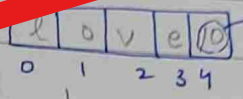
Char Arrays and Strings (Live)

Arrays and Strings - Class-1 — 3 June 24

- char arr[];

creation → array → sequence of characters
→ char arr[10];

eg: char name[5];



→ null character

acts as terminator

access: name[

i/p: cin >> arr[0] → cin >> arr → love

o/p: cout << arr[0] → cout << arr → love

- ASCII value of null — zero

a → ASCII 97 → z → o

A → 65 → Z → o

charArray
→ delimiter → enter or '\n'
→ tab or '\t'
→ space " "

- cin.getline(arr, 100);
  - → isse space ya tab se delimit nhi hoga sentence
  - size of char to be stored

eg: (arr, 2):
i/p: love Babbar
o/p → love Ba

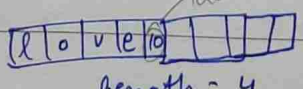
// multiple line input

- cin.getline(arr, 100, '\t');
  - → jese hi tab ayega, tab hi delimit ho jayega

delimeter: jese hi delimiter ayega, input lena band ho jayega.

Q. find length of char array.

char arr[10];
cin > arr; → love



length = 4

TC: O(n)

Q. Replace character. '@' → '_'

i/p : My@Name@is@ ...

o/p : My_Name_is_Samyak_Yadav

TC: O(n)

SC: O(1)

Q. upper → Lower case

Lower → upper case

A→65 ... Z

B→66 ... Z

a→97

b→98

c→99 ... z

⇒ upper → lower

ch = ch + 'a'

= 'A' - 'A' + 'a'

= 'a'

⇒ lower → upper

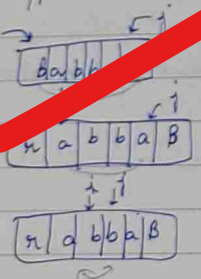
ch = ch - 'a' + 'A'

Q. Reverse char array

eg: i/p : LOVE

o/p : EVOL

TC: O(n)



Palindrome (same from left and right)

i/p → word

o/p → T/F (tell if its palindrome)

if (arr[i] == arr[j]) i++; j--;

else → false

Append two char arrays

cout << strcat(arr1, arr2);

(hw)

a) strlen ✗ but we won't use this

b) strcat

c) ⋮

string

↳ dynamic, safe (more safe than char arrays to store sequence of characters)

string name;

access name[3]; or name.at(3);

name.push_back('a'); → insertion

name.pop_back(); → deletion.

cin >> name;

cout << name;

↳ yaha bhi default deliniter space hai.

getline(cin, arr_name, deliniter); ← optional

Member Functions : at(), front(), back(), length(), begin(), end(), clear(), empty(), find(), concat

substr(starting_index, length) ← optional

a) if (name.find(word) != string::npos) → word not found

b) concat : s1 + s2.

c) compare : s1.compare(s2); → gives 0 if s1 & s2 are same else non-zero value

Char Arrays and Strings class-2 (Live) (4 June 24)

Leetcode 1047.

↳ Q. Remove all adjacent duplicate in a string.

i/p → string → "abbaca" abbaca
o/p → ca ← aaca

eg: a z x x z y
a z z y
a y

str → abbaca ans = "_" same → pop back
→ "a"
→ "ab" rightmost
→ "a" different char → push back
"c"
"ca"

extra case → when ans is empty → sidha push back without any condition.

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

Leetcode 1209. (hw) → fol" in week-5 Past video)

Leetcode 1910. Remove all occurrences of a substring.

i/p → daabcbaabcbc"
pattern → "abc" (to be removed)
o/p → ~~daba~~ dabaabc
→ dab

→ while (s.find()) { → shifting
erase O(n-K)
}

K-length pattern
n length string
n/K times loop chlega

~~TC: O(n(n-K))~~ TC: O((n/K)(n-K))

Leetcode 680. Valid Palindrome II

"abca"
i j

same: if s[i] == s[j]
↳ i++
↳ j--

delete i → aca
↳ ans → true
delete j → aba
↳ ans → true

return fAns || secondAns

0	1	2	3	4	5	6
~~r~~	a	b	b	c	a	r
i	i	i	j	j	j	

bbc
i j
delete i → bc → false
delete j → bb → true
or → final Ans → True

[only one time deletion is allowed]

TC: O(n)

Leetcode 539. (Raw) → sol" in assignment → see if tried for 1 hr and couldn't do.

Leetcode 647. Palindromic Substring.

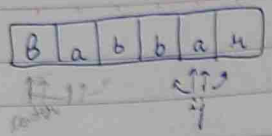
(bdia ques) abc → a
b
c
ab
bc
abc

substring should be continuous
↳ subtype of the topic subseq

Palindrome → odd length
→ even length

expand around center

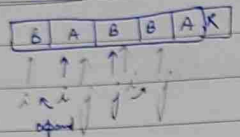
i j



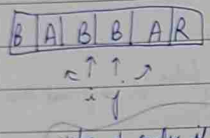
B
a
Bab
b
abb
Babba
bba
abbaa
a
baa
a

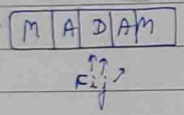
add length substr

even length substr



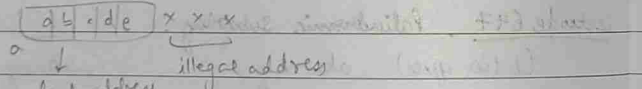
B
AB
BABB
BB
ABBA
BABBAR
BA
BBAR
AR





TC: O(n²)
SC: O(1)

Q.
eg: char a[5];
cin>>a;

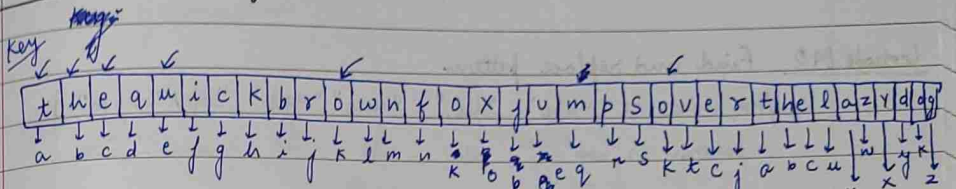


⇒ g++ address sanitizer filename.cpp
⇓
illegal address wali baate dega

Char Arrays & Strings class-3 (Live) 5 June 2024

~~leetcode 890. Find and replace pattern.~~

leetcode 2325. Decode the msg

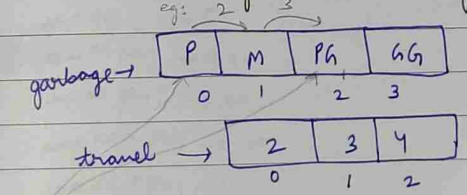


message: v K b s b s t s u e p u v
this is a secret

steps
① create mapping
② use mapping to decode msg.

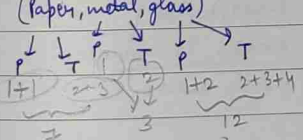
~~TC: O(m+n)~~
TC: O(max(m,n)) → linear TC
SC: constant space

leetcode 2391. Minimum amount of time to collect garbage.

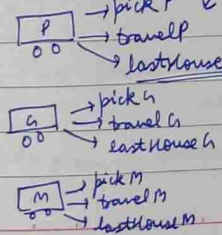


total time
⇓
picking + travel time
(P) (T)

(paper, metal, glass)



⇒ har truck ka picking and travel time nikal ke & add krdo



→ to get travel time

(Brute force)
TC: O(n) → linear
SC: constant

second loop ko neglect krdia (size = 10 lac max is like)

Leetcode 791. Custom Sort String. use comparator

TC: nlogn

Leetcode 2125. (HW) → weak-5 assignment (hard ques)

Leetcode 890 Find and replace pattern

i/p → words → ["abc" "deq" "mee" "aqq" "dkd" "ccc"]

pattern → "abb" abc deq mee aqq dkd ccc

abb abb aba aaa

(mee ✓)

- Normalize the pattern
- → then check if they are equal to pattern or not

BASIC MATHS and POINTERS (Live)

Pointers- class-1 (10 June 2024)

Variable : named memory location

→ int a=5;

→ store address of another variable → POINTERS

104
a

Symbol table

a → 104

cout<<a ; ⇒ 5

cout<<&a; ⇒ print address → 104

⇒ cannot store address in int (address ko POINTER me store karenge)

eg : int c=&a; ⇒ gives error.

syntax:

int * ptr = &a;

- int * : pointer to an integer
- ptr : variable name
- &a : a ka address

104 5 a

420 104 ptr

cout<<a; → 5

cout<<&a; → 104

cout<<ptr; → 104

cout<<&ptr; → 420

cout<<*ptr; → 5

DEREFERENCE OPERATOR

value at address stored in ptr

→ value at 104 ⇒ 5

declaration: int *ptr;

BAD PRACTICE

Q Differences :

→ reference variable → int a=5; int &b=a

→ pointer → int *b=&a

(HW) difference?

