

$i=m-1, j=0$ $O(m)$

$O(m(m+1)) = O(m^2)$

Like Nested Loop.

SC: $O(1)$

① Nested Loop not necessary

TC is $O(n^2)$

Q Factorial (N) =

$N \cdot N-1 \cdot N-2 \cdots 3 \cdot 2 \cdot 1$

$\boxed{fact(N) = N \cdot fact(N-1)}$

$N=0$ $fact(0) = 1$

Eq^n doesn't hold true for $N=0$.

Recurrence Relation hai

```
int fact(int N)
{
```

```
int fact(int N)
{
    if (N==0) ret 1;
    ret N * fact(N);
}
```

TC : $O(n)$

Catalan Numbers.

pdf

1. Problems

1.1 Balanced Parenthesis

n pair

'(' - +1 ')' - -1.

Cond ① no. of '(' == no. of ')'

cond ② Sum of sequence >= 0

cnt(n=0) = 1.

1.2 Mountain Ranges.

n=0 *

n=1 /\ 1 way

n=2 /\/\, /\\/ 2way

n=3 /\/\/\, /\\/\, /\/\\, /\\\/, /\\\\/ 5way

/ → +1 \ → -1

cond 2 cnt(/) >= cnt(\)

Sum >= 0

Same cond's.

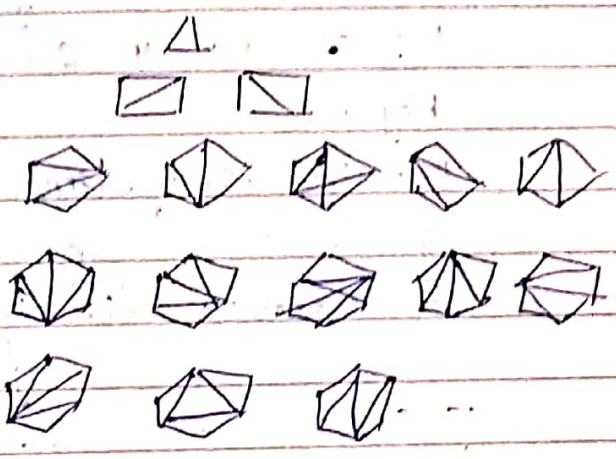
1.3 Diagonal-Avoiding Paths

moves(H) moves(V)

+1 -1

1.4 Polygon Triangulation

· Triangulate polygon, (n+2) sides.



1.5 Hands across a table.