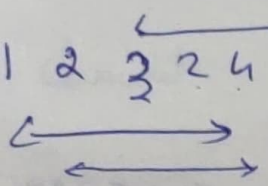
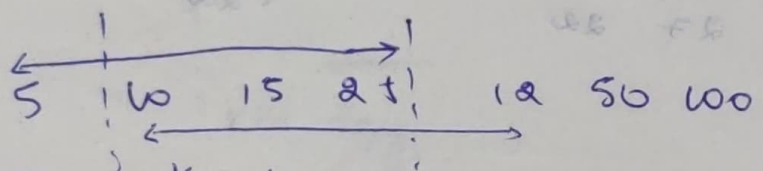
28/12/21 Elite Programming

1 n = 11 k = 4 (window size)



o/p = 8, 11, 14,



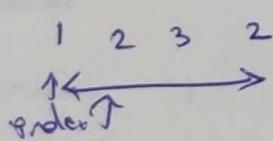
n = 7 k = 4

Sum = 55 - 5
= 50 + 12
= 62

o/p = -

Sum = 55

n = 11 k = 4



O(n²) ∝.

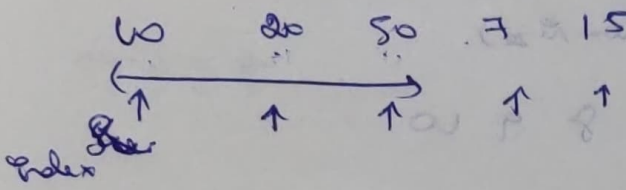
Sum = 8
index =

o/p = 8

Ans

index - 1 = subtract
index = add
index + k - 1

N = 7 k = 3



Sum = 80, -10 +7 = 77 - 20 +15 = 72 - 50 + 5 = 27

= 27 - 7 + 2

= 22

O/p:

80 77 27 22

O(n)

---

4 -K

environment

2 1 2

Boy Chocolate.

Day 1 2 3

C C C ✓

I I

I I I

C C I ✓

C I C ✓

I C I ✓

I C C ✓

N = 2

Day	1	2	3	4	...	N
Chocolate	1	1+1 = 2	3	5		
Icecream	1	1	2	3		
Total	2	3	5	8		

Kth day

Multiple of k = 2

N = 6, k = 3 (3rd day)

Day	1	2	3	4	5	6
Chocolate	1	2	3	6	9	15
Icecream	1	1	3	3	6	15
Total	2	3	6	9	15	30

o/p

Equal left & Right Sum

N = 5

2 3 10 10 15 0

0 1 2 15 5 5 15 15 5

o/p 10 -10

for ——

{

for left sum

for right sum

left == right

}

arr = 2 3 10 -10 15

Sum = 2 5 15 5 (20) → Total sum

2+3 ↓ 2+3+10 ↓ 2+3+10-10 ↳

2 < 2-2=0 / Tot - 2 = 18 / 20 - 2 = 18

3 < 5 (summary) - 3 = 2 / 20 - 5 = 15

-10 < 5-(-10) = 15 / 20 - 5 = 15

10 < Sum - arr 15 - 10 = 5 / Sum - arr Sum 20 - 15 = 5

15 < 20 - 15 = 5 / 20 - 20 = 0

N = 7

Arr 2 4 6 -10 12 22 -22

Sum 2 6 12 2 14 36 (14) → Total sum

2 < 2-2=0 / Total - sum at that place

4 < 6-4=2 / 14-6=8

6 < 12-6=6 / 14-12=2

-10 < 2-(10) = 12 / 14-2=12

Sum - arr < Total - sum arr

Maximum Sum Sub. Array

N = 5

3	2	-2	5	-4

3 → (3) Sum

3 2 → (5)

3 2 -2 → (3)

3 2 -2 5 → (8)

3 2 -2 5 -4
↳ (4)

2 → (2)

2 -2 → (0)

2 -2 5 → (5)

2 -2 5 -4.
↳ (1)

-2 → (-2)

-2 5 → (3)

-2 5 -4
↳ (1)

5 → (5)

5 -4
↳ (1)

-4 → (-4)

o/p = 3 2 -2 5 → (8)

Sub array is Continuous.

Kadane's algorithm

Swings debt

100 -50 } +ve ✓

100 -50 -1 } -ve

~~Max sum = 3~~

Current Sum = 0 3 3 3 3 4

Max Sum = 0 3 3 8

---

2 5 8 10 4

2+0
2

Curr Sum = 0 2 7 15 25 29

Max Sum = 0 2 7 15 25 29

N = 5

3	2	-2	5	-4

3+2 = 5
2

3+2-2 = 8 = 4
-2 5 -4

-5 -6 -1 -4 -10

-5+0 -5 -5-6 -6

-1-6 = -7 -4-1 -4-10

-1 -4 -10

Currsum = ~~-5~~ -5 -6 -1 -4 -10

Maxsum = 0 -5 -1 "