HW → read articles given in pointers section

Q. `int a=5` `int *ptr=&a;` `sizeof(ptr)` → 8 (store address is fixed)

`char ch='a';` `char *cptr=&ch;` `sizeof(cptr)` → 8

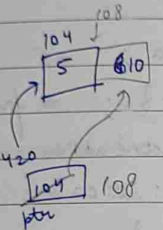
`long *l=1040;` `long *lptr=&l;` `sizeof(lptr)` → 8

* `int *ptr;` ✗

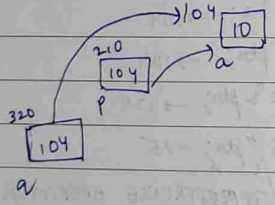
↳ create null pointer

* `int *ptr=0;`

eg:
```
int a=5;
int *ptr=&a
```
→ a → 5
→ &a → 104
→ *a → error
→ ptr → 104
→ &ptr → 420
→ *ptr → 5
→ ptr=ptr+1 → 108
→ *ptr=*ptr/2 → 10/2 = 5
→ *ptr=*ptr+1 → 10+1=11



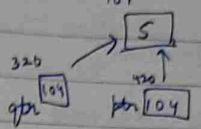
eg:
```
int a=10;
int *p=&a;
int *q=p;
```
→ copy



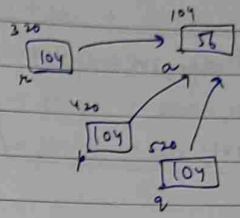
a → 10
&a → 104
*a → error (int ke upar reference Kaise 😊)
p → 104
&p → 210
*p → 10
q → 104
&q → 320
*q → 10

Pointer copy

```
int a=5;
int *ptr=&a;
int *qtr=ptr;
```

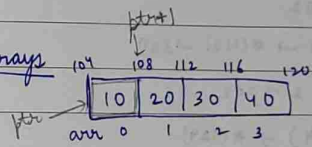


eg:
```
int a=56
int *p=&a;
int *q=p;
int *r=q;
```



r : 104
&r : 320
*r : 56

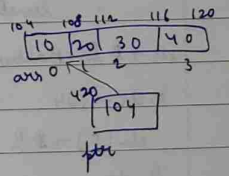
Pointer with arrays



arr → Base Address → 104
arr[0] → 10
&arr[0] → 104
&arr → 104 (give base address)

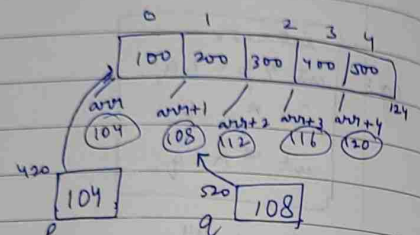
* `int *ptr=arr;`
`ptr=ptr+1` ~~`arr=arr+1`~~

eg:
```
int arr[]={10,20,30,40}
int *ptr=arr
```
→ arr → 104
→ &arr → 104
→ arr[0] → 10
→ &arr[0] → 104
→ ptr → 104
→ &ptr → 420
→ *ptr → 10
→ *arr+1 → 10+1=11
→ *(arr+1) → *(104+1) → *108 = 20
→ *(arr+3) → *(104+3) → *(104+3X4) → *116 = 40



catch
arr[i] = i[arr]
↓ ↓
*(arr+i) *(i+arr)
⇒ arr[2] → *(arr+2)

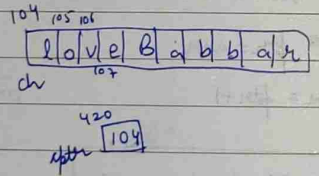
eg: int arr[] = {100, 200, 300, 400, 500};
int *p = arr;
int *q = arr+1;



q → 108
&q → 520
*q → 200
*p+1 → 100+1 → 101
*(p)+2 → 100+2 → 102
*(p+2) → *(104+2) → *(112) → 300
*(q+2) → [illegible] → 200+2 → 202
*(q+4) → *(108+4) → *(124)

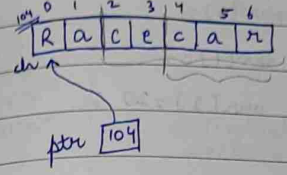
Char arrays

char ch[100] = "loveBabbar";
char *cptr = ch;
cout << cptr; → ?
↓ 104
loveBabbar



ch → B loveBabbar
&ch → 104
ch[0] → B l
&cptr → 420
*cptr → l (check it) ✓
*(cptr+3) → e (") ✓
cptr → 104 (") → but prints loveBabbar

eg: char ch[] = "Raccecar"
char *ptr = &ch[0]



→ ch → raccecar
→ &ch → 104
→ *(ch+3) → ch[3] → e
→ ptr → raccecar
→ *ptr → ptr[0] → r
→ *(ptr+3) → ptr[3] → e
→ ptr+2 → ccecar
→ ptr+4 → car

eg: char *ptr = "babbar";
↳ bad practice

(Live) 17 June 2024 ~~17 June 2024~~

Pointers - Class 2

1) array is taken as pointer in function.

```
main(){
    int arr[] = {10,20,30}
    int n=3;
    cout<<arr[0]<<arr[1]<<arr[2];
    solve(arr,n)
    cout<<arr[0]<<arr[1]<<arr[2];
    return 0;
}
```

```
solve(int arr[], int size){
    arr[0]=1
    arr[1]=2
    arr[2]=20;
    *(arr+1)=2
    *(108)=2
}
```

int *arr → pointer

*(arr+0)=1
*(104)=1
*(112)=20

Multilevel Pointers / Double pointers

```
int a=5;
int *ptr=&a;
int *q=ptr;
```

1 level pointer

int *q = &ptr ; → error

pointer ka address is tarah nhi lete.

```
int **q=&ptr;
int ***r=&q; and so on....
```

double level pointer

→ int * → pointer to an integer

→ int** → pointer to a integer*

pointer to [pointer to an integer]

q1:
```
int a=5;
int *p=&a;
int *q=p;
```

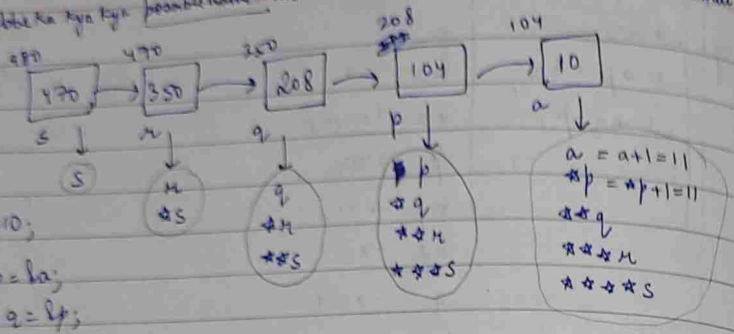
a → 5
&a → 104
*a → error
p → 104
&p → 212
*p → 5
q → 104
&q → 316
*q → 5
**q → error

q:
```
int a=10;
int *p=&a;
int **q=&p;
int ***r=q;
int ****s=&q
```

a → 10
&a → 104
*a → error
p → 104
*p → 10
&p → 212
**p → error (2 ko liye 2 level age nhi jaa payenge)
q → 212
&q → 316
*q → 104
**q → 10
***q → error
r → 212
**r → 10
s → 316
**s → 104
***s → 10 (3 bar age jana hai)
*s → 212

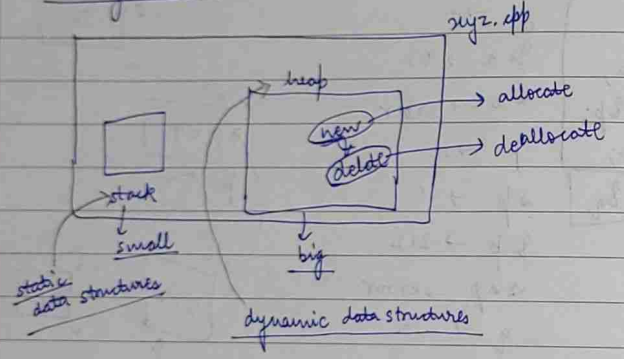
Har ek variable ka kya kya possible name hota hai or use kaise manipulate kr skte hai

eg:




```
int a=10;
int *p = &a;
int **q = &p;
int ***m = &q;
int ****s = &m;
```