first element in array

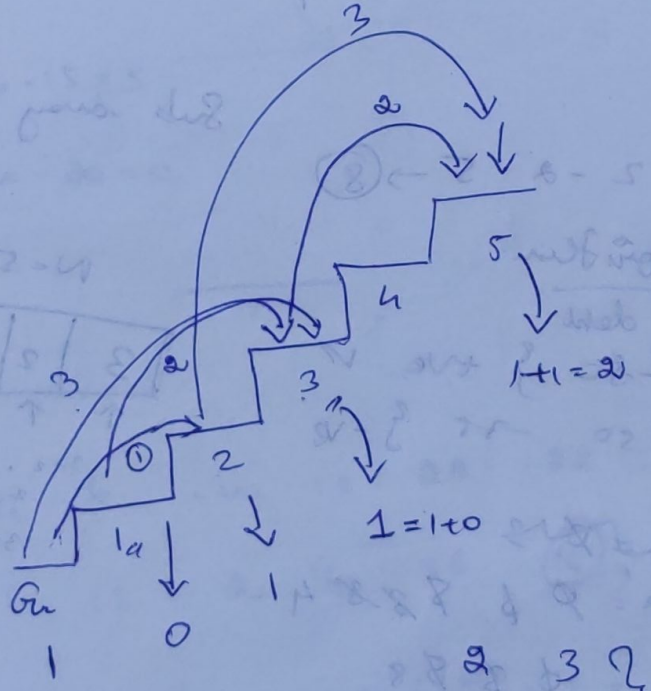
29/12/21

Staircase

Stair N = 5

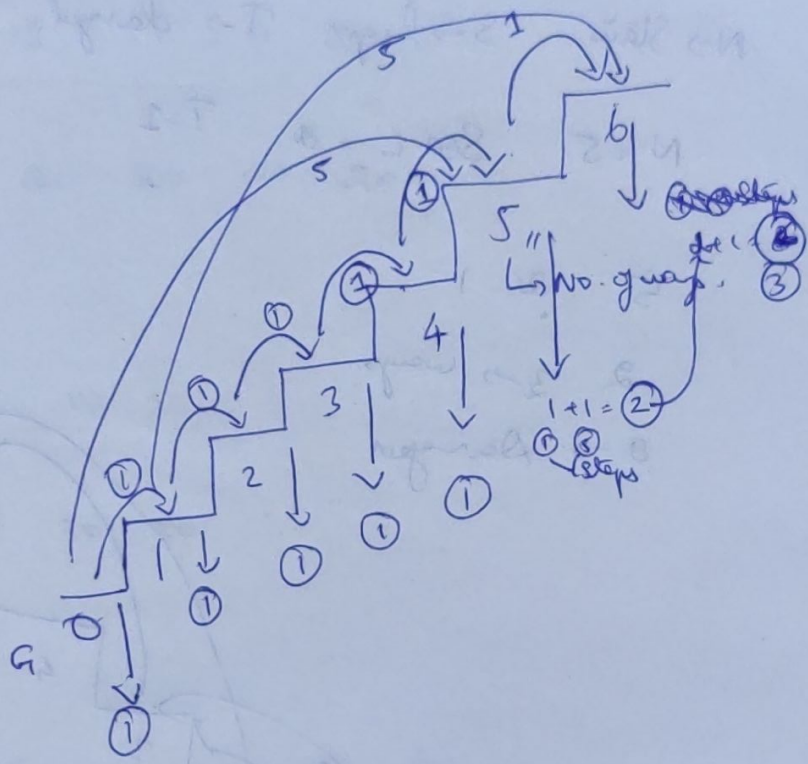
5 2

2 3



2 3 } ways to
3 2 reach (5)

6 2
1 5



N = 5 2, 3 steps analysis = ⑥

ways = 0 1 2 3 4 5

value = 1 0 1 ~~1+0~~ ~~1+0~~ ~~1+1~~
1 1 ②
↳ o/p.

N = 6 1, 5 steps

ways = 0 1 2 3 4 5 6

Value = 1 1 1 1 1 ~~1+1~~ ~~2+1~~
2 ③
↳ o/p

Pb ②

Damaged

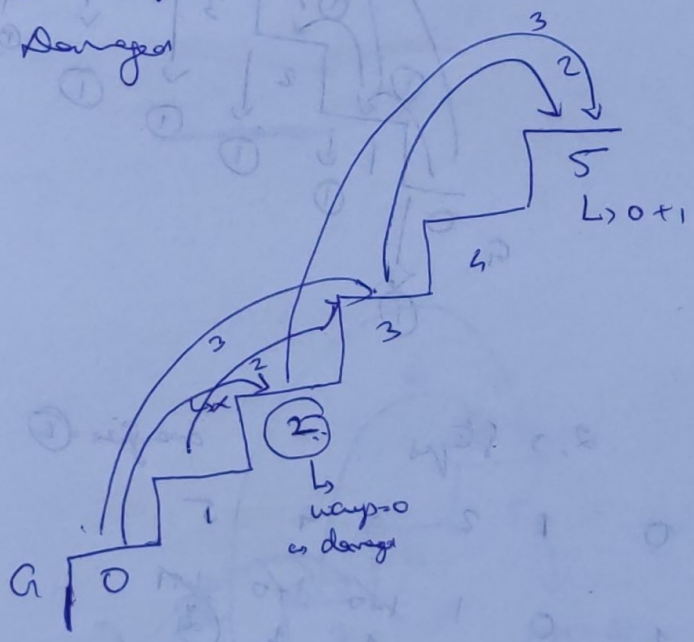
N → Stairs S → leaps T → damaged.

N = 5 So L = 2 T = 1

5 2 1

2 3 → ways

2 → Damaged



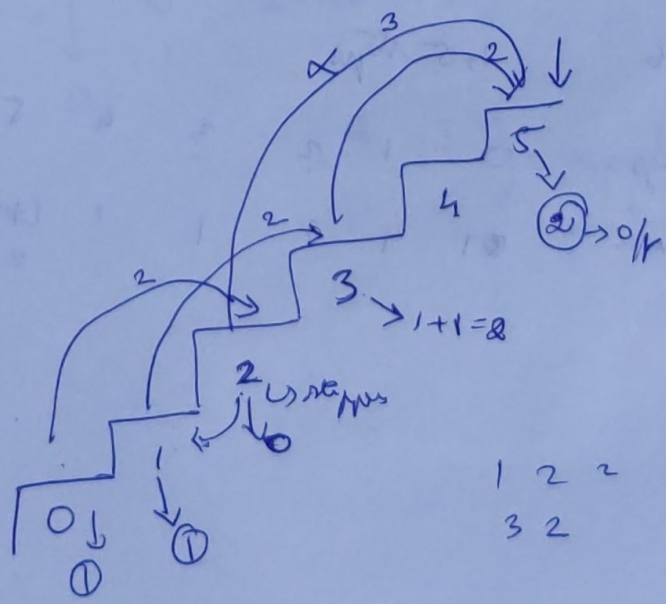
Pb ③ "slippery"

N = 5

5 2 1 → T

2 3 → leaps.