Memory allocation



heap memory

stack memory : int a;

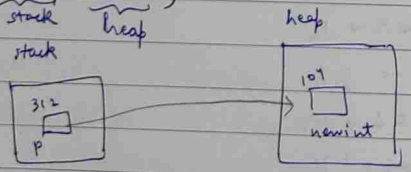
⇒ new int;

↓

return address (stored in pointer)

⇒ int *p = new int;

stack heap

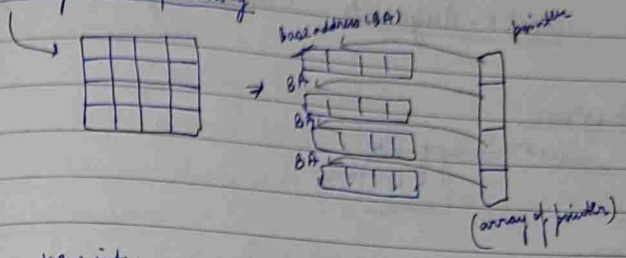


⇒ delete p; (stack me delete krne ki need nhi padhti)

delete[] p;

↳ array

2D array → heap memory

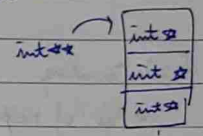


new int;

new int [10]; → array of integer of size 10

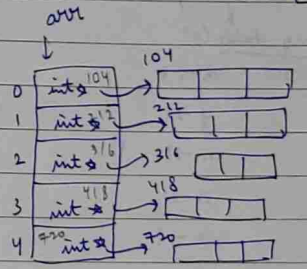
new int *[10]; → array of *int* of size 10

→ array of (pointer to integer) of size 10.



eg:

```
int **arr = new int *[5];
for (int i=0; i<5; i++){
    arr[i] = new int[3];
}
```



i=0

arr[0] arr[1] arr[2]

arr[3] arr[4]

HW

* Memory leak : occurs in C++ when programmers allocate memory by using new keywork and forgets to deallocate the memory by using delete[].

[redacted] Week 5 - Assignment [redacted] Strings]

Leetcode 242.

① Valid Anagram

word → s → 'anagram'
⇓ rearrange
t → nagaram

(M1) Brute force / sorting ['a' → 97 ; 'b' → 98 , ...]

s → aaagmnr
t → aaagmnr

(s == t) ? ⇒ return true

TC: O(n log n)

(M2) Counting

→ no. of respective chars in 's' string should be same as that of 't'.

freq table (s)	freq table (t)
a → 3	a → 3
g → 1	g → 1
m → 1	m → 1
n → 1	n → 1
r → 1	r → 1

• Table freq (s) = freq (t) → return true

eg : s = rat t = car

s	t
a → 1	a → 1
r → 1	c = 1
t → 1	r = 1
c = 0	t = 0

return false

• How to make frequency table for strings?

ASCII → 256

0	0	0	0	0	~~0~~ 1 3	...	0	0	0

0 ... 97 → a ... 255

anagram → 97

TC: O(n + m)

• instead of making 2 freq table, iterate through freq table of s and --
count = 0 vahi anagram hai

Leetcode 917

② Reverse Only Letters

~~s = ab-cd~~

s → Test1ng-Leet=code-Q!

simple reverse → !Q-edoc=teeL-gn1tseT

but ans should be → Qedo1ct-eeLg=ntse-T!

• Two pointer approach ✓

→ l = 0 ; h → s.size() - 1 ;

if (s[l] → alphabet & s[h] → alphabet) { swap(s[l], s[h]), l++, h--; }

else if (s[l] → not alphabet) { l++; }

else { h--; }

① a - z → [97 - 122]
A - Z → [65 - 90]

OR

if (ch ≥ 97 && ch <= 122) || (ch ≥ 65 && ch ≤ 90) { //alphabet }

② • C++ fxn : isalpha(ch);
↳ True → (✓)
↳ false → (alphabet ✗)

TC : O(n/2) → O(n)

Leetcode 14

③ Longest Common Prefix

① ["flower", "flow", "flight"]
0 1 2

ans = fl

curr-ch = 'f' ← strs[0][0] i

strs[1][0] = 'f' ✓

strs[2][0] = 'f' ✓

strs[0][1] = l

strs[1][1] = l ✓

strs[2][1] = l ✓

strs[2][2] = 0 ✗

Leetcode 345.

④ Reverse vowels of a string

vowels : aeiou

eg : hello
o/p → holle

eg : leetcode
o/p → leotcede

leetcode 205.

(5) Isomorphic Strings

eg: egg → s t → add ✓

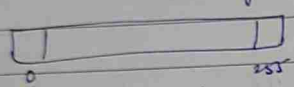
map ↳ e → a
g → d

eg: s → foo t → bar ✗ not isomorphic

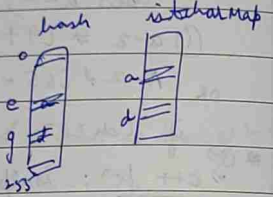
map ↳ f → b
o → a

sol^n ⇒ s t

hash → [256] ↳ array



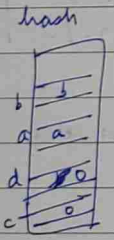
↳ index → character ✓

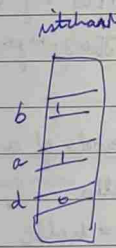


① s → iterate
hash[s[i]] = t[i]
isTokenChar[t[i]] = 1

if (successful mapping) ⇒ hash : using hash i can make 'add' from 'egg'

eg: s ⇒ badc t = baba





leetcode 767.

(6) (Imp) Reorganize strings

eg: s ⇒ aab
↳ aba ✓

eg: aaabc
↳ abaca ✓

eg: s ⇒ aaab
↳ abaa ✗ ; aaba ✗

(M1) using Priority queue TC: O(nlogn)

(M2) greedy
① most occurrent character ko fit it non-adjacently in one go.
② fill the rest of "

eg: aaabc
↳ a:3, b:1, c:1

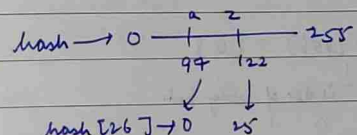
a b a c a
0 1 2 3 4

eg: aaabbcf

a b a c a f b
0 1 2 3 4 5 6

:) When I am placing most occurrent char, it should be placed in one go. (ghum ke vapas wli jgh pe → adjacent ko jayega warna)

① Count hash
② Try to place most occurrent char in one go.
if (not possible)
↳ return ""
③ place other characters with i index gap.



TC: O(n)

[s[i]-'a'] → 0
↓
97-97 → 0

leetcode 49.

(7) Group Anagrams

A B
↓ ↓
no. of character: same
& no. of occurrence:

strs = ["eat", "Tea", "tan", ---]
grp → ['bat'] ['nat', 'tan'], ['ate', 'eat', 'tea']
A B C

Method → identify anagrams wrt each string

(M1) A B

if (Anagram) { sortA, sortB ; A = B ; }

sort: nat → ant
tan → ant

i/p ⇒ cat, tea, tan, ate, nat, bat

sort ⇒ act, ait, ant, ant, ant, abt

group group group group

create map ⇒

Key	value (vector)
act	cat, tea, ate
ant	tan, nat
abt	bat

TC: O(n K log K)

SC: O(nK) → length of largest string (max length)

(M2) anagram → hash[256] = {0}

↳ index of array hash

A B

hashmap = hashmap

0 : 65 → A → 4 (occurrence)
97 → a → 5

hash Key	vector string
① e-1, a-1, t-1	cat, tea, ate
② t-1, a-1, n-1	tan, nat
③ b-1, a-1, t-1	bat

TC: O(n × K)

SC: O(n × K)

Leetcode 5.

(8) Longest Palindromic Substring

babad

i=0 ⇒ b → ✓ j=0
ba → j=1
bab → ✓ → ans (longest) j=2
baba → j=3
babad → j=4

a ✓
ab
aba → ✓ → ans
abad

① substring of string
② extract palindromic ones
③ ↳ max length string from all palindromic strings.

TC: O(n³)
: O(n) × O(n) × O(n)
loop palindrome fn

SC: O(1)

Leetcode 28.

(9) Find the index of First Occurrence in a String

haystack = 'sadbutsad'
needle = 'sad'

s a d b u t s a d
0 1 2 3 4 5 6 7 8

0 index

haystack = letsadu
needle = sad

sliding window method — compare till n-m only agar needle exist krti hogi toh miljayegi

→ outerloop ⇒ i=0 → haystack → size
innerloop ⇒ j=0 → needle
if (needle[j] == haystack[i+j]) ⇒ j++
else break;

→ if (j == m) { size of needle return i }

① i=0 letsadu sad j=0 ×
② i=1 letsadu sad j=0 ×
③ i=3 letsadu sad j=0 ✓

TC: O((n-m+1) * m)
Tc: O(mn)

leetcode 8.

(10) String to an Integer (atoi)

string = s = "1234"

integer → 1234

c++ std fn → atoi ⇒ atoi [char *i] ⇒ gives integer

① "----42" → "42" → 42 → leading space : ignore

② "-42" → -42 → sign determine

③ "---123hello" → ignore

④ "4193 hello123" → 4193 (ignore)

if digit found → int ✓
non digit found → return;

⑤ "--lak4139" → 0

⑥ special case ⇒ out of range no.

leetcode 443.

(11) String Compression

"aabbccd" → "a2b2c3d"

0 1 2 3 4 5 6

⇒ a a b b c d e e

0 1 2 3 4 5 6 7

previous, i

index = 0

```
if (s[i] == prev) { count++; }
else {
    s[index] = prev;
    if (count > 1) {
        s[index] = count;
    }
} count = 1;
```

leetcode 12

(12) Integer to Romans (asked in many companies) → simple soln hai iska

27 → XXVII (larger to smaller symbol)

10+10+5+1+1 → 27

But 4 → IIII ✗

↳ IV (smaller to larger symbol)

M → 1000
D → 500
C → 100
L → 50
X → 10
V → 5
I → 1

Special entries:
CM → 900
IV → 4
CD → 400
XC → 90
XL → 40
IX → 9

int → roman

eg : 58 if (num >= roman value)

58 > 50

⇒ 50 + 8

⇒ L → num = 58 - 50

num = 8

num ≥ roman value

⇒ 8 ≥ 5

⇒ 5 + 3

⇒ V + 3 → num = 8 - 5

num = 3

3 > 1 ? ⇒ I + I + I

⇒ L V III

```
while (num > 1) {
    // add symbol
    num = num - value;
}
```

eg : 3 3 3 3

↳ ① 1000 + 1000 + 1000 + 333 → MMM + 333

num = 3333 - 3000 = 333

② 333 → C+C+C+33 → CCC + 33

num = 333 - 300 = 33

③ 33 → XXX + 3

num = 33 - 30 = 3

④ 3 → III

ans ⇒ MMMCCCXXXIII

eg : 1994

↳ ① 1000 + 994 → num = 1994 - 1000 = 994

② 994 → MDX

↳ MCMXCIV

algorithm → Roman map

② map iterate (larger to smaller)

```
while (num > value[i]) {
    roman += symbol;
    num = num - value[i];
}
```

Leetcode

⑬ Zig-Zag Conversion

str = A B C D E F G, numRows → 3

output → A E I → AEI

B D F H J → BDFHJ

C G → CG

numRows → 1

A C E G I

B D F H

① numRows = 1 → return original string

② change direction after some time.

A E I (numRows-1)

B D F H

C G

↓ numRows

① Top to Bottom

② BT & TB switch after every (numRows-1)

③

s_1	0	A	E	I	
s_2	1	B	D	F	H
s_3	2	C	G		

① ~~Iterate~~ row = 0, < numRows → [numRows] times

② row = numRows - 2

② direction change → row >= 0

Leetcode 179

⑭ (Imp ques) Largest number

{1, 2, 3, 5, 43}

inefficient way: ↳ 12345 → find all PnC → n! complexity.

12354

12453

54321 → Largest

single digit case

{1, 2, 3, 5, 4}

↳ "1" "2" "3" "5" "4"

→ sort in descending order (using comparator) or sort in ascending order and then read from backwards

→ lexicographically sorting is done in string ↳ like dictionary me hota hai

→ eg: 3|30|34|5|9

↳ 34, 30, 3

→ 9|5|34|30|3 → 9534303 ✗

→ 9534330 ✓

swap

eg: 10|2

→ 2 10 ✓

eg: 3 30

303 ✗

330 → we want

→ mycomp (a, b) {

a = 30, b = 3

t_1 = 303 → a+b

t_2 = 330 → b+a

return $t_1 > t_2$

true → $t_1 > t_2$ → a+b

false → $t_2 > t_1$ → b+a

agar $t_1 > t_2$ toh a ko pehle rakhdena

warna b ko pehle rakhdena

}

Leetcode 539 (Imp)

⑮ Minimum Time Difference

"HH:MM" (int, int)

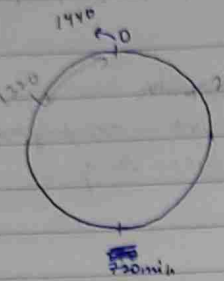
0 1 2 3 4

"01:00"

initial pt → 0 mins

00:00 (0 min) 01:00 (60 min) 02:00 (120 min) 03:00 (180 min) ---- 12:00 (720 min) 13:00 --- 23:59 (1439 min) 00:00 / 24:00 (1440)

min = 60 * HH + MM

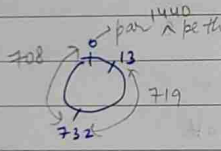


eg: → 00:00, 03:00, 09:00, 04:00
0 | 180 | 540 | 241 mins
↳ sort
0 | 180 | 241 | 540
diff: 180 61 299
↑ min

eg: [00:00, 23:59] → [0, 1439]

eg: [00:00, 04:00, 22:00] → [0, 240, 1320]
240 1080 1320
→ last entry ka 1440 se difference bhi do

eg: 12:12, 00:13
732 13
⇒ 13 | 732
719



toh mins[0] bhi add krna hoga

Leetcode 2125.

(16) Number of Laser Beams in a Bank

r1 0 1 1 0 0 1 → 3 security devices [no. of laser beams → ?]
r2 0 0 0 0 0 0 → 0
r3 0 1 0 1 0 0 → 2
r4 0 0 1 0 0 0 → 1

3 → 3×0 + 3×2 = 6
0 → 0×2 = 0
2 → 2×1 = 2
1 Total = 8

eg: 0 1 1 0 0 1 → 3 → 3×0 + 3×2 = 6
0 0 0 0 0 0 → 0 → 0×2 = 0
0 1 0 1 0 0 → 2 → 2×1 = 2
0 0 1 0 0 0 → 1 → 1×5 = 5
1 1 1 0 1 1 → 5.

3 | 0 | 2 | 1 | 5
0 1 2 3 4
i j

TC: O(n²)

Leetcode 1209.

(17) Remove all Adjacent Duplicates in String II

eg: deeedbbcccbdaa, K=3
dd bb ccc bdaa
ddbbbdaa
dddaa
aa

(M1) d e e e d b b c c c b d a a

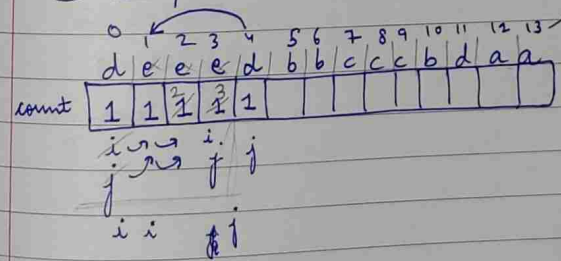
ans → d e e e d b b c c c b d a a
K-1 (pop) K-1 same

while adding new ch in ans, check if last (K-1) char are same to new ch.