2 → slippery



Next greater element

N=7

2 1 5 15 10 6 20

o/p 5 5 15 20 20 20 20

N=4

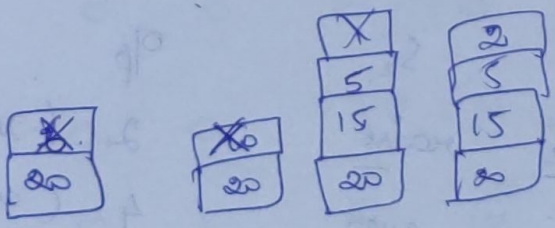
20 15 10 30

o/p 30 30 30 30

N=7

using stack: 2 1 5 15 10 6 20

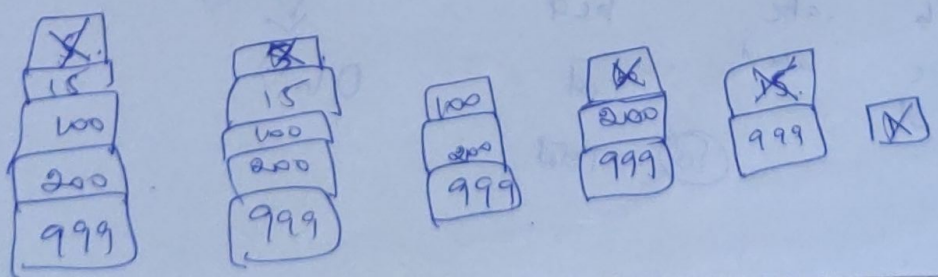
5 5 15 20 20 20 20



N=10

7 5 3 15 100 60 200 15 999 1

15 15 15 100 200 200 999 999 999 1



Remove adj value if equal.

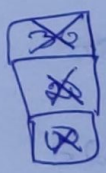
N = 9

I/p 20 15 20 30 30 40 45 50 90

O/p 20 50 90

N = 6

10 20 30 30 20 10



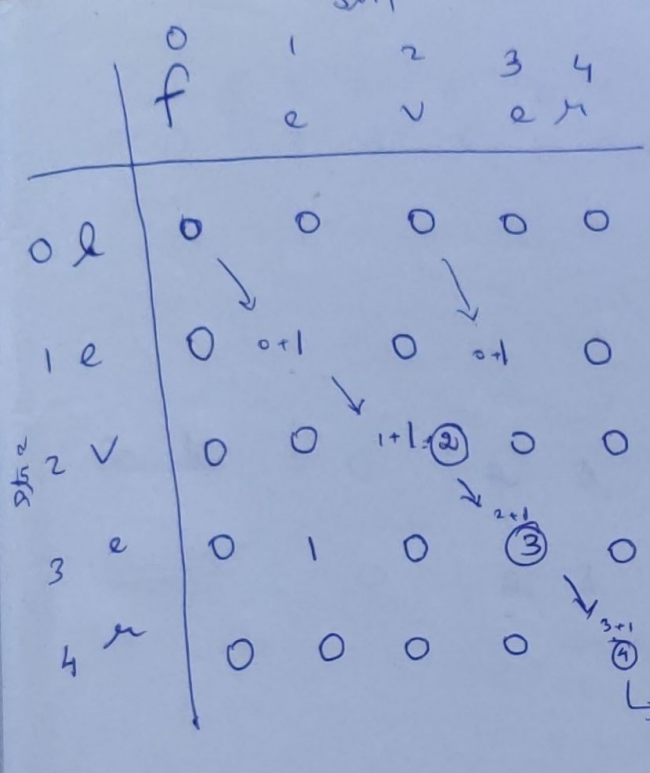
Longest Common Substring:

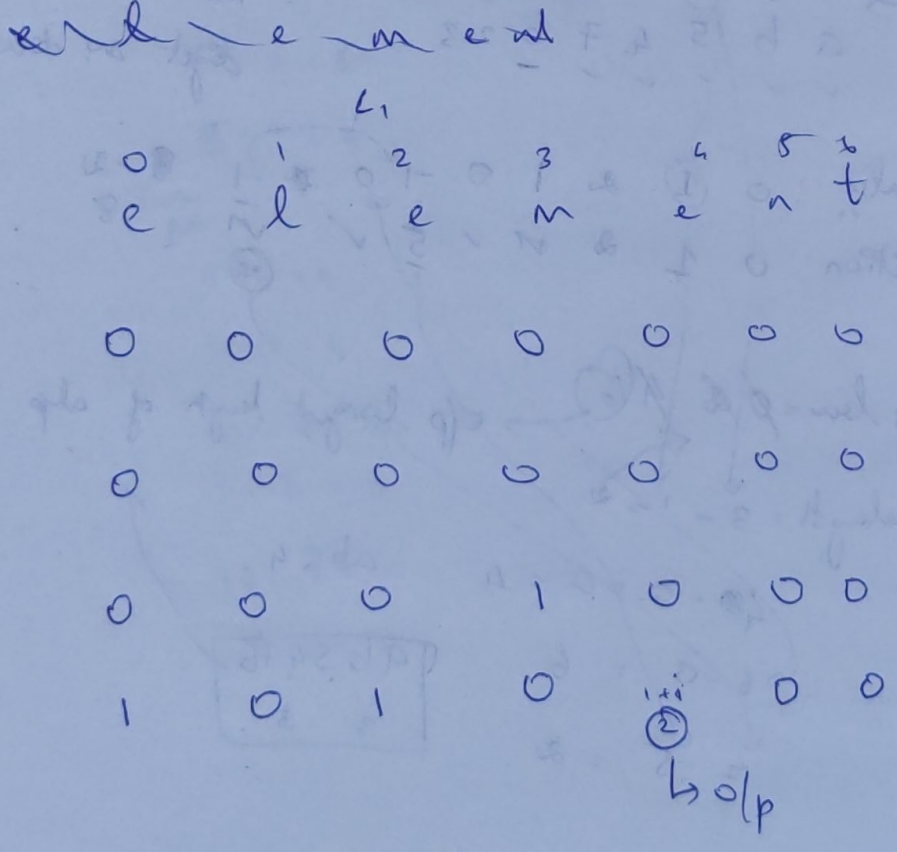
S_1	S_2	O/p
rose	raise	2 (se)
fever	ever	4 (ever)
ink	paper	0