① (ansize < K-1) → push ch
② if newch == last K-1 char in ans string
K-1 char → pop
new ch won't be pushed
else push new ch

TC: O(n × K)

(M2) Two pointer



→ inplace modification

→ if (i > 0) && s[i-1] = s[i]
→ count[i] += count[i-1];
→ if (count[i] == K) {i -= K}
→ s[i] = s[j]

Algo ⇒ i=0, j=0

① s[i] = s[j]

② count[i]=1;

③ if i>0 && s[i-1] = s[i]
count[i] += count[i-1]

④ if (count[i]==k) { i -= k }

⇒ i++, j++

Tc: O(n)

SC: O(n)

s.substr(0, i)

Pointers - Megaclass (Practice) (Live class)

6 July 2024

eg: void swap(int *x, int *y) {
int t;
t = *x;
*x = *y;
*y = t;
}
main() {
int a=10;
int b=20;
swap(a,b); ⇒ swap(&a, &b);
}

o/p ⇒ error

o/p ⇒ 20, 10

eg: junk(int *i, int *j) {
*i = *i * *i; ⇒ *i = *i * *i;
*j = *j * *j;
}
i=5, j=2;
junk(&i, &j); → o/p ⇒ 25 4

eg: int fn(int *m) {
return m+2;
}

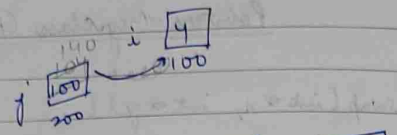
address +2
int* type

main() {
int i=35, *z;
z = fn(&i);
cout<<z;
}

⇒ error

eg: int i=4, *j, *k;

j = &i; → 100
j = j+1; → 104
j = j+9; → 140
k = j+3; → 152



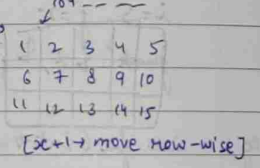
eg: int num[] = {24, 34, 12, 44, 56, 17};
100 104 108 ----

int i, *j;
j = &num[0];
for(i=0; i<=5; i++) {
cout << j; → 100, 24, 104, 34, 108, 12,
cout << *j; 112, 44, ---
j++;
}

eg: int x[3][5] = {{1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15}};

~~int *n = &x;~~ ~~wrong~~ (base array ka address) ⇒ int (*n)[5] = x;
or int *n = &x[0][0];

[(x+2)+1) → 12
*(*x+2)+5 → 3+5 ⇒ 8
((x+1)) → 6
((x)+2)+1 → 4
((x+1)+3) → 9
*n → 1
*(n+2) → 3
(*(n+3)+1) → 5
*(n+5)+1 → 7 ; *n++ → 1
++*n → 3 ← after increment

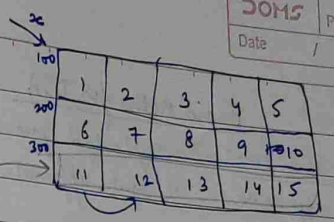


[x+1 → move row-wise]

eg: int a[5] = {1,2,3,4,5};
&a[0] → address of first element (1 element address)
&a → address of an array of size 5. (address of whole array)

① &a[0][0] → address of single box
② a → whole array → 5 element (ajit ne bola hai) → with first row
&a

x ⇒ base address (100)
*x ⇒ 100
** x ⇒ 1
x+2 ⇒ 3rd row
*(x+2) ⇒ *(300)
((x+2)+1) ⇒ 12



same chez ko point kar raha hai

Basic Maths for DSA (recording)

① Prime numbers → exactly/only 2 factors. eg: 2, 3, 5, 7

- naive approach
- sqrt approach
- sieve of Eratosthenes
- segmented sieve.

Naive approach

i/p → N

o/p → Prime or not

N = 10

2	- - -	9	10
↳ i			

(N % i) == 0 ⇒ not prime else Prime

★ Leetcode 204. Count Primes

given → N

[0 ⟶ N-1]

↳ 22 primenos. count

[0, 1 → not a prime]
[smallest → 2]

① Naive → (gives TLE)

```
for(i=2; i<n; i++){ if (isPrime(i))c++;
}
return;
```

isPrime

TC: O(n-2) ⇒ O(n)

outerloop ⇒ O(n)

Total: TC ⇒ O(n²)

② sqrt approach → (gives TLE)

Make better isPrime no.

originally ⇒ i = 2 ⟶ i < n

⇒ let N is non-prime

n = a × b

1	2 - - -	√n	- - - N-1	N

→ atleast 1 factor

if a > √n, b > √n ⇒ a·b > n ✗ not possible

⇒ atleast one of the factor must be less than √n

TC of isPrime ⇒ O(√n)

Total TC: O(n√n)

③ Sieve of Eratosthenes

N = 21 → mark all as prime

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
sieve →	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

prime 2 ⇒ table of 2 ⇒ 2 4 6 8 10 12 14 16 18 20

prime 3 ⇒ table of 3 ⇒ 3 6 9 12 15 18

prime 5 ⇒ 5 10 15 20

(8 primes <21)

mark multiple of primes as false.

① 2 → n-1 ⇒ array was, mark all of them as prime

② start from 2 till end, mark all the no. comes in table of 2 as non-prime.

③ Repeat ② till N-1 (only for primes)

④ Remaining elements will be counted.

④ Segmented sieve : find prime between l (low) and h (high)

↳ google

TC of sieve of Eratosthenes:

T.C → n [n/2 + n/3 + n/5 + n/7 + n/11 + ..]

→ n (1/2 + 1/3 + 1/5 + 1/7 + ---) → Taylor series

→ HP of prime nos.

→ n (log(log n)) ⇒ O(n * log(log N))

GCD / HCF

(greatest common divisor → highest common factor)

1. Euclid's Algorithm for GCD ⇒ gcd(a,b) = gcd(a-b, b) (a > b) OR gcd(a, b-a) (b > a) → apply this till one of the parameter becomes 0
2. LCM (a,b) × gcd (a,b) = a × b.

eg: 24, 72 → gcd?

24 = 1×2×2×2×3; 72 = 1×2×2×2×3×3

hcf = 1×2×2×2×3 = 24

eg: gcd(72,24) = gcd(48,24) = gcd(24,24) = gcd(0,24) ans

- LCM

LCM × HCF = a × b

$$lcm = \frac{a \times b}{gcd}$$

- Modulo Arithmetic

1. $(a \% n) \Rightarrow [0, \dots, n-1]$ eg: $10 \% 3 \Rightarrow [0,1,2]$
2. Properties
   - $(a+b) \% M = a\%M + b\%M$
   - $a\%M - b\%M = (a-b)\%M$
   - $((a\%M)\%) \% M = a\%M$
   - $a\%M \times b\%M = (a \times b)\%M$

⇒ To avoid overflow while storing integer we do modulo with a large no.

- Fast Exponentiation

eg: a^b

1. Normal solution to find $a^b \rightarrow O(b)$ $\Rightarrow 2^{10} \Rightarrow \underbrace{2 \times 2 \times 2 \dots 2}_{10}$
2. Better solution $a^b \rightarrow O(\log b)$

   → loop (ans = 1) $\Rightarrow [0 \rightarrow b-1] \Rightarrow ans = ans \times a$

   → $a^b \Rightarrow$ if b is even $\Rightarrow a^b = (a^{b/2})^2$ eg: $2^{10} = (2^5)^2$

   if b is odd $\Rightarrow a^b = (a^{b/2})^2 . a$

   eg: $2^5 \rightarrow (2^4).2 \rightarrow (2^2 \cdot 2^2).2 \rightarrow ((2^1.2^1).(2^1.2^1)).2$

Advanced Topics (C.P)

1. Pigeon Hole
2. Catalan number (BST)
3. Inclusion - Exclusion principle
4. Chinese Reminder Theorem
5. Lucas' Theorem
6. Fermat's Theorem
7. Probability concepts

OPTIMISING SIEVE → SEGMENTED sieve after OOPS (in end)
+ Pointers + Assignment

RECURSION

Recursion - Class-1 (Live)

(18 June 2024)

What is recursion?

- British way: (WHAT?) When a function calls itself directly or indirectly.
- Libabbar way: (WHERE?) (to solve apne) jab badi problem ka sol^n depend krta hai choti problem pe
- magical line: (How to apply?) bas ek baar hum solve krenge baaki recursion sambhal lega.

eg: factorial : $7! = 7 \times 6!$

big problem — small problem

$fact(n) = n * fact(n-1)$

eg: power : $2^7 = 2 \times 2^6$

pow(7) = 2 × pow(6) (n = 7)

Recursive / recurrence relation

$fact(n) = n * fact(n-1)$

big problem — mane liya — recursion (small problem)

Recursion components

- Base case / condition (tells when to stop)
- recursive call
- processing (optional)

eg: ~~tet~~

eg: let n=5

fact(5) = 5 × fact(4)

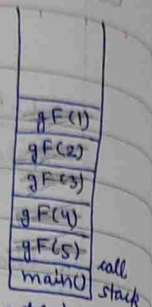
fact(4) = 4 × fact(3)

fact(3) = 3 × fact(2)

fact(2) = 2 × fact(1)

fact(1) = 1 × fact(0)

fact(0) = 1



Base case = ek aisa number jiska answer humko pata ho.

0! = 1

fact(0) = 1

*) recursion me base case nhi lagaoge toh stack overflow error dega.

HW → head recursion v/s tail recursion.