abcd

			check	compare
a	ab	bc	$O(n^2)$	$O(n^2)$
b	abc	bcd	↓	
c	abcd	cd	$O(n^4)$	
d				

(10) → Total





$m \rightarrow 2$

$O(L_1) \times O(L_2)$

$O(N^2)$

Longest Substring

ab547b23

ab54
2 2

7b23
alp-1
3

ab547b
3
3 → alp count
↳ digit count

Position → 1 2 3 4 5 6 7 8
a b 5 4 7 b 2 3
✓ ✓ ✗ ✓ ✗ ✗

queues with alphabet digit so decrease

if alp → Counts : 0 1 2 1 0 -1 0 -1 -2
++ 8
position : 0 1 2 ✓ ✓ 5 ✓ 7

max len = 0 2 4 6 → op longest length of alp & num

Curr length = 3 - 1 = 2

4 - 0 = 4

6 - 0 = 6

7 - 5 = 2

ab54

ab547b
3 3

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	z	2	3	a	a	c	c	1	5	6	7

Counts = 0 1 2 3 2 1 2 3 4 5 4 3 2 1

0 1 2 3 ✓ ✓ ✓ ✓ 8 9 ✓ ✓ ✓

maxlen → 2 → o/p.

Comlen =

4 - 2 = 2

5 - 1 = 4

Curve

6 - 2 = 4

7 - 3 = 4

10 - 8 = 2

11 - 3 = 8

12 - 2 = 10

13 - 1 = 12

30/12/21

Spiral matrix - Clockwise

R = 6 C = 5

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25
5	26	27	28	29	30

function for printing

L → R →

T → B

R → L

B → T

StartRow = 0 1

endRow = 6 - 1 = 5 4

After first iteration do the reduction

StartCol = 0 1

endCol = 5 - 1 = 4 3

* L → R 1 2 3 4 5
* T → B 10 15 20 25 30
  1 → 5
* R → L 29 28 27 26
  3 → 0
* B → T 21 16 11 6
  4 → 1

Collect Maximum Points

10	5	30
20	100	2
5	50	40

10	15	45
30	130	132
35	180	220

max(130, 50) = 130

output

4	2	9	61	
7	9	6	54	
5	7	3	88	
7	4	99	4	

Max

4	6	15	21	22
11	20	26	31	35
16	27	30	39	47
23	30	40	49	53

output

---

2) Collect max Points from 4 5 a given cell

	0	1	2	3
0	4	2	9	61
1	7	9	6	54
2	5	7	3	88
3	7	4	9	94

(2, 3) → start point.

(1, 1)

3 3

	0	1	2
0	48	64	47
1	63	33	14
2	44	82	52

	0	1	2
0	48	112	159
1	111	145	173
2	155	237	289

o/p.

8	16	
17	21	

o/p.

9	15	20	24
16	19	28	36
20	29	38	42

o/p

Finding the Majority element in an array.

1) N = 6

5 7 5 7 5 2

$N/2 = 3 \rightarrow (4) \rightarrow$ minimum for majority element

o/p = No.

N = 9

2 3 3 3 2 2 2 2 2

$N/2 = 4 \rightarrow (5)$

majority element = 2 → o/p.

N = 6

4 3 4 3 4 4

$4 > 6/2 \rightarrow$ o/p = 4

N = 6

4 3 2 2 3 4

2.

N = 6

4 3 4 3 4 4

Starting with counter = 1 & majority element as first element of array

Max element or majority element = 4 3 4 3 4 4

counter = 1 0 1 0 1 0 1 2

$2 > 0 = (4)$ Maj element

arr element == max element

or decrease counter and replace majority element

change the counter, check counter > 0

Then print max element

Subarray Sum:-

Eg1

N = 5 S = 45

LI	↓	↓	↓	↓	
	5	10	50	20	25
RI	↑	↑	↑	↑	↑

CurrSum = 5 + 10 = 15 65 60 50 0 20 45

65 - 5

if currentSum < S

Right ++ CurrentSum + indexvalue.

else. CurrentSum > S

leftIndex ++

CurrentSum − indexvalue

31/12/21

Vertical zigzag pattern :-

N = 5

No. of rows above

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

4 rows

4 × 2 + 1 = 9 0 × 2 + 1 = 1

3 × 2 + 1 = 7 1 × 2 + 1 = 3

2 × 2 + 1 = 5 2 × 2 + 1 = 5

1 × 2 + 1 = 3 3 × 2 + 1 = 7

0 × 2 + 1 = 1 4 × 2 + 1 = 9

Down	Up
11	1
9	3
7	5
5	7
3	9
1	11

Down
2 x down row + 1 =

Up
2 x up row + 1 =

N = 6

1	12	13	24	25	36
2	11	14	23	26	35
3	10	15	22	27	34
4	9	16	21	28	33
5	8	17	20	29	32
6	7	18	19	30	31

Zigzag Reducing:-

N = 5

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

Down

$3 \times 2 + 1 = 7$

$2 \times 2 + 1 = 5$

$1 \times 2 + 1 = 3$

$0 \times 2 + 1 = 1$

Up

above / left col.

$2 - 1 = 1 \times 2 = 2$

$3 - 1 = 2 \times 2 = 4$

$4 - 1 = 3 \times 2 = 6$

$4 - 3 = 1 \times 2 = 2$

Combination

abc → (3)

1 to $2^3 - 1 = 7$

001	c
010	b
011	b c
100	a
101	a c
110	a b
111	a b c

Sub 1 with corresponding letter in string

Binary

left shift

1 << 0 → 1

Str check at position 0

1 << 1 → 10 (2)

1 << 2 → 100 (8)

1 << 3 → 1000.

Dec → consider

for ctr 1 to 7 $(2^3 - 1 = 7)$

001 and 1 → 1 → So a (a)

001 and 10 → 000 reject

001 and 100 → 000 reject

ctr = 4

100 and 001 → 000

100 and 010 → 000

100 and 100 → 100 → ac (ac)

ctr = 5

101 and 001 → 001 → a

101 and 010 → 000

101 and 100 → 100 → c

= (ac)

ctr 2

010 and 1 → 000 → So reject

010 and 10 → 010 → be (b)

010 and 100 → 000 → so reject

ctr = 3

011 and 001 → 001 → a

011 and 010 → 010 → b (append)

011 and 100 → 000 reject.

Result (ab)

Ch = 6

```
110     110     110
001     010     100
---     ---     ---
000     010     100
---     ---     ---
        b       c
```

bc

Ch = 7

```
111     111     111
001     010     100
---     ---     ---
001     010     100
---     ---     ---
a       b       c
```

ab

Result list = a, b, ab, c, ac, bc, abc

Combination two sum:

```
001         1100
 1
---
 1
---
```

Single Value Repeated Odd no of times

XOR - Exclusive OR

N

① 2 2 5 5 2 5 2 → O/p = 5

② 1 4 3 1 4 39 - O/p = 9

Truth table XOR

A	B	Result
0	0	0
0	1	1
1	0	1
1	1	0

2 ^ 2 = 0

5 2 5 2 5 → 0 5 → 101 ^ 000 = 101

Count 1s in Binary Representation of N.

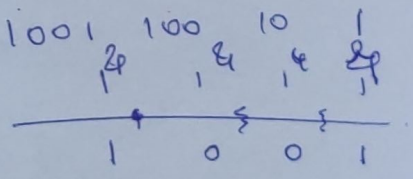
N	Binary	o/p
10	1010	2
3	11	2
8	1000	1
15	1111	4

*implementation

9	1001	

N>>1	100 → 4	
n>>2	10	

1st	n>>1	100	loop till n becomes 0
2nd	n>>1	10	
3rd	n>>1	1	
4th	n>>1	0	



counter = 0 1 2

Bit Flip Count

A	B	Flip
10	6	1 → 0
		0 → 1

Binary 1010 0110

Count = 0 1 2

XOR used,

A ^ B = C

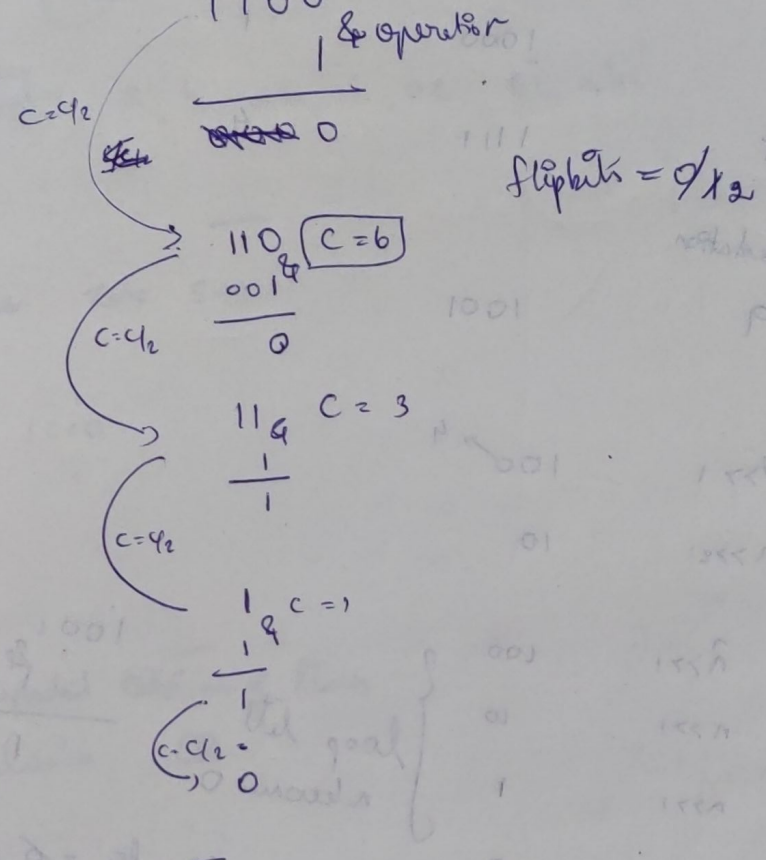
↳ count 1's in C binary

Check for 10 and 6

10 → 1010 → A

6 → 0110 → B (XOR)

1100 → C = 12 → Don't consider



flipbits = 9/12

2/11/22,,

Alphabet Count.

I/P	O/P
apple	a1e1l1p2
energy	e2g1n1r1y1

ASCII Table

a	b	c	d	e	f	g	h	i	j	K	l	m
97	98	99	100	101	102	103	104	105	106	107	108	109

n	o	p	q	r	s	t	u	v	w	x	y	z
110	111	112	113	114	115	116	117	118	119	120	121	122

arr[128]

I/p: apple.

loop a → z

o/p:

a1e1l1p2

arr[97] = 0 → 1 (a)

arr[112] = 0 → 1 → 2 (p)

arr[108] = 0 → 1 (l)

arr[101] = 0 → 1 (e)

- First Repeating Character

	O/p	
abcbaa	b	a - 3 since b is repeated 1st so b as o/p
engine	n	
cool	n.	

Implementation:

abcbaa

arr[128] → 0

arr[97] → 0 → 1

arr[98] → 0 → 1

arr[99] → 0 → 1

So print b

Recursion:

I/p N = 5

o/p 5 4 3 2 1

```
fn ( int val ) {
    val = = 0  → base or Exit condition
    print val
    fn ( val - 1)
}
```

Tail recursion

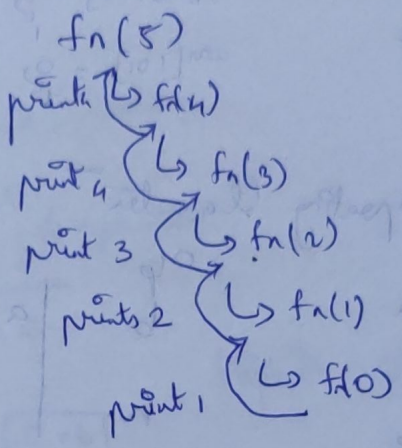
First logic then recursion.

fn(5)
print 5
↳ fn(4)
print 4
↳ fn(3)
print 3
↳ fn(2)
print(2)
↳ fn(1)
print 1
↳ fn(0).