↳ recursive call last me ho toh tail recursion.

eg: print counting N to 1

```
void PC(n){
    cout<<n;
    PC(n-1);
}
```

eg: counting 1 → n

n=5

1 2 3 4 5

print(5) = 1 2 3 4 5

print(4)

cout<<5;

→ print(n)

↳ print(n-1)

↳ cout<<n

Fibonacci Series

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

nth term = (n-1)th term + (n-2)th term

n==0 → 0th term → 0

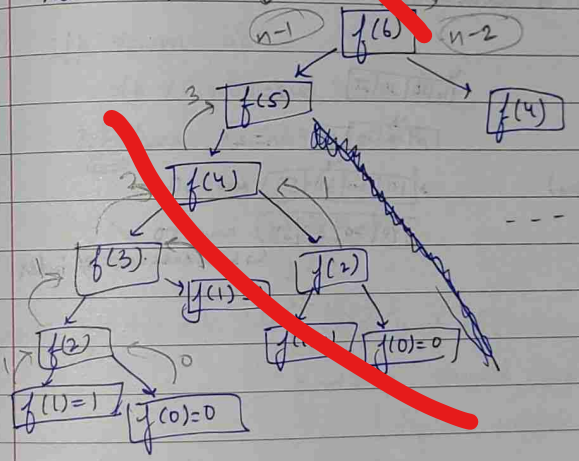
n==1 → 1st term → 1

} Base case

fib(n) = fib(n-1) + fib(n-2)

```
if(n==0 || n==1) return n;
int ans = fib(n-1) + fib(n-2)
return ans.
```

n → 6



eg: sum ⇒ 1 → n

getSum(5) → 1+2+3+4+5

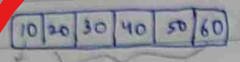
getSum(4)

getSum(5) = getSum(4) + 5

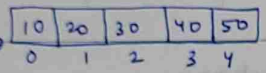
getSum(n) = getSum(n-1) + n

Recursion class-2 (live) (21 June 2024)

1. [illegible]
   - cout << arr[0]
   - (1 → n-1) remaining array → recursion will [illegible]



2. search in array target: 50

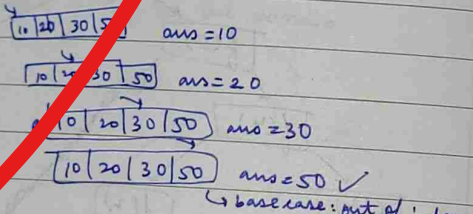


   - base case: index == size
   - arr[index] == target ⇒ return tru[illegible]

3. max no. in an array

   ans = INT_M[illegible]

   a[illegible] = max(arr[i], ans)

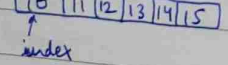


☆ → jab bhi kisi fun me [illegible] ka track rkna chahte ho toh by reference call krdo, copy[illegible]at banao
(e.g. int &x in this example)

4. min no. in an array

   ans = INT[illegible]MAX

5. Print all odd nos.

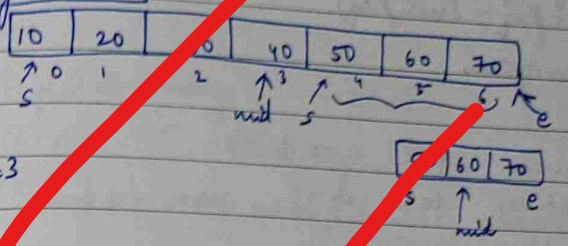
   i/p ⇒ 

   if (arr[index] % 2 == 1) ⇒ cout << arr[index]
   or &1

6. Print all even nos.

   (arr[index] % 2 == 0) ⇒ pri[illegible]

Binary Search

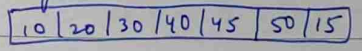


   base case ⇒ s>e ⇒ return -1.
   To go left or right ⇒ recursive call.

7. i/p ⇒ num = 389
   o/p ⇒ digits ⇒ 3, 8, 9

   ~~3 8 9~~ 89 % 10 → 9
   ↓ /10
   38 % 10 → 8
   ↓ /10
   3 % 10 → 3
   ↓ /10
   0 → stop

HW → T[illegible] ⇒ array is sorted or not



Char Arrays & Strings Mega Class (live class) (23 June 2024)

Leetcode 1781. Sum of Beauty of All Substrings

Beauty of s ⇒ (Most freq - least freq)

Find ⇒ All substring's beauty sum

s = "aabcb"

	beauty				
a	→0	a →0	b →0	c →0	b →0
aa	→0	ab →0	bc →0	cb →0	
aab	→1	abc →0	bcb →1		
aabc	→1	abcb →1			Total = 5
aabcb	→1				

(M-1) Find all substrings (nested loop)

aabcb

① a ⇒ 1-1 = 0

② aa ⇒ 2-2 = 0

③ aab ⇒ 2-1 = 1

④ aabc ⇒ 2-1 = 1

⑤ aabcb ⇒ 2-1 = 1

map

a	✓	2
b	✓	2
c	1	

abcb

a ⇒

ab ⇒

map

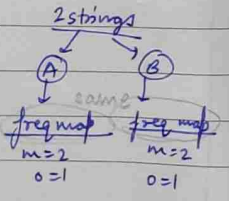
a	1
b	1

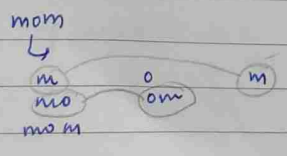
TC: O(n³)

HackerRank: Sherlock and Anagrams.

mom — mmo

(or) sort and check if they are equal ⇒ Anagrams ✓





Total = 2 pairs

abba

a — b — b — a

ab — bb — ba

abb — bba

abba

Total = 4 pairs

Jugaad tareeka

① find all substrings → O(n²)

② store " : 10 substring in this eg. → extra space

③ Loop over stored substrings to check anagram or not. → O(no. of substring)

HW [code]

acha tareeka ⇒ abba

a

ab ⇒ sort ⇒ ab

abb ⇒ sort ⇒ abb

abba ⇒ " ⇒ aabb

b ; bb ; bba ⇒ abb

b ; ba ⇒ ab

a

map (substring counts)

string	int	
a	✓	2
ab	✓	2
abb	✓	2 (anagram)
aabb	1	
b	✓	2
bb	1	

∴ calc. no. of anagramatic pairs.

⇒ count > 1

∴ $^nC_2 = \frac{n!}{(n-2)!\,2!} = \frac{n(n-1)}{2}$

① a=2 = $\frac{2(2-1)}{2}$ = ① pair

② ab = 2 ⇒ $\frac{2(2-1)}{2}$ = ①

③ abb = 2 ⇒ ①

④ b = 2 ⇒ ① ⇒ 4 pairs

① Generate all possible substrings

② sort each substring

③ count occurrence of each sorted string ↳ map

④ calc. pairs.

Leetcode 151. Reverse Words in a String

"---hello--world---"

SC: O(1)

→ reverse individual char

i=0, start=0, end=0;

--dlrow--olleh---

→ after reversing whole string only 2 thing left
① Remove extra space
② individual word reverse

① skip leading space

```
while (s[i]==' ') i++;
```

② copy char of a ~~word~~ word to the correct position.

```
while (s[i]!=' ') {
    s[end++]=s[i++];
}
```

"--dlrow--olleh---"

--dlrow--olleh---

⇒ dlrow--olleh---

③ Reverse individual word.

```
reverse(s.begin()+start, s.begin()+end);
```

④ add space btw words.

```
s[end]=" "; end++;
```

world_w_olleh---

```
start=end;
```

⇒ we mark one iteration complete ⇒ i++.

world_w__olleh---

world_w__olleh---

⇒ world_olleh_leh---

reverse

⇒ world_hello_leh---

⇒ world_hello_eh---

* s.resize(N)
↓
no. of char

final step ⇒ s.resize ~~(end)~~ (end-1)

TC: O(n²)

(HARD)
Leetcode 68. Text justification

① spaces added between words
↳ atleast one space
↳ aur spaces thi ho skte hai but maxwidth se jyada nhi overshoot hona chiye.

eg: This, is, an, example, of, text, justification.
maxwidth=16.

This_is_an_example → move to next line
4 1 2 1 1 7 ⇒ 18 == 16? X

② ∘ leading and trailing spaces nhi ho skti.
③ ↳ sirf last line me trailing space ho skti hai.
ya fir single word hi aa rha hai sirf line me.
④ last line should be left-justified (or even in case of single-word)

full-justified
↳ when spaces can be evenly divided
↳ can't be evenly divided
↳ greedily add space from left to right.