I/P N=5 O/P 1 2 3 4 5

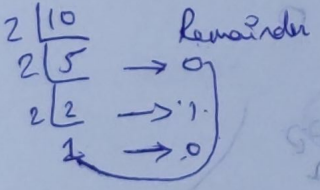
Head recursion then logic
↳ First recursion and then print

```
f(val)
{ value == 0 -> Exit
  fn(val-1) -> Head
  print(val)
}
```



Integer to Binary.

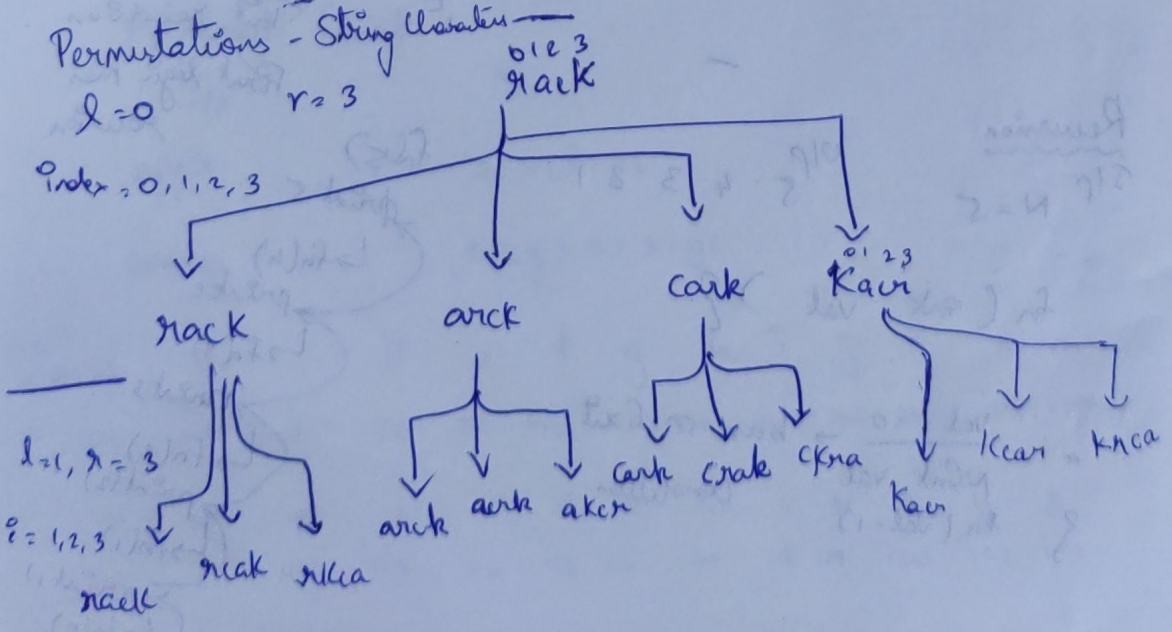
N = 10



1010

Binary = 1010

Permutations - String Characters



l = 2 r = 3

i = 2, 3.

- rack
  - rack
  - rakc
- rcak
  - rcak
  - rcka
- rkca
  - rkca
  - rkac

- arck
  - arck
  - arkc
- acrk
  - acrk
  - ackr
- akcr
  - akcr
  - akrc

- cark
  - cark
  - cakr
- crak
  - crak
  - crka
- ckra
  - ckra
  - ckar

- kacr
  - kacr
  - karc
- kcar
  - kcar
  - kcra
- krca
  - krca
  - krac

l = 3 r = 3
print all the values

N = 123 x = 200

0 difference from x

(77) — 123 213 → (13) 312 → (112)

(68) — 132 231 → (31) 321 → (121)

o/p = 213.

check in C programming
sscanf
atoi
↳ string to integer

N = 54 x = 60

54 → (6)

45 → (15)

o/p = 54

ELITE NOTES CONTINUATION:

Print Prime Number from 1 to 2

→ 1 is not a prime.

→ take 2, strike square of 2 and strike all by adding 2.

→ Take 3, find square of 3 (strike) and strike all by adding 3

HCF

Brutefore

HCF = 5

25 35

~~25~~ 26 ~~27~~ ~~28~~ ~~29~~ ~~30~~ 31 ..... ≠ 5 → GCD or HCF

take smaller number

Euclidean

Eg1:

a	b	
25	35	
35	25	(25, 35)
25	10	(35, 25)
10	5	(25, 10)
5	0	(10, 5)

GCD or HCF

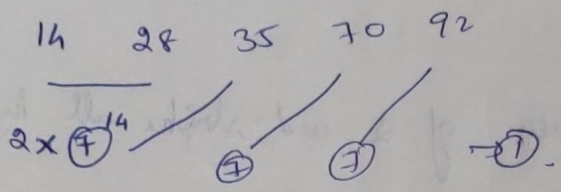
Continue till b is 0

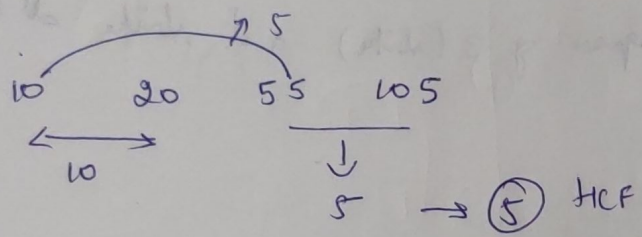
Eg2:

a	b	
20	100	
100	20	(20, 100)
20	0	(100, 20)

HCF

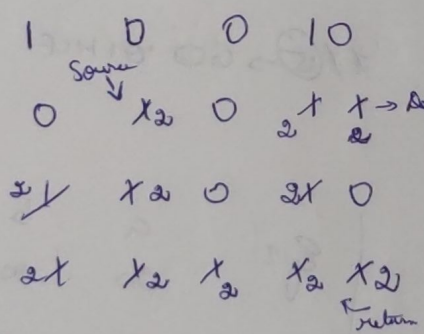
N = 5





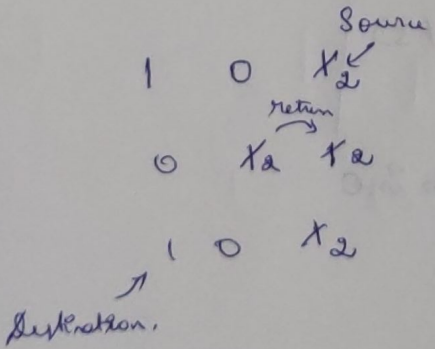
—

Depth First Search:-

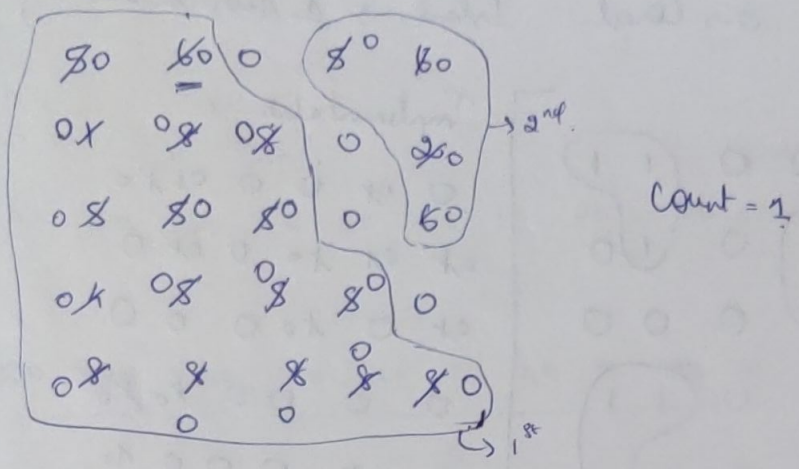


1 → Path exists
0 → cannot go.
2 → Already visited

Left
Right
Bottom
Top



R = 5 C = 5 K = 6.



Count = 1

—

Backtracking:-

Eg1:

S_1 = m n o

S_2 = x y z

S_3

x m n y z o

Yes o/p

Eg2:

m a k n o o n.

Yes o/p.

k o n

S_1 m a n o

S_2 k o n

m a k o o n n.

No o/p.

S_3.

S_1 k o o l backtrack.

S_2 b o o k

k b o o t o o l

o/p Yes.

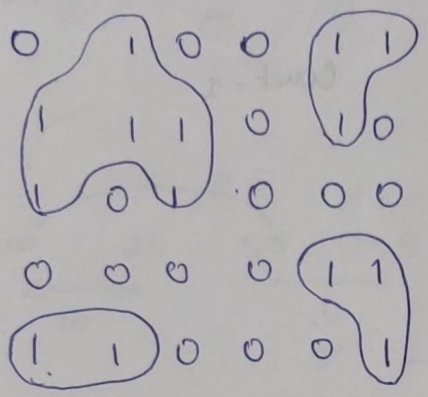
Backtracked (-k).

4/1/22

Island Count:-

1 → Land 0 → Land. Island → 1 surrounded by 0

Eg:



Implementation:

```
0 or 0 0 or x0
or or x0 0 or 0
or 0 x0 0 0 0
0 0 00 x0 x0
or or 0 0 0 x0
```

Island count = ~~21~~ 2

Word Search in matrix:

```
K e m t u n o p
r a r i n g b o w
v a r n g u r e l
r a t r o n g h
h w y t r x o g
```

Searchword: ringtone

(backtrack)

Array rotation left. Rotate Reversal Algorithm:-

N = 4.

R = 1000002

R = R % N = 1000002 % 4 = 2

So 2 rotation is enough

	10	20	30	40
1st	20	30	40	10
2nd	30	40	10	20
3rd	40	10	20	30
4th	10	20	30	40

After rotation of N times the array is same as the original time

N = 100000 R = 99999.

R % N =

N = 10 R = 7

10 20 30 40 50 60 70 80 90 100

Steps

1) 100 90 80 70 60 50 40 30 20 10 → Reverse array

2) 80 90 100 10 20 30 40 50 60 70. → rotate N - R = 3, first rotate then rotate remaining

Rotate right:

N = 10 R = 7

10 20 30 40 50 60 70 80 90 100

Steps

1) 100 90 80 70 60 50 40 30 20 10 → Reverse array

2) 40 50 60 70 80 90 100 10 20 30 → rotate R times, first rotate then rotate remaining

Left shift - N - R elements

Right shift R elements

N = 5 Swaps = 02

O/p = 1 2 3 4 5

I/p = 1 4 2 3 5
✓

Cycle = 1 & 3

2nd = 4 — 3
\ |
2

Swaps = Cycle − 1
= 3 − 1
= 2

1st swap 1 2 4 3 5

2nd swap 1 2 3 4 5 ✓

Eg 2:

1 2 3 4 5 6 7 8 9 10 11

2 1 4 3 6 5 8 7 10 11 9

2 − 1 = 2 − 1 = 1

4 − 3 = 2 − 1 = 1

8 − 7 = 2 − 1 = 1

6 − 5 = 2 − 1 = 1

10 — 11 = 3 − 1 = 2
\ |
9

Swaps = 0 & 1 & 1 & 1 & 2

5/1/22

Greedy Approach:

Minimum Operation - Zero to N:

N = 5 count = 0 1 2 3 4

+1 or *2.

0 +1 = 1 +1 = 2 *2 = 4 +1 = 5 ||

N = 8 count = 0 1 2 3 4

+1 or *2

0 +1 = 1 *2 = 2 *2 = 4 *2 = 8 ||.