Approach

case1: current line is complete.
This_is_an_example

{ This, is, an, example } → This_ is_ an_ example_
↳ jitne words hai utne hi space diye

(if) cond"
I don't need "example"
⇒ [This, is, an] → ise mila kar mujhe poori line banani hai
extraspace ⇒ maxWidth − currentline ke Total chars
~~=8 − 8~~ = 16 − 8 ⇒ 8

⇒ spaces in between ⇒ $\frac{8}{3-1}$ = $\frac{\text{extraSpace}}{\max(\text{currline.size()} - 1)}$
↑ to avoid any error, take max with 1

⇒ remainder = extraSpace % (word−1)
⇒ 9 % 2 ⇒ 1

TC: O(n^2)

Date / /

Recursion class-3 (live) (24 June 2024)

eg: (~~last classHW~~) i/p ⇒ | 10 | 15 | 25 | 35 | 60 | 95 |
0 1 2 3 4 5

arr[i+1] > arr[i] ✓ sorted (↑ order)

V.V. Imp.
Include / Exclude

X_A, X_B, X_C → C_1, C_2 → inc, exc, inc, exc

Subsequence → ch → inc / exc
◦) String → Racecar (substring) ⇓ not necessarily continuous
condition ↳ order maintain ✓

L O V E
↳ L o e → subsequence

eg: abc → a, b, c, ab, bc, ac, abc, " " ↳ subseq.

string length → n (in above eg. (3))
subsequence ⇒ 2^n (2^3) = 8

(M2) → done in doubt class
abc
000 → " "
001 → c
010 → b
011 → bc
:
111 → abc

HINT → N = str.size()
Total count ⇒ 2^N ⇒ 1<<N, POW(2,N);
for(i=0 < totalCount){
string ans = " ";
for(j=0; j<N; j++){
check if j-th bit is set in i;
}

eg: abc → i/p ; o/p ⇒ " "
inc / exc
" " + a = "a"   " "
in / ex   inc / ex
ab   a   b   " "
in / ex   in / ex   in / ex   in / ex
abc   ab   ac   a   bc   b   c   " " → subsequences
Base class ⇒ loop ke bahar giao ab

Leetcode 198 House Robber → TLE

eg: | 2 | 7 | 9 | 3 | 1 |
0 1 2 3 4
i ✓ chori; i+1 ✗; i+2 ✓

or i → chori ✗
i+1 → chori ✓

Base case ⇒ i > = n

→ TLE

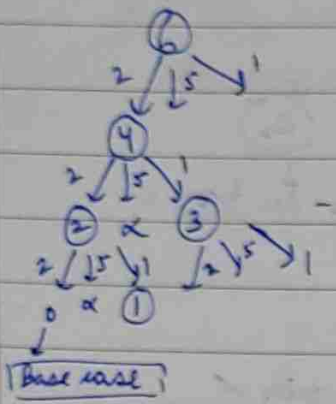
leetcode322: coin change

eg: coins : 2, 5, 1

amount : 6 Rs.

2+2+2, 2+2+1+1, 1+1+1+1+1+1, 5+1, etc...

ans → min count of coins ✓



- har amount par poore coins wale array par travel kra hai
- if coins > amount → call mt kro ~~return -1~~.

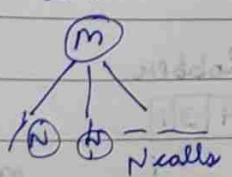
Base case

```
int solve(coins[], amount) {
    //base case
    if (amount == 0) → return 0;
    int mini = INT_MAX;
    for(i=0 → i < coins.size()){
      int coin = coins[i];
      if (coin <= amount){
          int recAns = solve(coin, amount)
          if (recAns != INT_MAX){
          int coinUsed = 1 + recAns;
          mini = min(coinUsed, mini)
      }
    }
    return mini;
}
```

TC: 3^n → n^m → N^m

n: coin size

m: amount



Week-7 Assignments

① Last occurrence of a char

eg : string = "abcddefg"
char X = "d"

(M1) search from left to right (Recursively)
(M2) " " right to left (")
(M3) STL fn : strrchr()

(M1) L TO R

1) Base case
2) ek case solve
3) Rec. sambhal lega

a b c d d e d g
0 1 2 3 4 5 6 7
i

if (s[i]==x) { ans = i; }

(M2) R to L

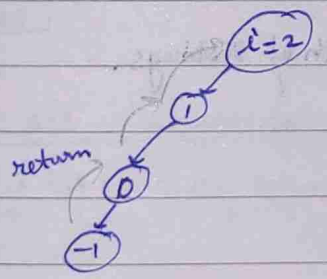
a b c d d e d g
0 1 2 3 4 5 6 7
i

i = N-1

T.C

a b c
0 1 2

x = 'd'



O(n+1) = O(n)
SC : O(n)

② Reverse a string RE

recursively

eg : s = "abcd" inplace reverse ✓

start = 0
end = n-1
↓
size of string

a b c d
↑ start ↑ end

↳ 1 case : swap(s[start], s[end])

Base case : start >= end

T.C : reverse(start, end) → $O(\frac{n}{2}) \Rightarrow O(n)$

reverse(start+1, end-1)

⋮

reverse(start==end) → return

S.C : ~~O(n/2) ⇒ O(n)~~ $O(\frac{n}{2}+1) = O(n)$

leetcode 415.

③ Add string RE

Num-1 ⇒ "1234"

↳ each is an int

eg : num-1 → "456" p1 → 11 / 456
num-2 ⇒ "77" p2 → + 077
533
carry ① ③

num1[p1] + num2[p2]

base case ⇒ p1, p2 dono left end me pahuch gye.

TC : O(N+1) = O(N)

SC : ~~O(N+1)~~ O(N+1) = O(N)

max length of 2 strings.

④ Palindrome check RE

eg : racecar
start ↑ ↑ end

if (s[start] != s[end]) return false;

TC : $O(\frac{n}{2}) = O(n)$

SC : $O(\frac{n}{2}+1) = O(n)$

⑤ Print all Subarrays using RE

eg : 1 | 2 | 3 | 4 | 5
0 1 2 3 4

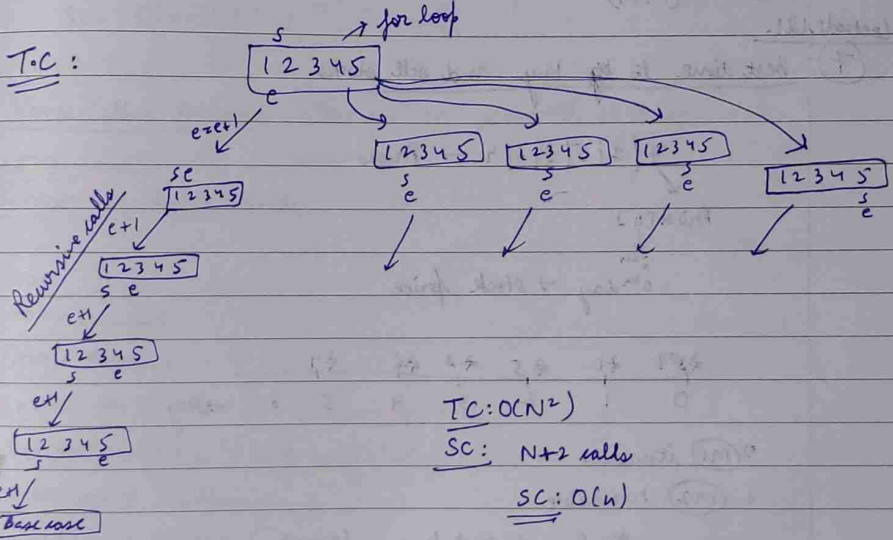
① Base case
② 1 sol° mai
③ baaki recursion sambhal lega

1 start=0, end=start
1 2 start=0, end=end+1=1
1 2 3 " e=2
1 2 3 4 " e=3
1 2 3 4 5 " end=end+1=4

Pointers → start, end

next set of sub-array
2 start=1, end=start=1
2 3 start=1, end=end+1=2
2 3 4 " , end=3
2 3 4 5 " , end=4

T.C :



TC : O(N²)

SC : N+2 calls

SC : O(n)

leetcode 1910.

⑥ Remove all Occurrences of a Substring

s ⇒ daabcbaabcbc , part = "abc"
0 1 2 3 4 5 6 7 8 9 10 11
left string right string

(already done in char array & string class-2 in better way doing now with recursion.)

Ⓐ ① case ⇒ find part string pos° in "s".
Ⓑ Remove part from s.
Ⓒ Call again fn for new string : ⇒ s = leftString + rightString

String :: substr(0, no. of char)
starting pos index ↳ length

N → s.size, M = part size

TC: ① find → O(NM)
② leftpart = O(n) } O(n)
③ rightpart = O(n)

⇒ O(NM) → TC of 1 case

recursive calls in worst case? $\frac{N}{M}$ $O(\frac{N}{M})$

s —s.size→ N
part —part size→ M

TC: $O(\frac{N}{M} \times NM) \Rightarrow O(n^2)$

SC: $O(n/m)$

leetcode 121.