N = 7

+1 OR *2

0 1 2 4 5 6 7

+1 +1 *2 +1 +1 +1.

count = 0 1 2 3 4 5 6

0 1 2 3 6 7

+1 +1 +1 *2 +1.

count = 0 1 2 3 4 5

5 is minimum, to avoid confusion, we use greedy approach. (checking from reverse)

Greedy approach

N = 7

+1 *2

-1 /2 → reverse operation

7 6 3 2

-1 /2 -1

count = 0 1 2 3 +2 = 5

0 → 2 = min 2 operation required (so adding 2)

If N is odd subtract = 1

If N is even divide by 2

Minimum Operation X to Y

Eg: 1 x y

5 → 8

*2 -1.

Reverse: → greedy.

8 → 5 count = 0 1 2

/2 +1 → inverted operation

8/2 = 4

4 +1 = 5

Eg: 2 10 → 1

×2 -1.

1 → 10

/2 +1

As 1 is odd, and we cannot divide 1 by 2 so add 1 nine times

count = 9 ||

Eg3:

4 → 31

×2 -1

31 → 4

/2 +1

count = 0 1 2 3 4

32/2 = 16/2 = 8/2 = 4 ✓

If N is odd, add 1

if N is even, divide by 2

Eg4:

4 → 35

35 → 4

count = 1 2 3 4 5 6 7

36/2 = 18/2 = 9 +1 = 10 = 10/2 = 5 +1

= 6/2 = 3

X Y

40 → 5

*2 -1

5 → 40

/2 +1

3 +1 = 4

if source > destination

we can directly subtract X - Y

Max Sum – K * K Sub Matrix

R
4 5 C

①	10	20	80	40	55
	90	50	② 90	200	65
	60	20	5	20	12
	10	50	40	60	8

K → 3

4 6

0	①	10	30 (0+20)	110	150	205
0		90	140 ② (90+50)	230	430	495
0		60	80	85	105	117
0		10	60	100	160	168

⇒ ① 110 − 0 + 230 − 0 + 85 − 0 = 425

⇒ ② = 430 − 90 + 105 − 60 + 160 − 10

= 340 + 45 + 150

= 485 + 50 = 535

Longest Substring Length:

End ↓ ↓ ↓ ↓ ↓ ↓ ↓ | K = 2

m i n n o n

Start ↑ ↑ ↑

unique = ~~1 2~~ 2

Max len = ~~0 2~~ 1

↳ o/p

m → 1 0

i → 1 0

n → 1 3

o → 1

6/1/22

Stock Buy & Sell Once = Max Profit

N = 7

50 100 40 60 70 50 80

↑

Min price = ~~50~~ 40

Max profit = ~~0~~ (50) → O/P

(100 - 50)

15 10 60 70 45 5 70 30 60 90

↑

minPrice = ~~15~~ ~~10~~ 5

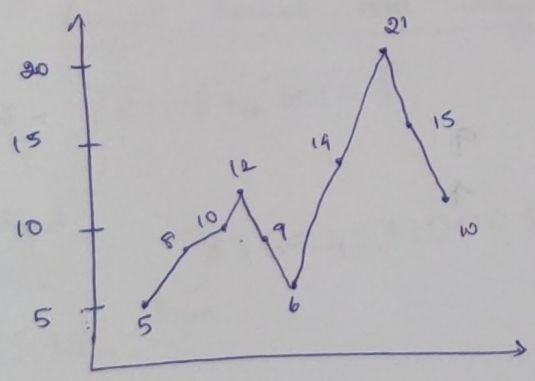
max profit = ~~0~~ ~~50~~ ~~60~~ ~~65~~ (95) → O/P.

Stock Buy & Sell Multiple times - MaxProfit

N = 10

5 8 10 12 9 6 14 21 15 10

↑



12 - 5 + 21 - 6 = 7 + 15

= 22.

8 > 5, so

Profit = ~~0~~ ~~8~~ ↓ ~~15~~ ~~7~~ ~~15~~ 22

(8 - 5) (10 - 8)

Non-Measurable smallest Weight.

Kgs = 1 2 1 5

1 = 1

2 = 2 or 1+1

3 = 2, 1

4 = 1+2+1

5 = 5

6 = 5+1

7 = 5+2, 5+1+1

8 = 5+1+2,

9 = 5+1+2+1

10 = Cannot be Measured.

N = 5

Kg = 1 1 2 9 5

Sort the weight

1 1 2 5 9

↑ ↑ ↑ ↑ ↑

M = ~~1~~ ~~2~~ ~~3~~ ~~5~~ ~~10~~ 19

1 to 2 3 ←

1 to 2

1+2 = 3

2+~~2~~ = 4

5 -

1+5 = 6

2+5 = 7

3+5 = 8

4+5 = 9 //

↵

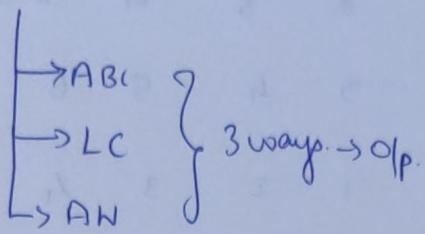
Till 9Kg confirmed that Can be measured

Decode Ways:

1 2 3

1 - A
2 - B
3 - C
⋮
26 - Z

1 2 3



1 2 3
↑ ↑ ↑

23 ≤ 26

12 ≤ 26

ways = 1 2
(ways + backup)
PrevWays = 1 2
(prevways=ways)
backup = 1
(backing up previous ways)

07/01/22. ## Largest Possible odd Integer:

N = 1 2 0 0 8 7 4 6 0 1 5 3

Largest odd.

1, 3, 5, 7, 9

8 7 6 5 4 3 2 1 0 0 0 1
↙ largest.
↓ smallest odd.

Matrix Zig-zag from Top-left:-

3 7

1	2	3	4	5	6	7
8	9	1	2	3	4	5
6	7	8	9	1	2	3

3 7

Direction = 1 -1

	Col → 0	1	2	3	4	5	6
Row → 0	-1	-2	3	4	5	6	7
1	-8	9	1	2	3	4	5
2	6	7	8	9	1	2	3

Boundary:

0 - R-1

0 - C-1

row = 0 + 1 * 2

col = 0 + 1 - 1