⑦ Best time to ~~buy~~ buy and sell ~~stocks~~

7|1|5|3|6|4 → Prices

Prices[0] → 0^{th} day → stock price

₹7	₹1	₹5	₹3	₹6	₹4	
0	1	2	3	4	5	6 → Day

- (M1) iterative → ek hi way hai dig
- (M2) recursive

~~Buy~~ Buy → stock price is lowest
sell → stock price is highest in future

- variable → minPrice → Buy

① Base case
② 1 case → ① minPrice
③ Rec. call ② maxProfit → sell
(prices[i] − minPrice)

7|1|5|3|6|4
→ 0 1 2 3 4 5

minPrice → INT_MAX
maxProfit → INT_MIN
1 case →
```
if (prices[i] < minPrice) {
    minPrice = prices[i];
}
if ((prices[i] - minPrice) > maxProfit) {
    maxProfit = prices[i] - minPrices;
}
```
TC: O(n+1) ⇒ O(n)
SC: O(n+1) ⇒ O(n)

leetcode 198

⑧ House Robber Problem → done in rec class - 3 already

~~leetcode 273~~ (Hard)

⑨ Integer to English Word

1 –
10 –
100 – hundred
1000 – thousand
10,000 – ten thousand
1,00,000 → 100K ✓ → 1L ✗
1,000,000 → 1 million
10^9 → 1 billion

eg: 1 2 3 4 5 6
→ 123 thousands
"one hundred twenty three thousands"

eg → 1 2 3 4 5 6 7
→ 1 million 234 thousand five hundred & sixty seven.

eg: 123 → 100
20
3

N=123 → 123>100

num → 123/100 → 1.23 ⇒ 1

100

→ one hundred

N = 123%100 ⇒ 23 recursively same chez ab 23 ke sath

konsi baar abd so on....

⇒ Twenty ✓

N = 23%20 ⇒ 3

N = 3 ⇒ three

① Find N just >= in the map (N >= mp.num)

⇒ if (N >= 100)

↳ How many mp.nums?

⇒ (N/mp.num) → RE call ✓

⇒ update N ⇒ N = N % mp.num

⇒ it ⇒ {100, "Hundred"}

first second

eg: 12345

(12) Thousand (345)

a → num/1000 ⇒ 12

b → it.second

c → num % it.first

TC : 1 digit ⇒ less than two words <2 words

eg: N = 12345

T.C ∝ no. of words

no. of words ∝ no. of digits

no. of digits

123 → ceil (log10 N) ⇒ ciel (log10 123) ⇒ ciel (2.09) ⇒ 3

digit → ciell (log10 N)

TC ∝ no. of digits

TC ∝ ciel (log10 N) ∝ (log10 N)

O(n) = (log10 N)

S.C : ∝ No. of digits

O(n) ∝ log10 N

O(n) = log10 N

↗ TLE

leetcode 44. (Hard)

⑩ Wild card matching

⇒ '?' → denote single character

'*' → denote any no. of char [0, ∞)

eg: ⇒ s ⇒ "aa", p = "a" → return false

eg: s ⇒ "aa", p = "*" → return true

p = '*'

eg: s = "aa"

p = "?a" → return true

eg: s = **ab (empty) → ✓

p = ab

Rule ⇒ ① out of all comparisons, if any comparison returns true ⇒ my answer will be true.

eg: s = abcdefg (si)

p = ab*fg

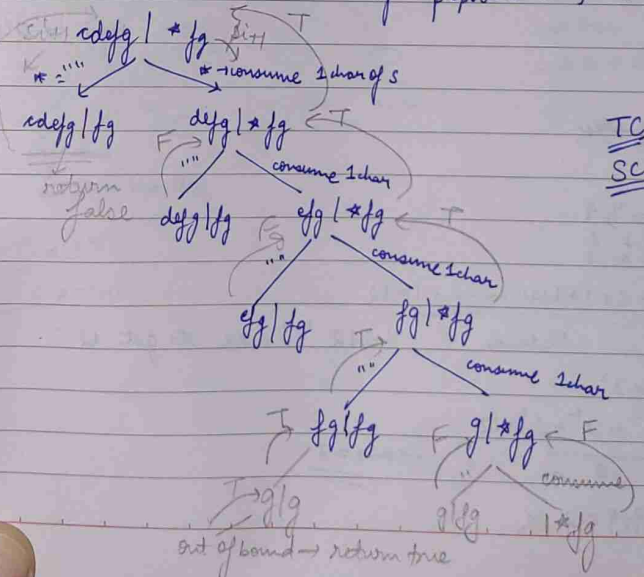
pi

if (s[si] == p[pi] || p[pi] == '?') {

// match

call to check pi+1, si+1 }

if (p[pi] == '*') ⇒ ??



TC : 2^n → fibonacci

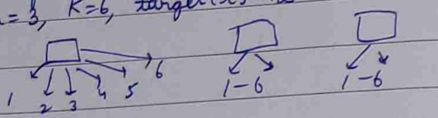
SC : O(n+1)

O(n)

leetcode 1155. → TLE

(11) Number of Dice roll with Target Sum (dp ka ques hai)

eg: n=3, K=6, target(t)=12



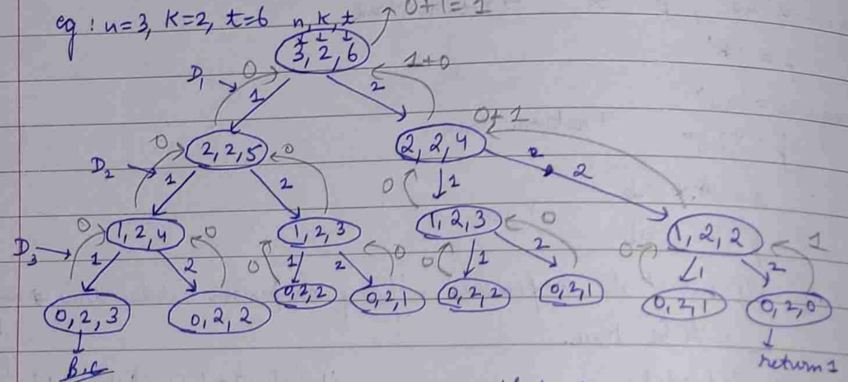
| 6 | 6 | 6 |

6 × 6 × 6 = 216 ways

$D_1 = 1$, $D_2 = 5$, $D_3 = 6$ = 12

1 6 5 = 12

eg: n=3, K=2, t=6



B.C

n=0
t!=0 ⇒ return 0

(n=0, t=0 → return 1)

⇒ n!=0 & t=0 ⇒ return 0

output ⇒ 1

TC: K^n → no. of dice, → no. of faces

SC: n+1 → O(n)

leetcode 279. → TLE

(12) Perfect Squares

N=12

↳ 1, 4, 9.
↓ ↓ ↓
1 2 3

⇒ 1^2 + 1+1+1+1+1+1 --- +1 = 12 (12 times) ↳ 12 '1' used to get 12.

⇒ $4 = 2^2$

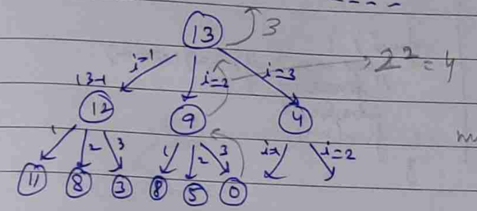
$2^2 + 2^2 + 2^2 = 12$ (3) → ans = 3

⇒ 9+9 > 12 ✗

i=1 ⇒ Perfect Square (PS) = 1

1 + (11)

↳ 1 + (10) - - -



$2^2 = 4$

TC: $(\text{sqrt } n)^n$ ⇒ $(\sqrt{n})^n$ ⇒ $O(n^{n/2})$

SC: O(n)

numSqHelper (13) = 3

ans ⇒ 3-1 = 2

leetcode 983. → TLE

(13) Minimum Cost For Tickets

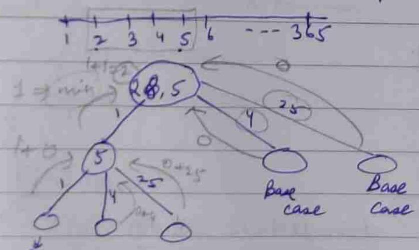
eg: ⇒ 2day ⇒ 1 3

1day → 2Rs

7days → 50

30days → 250

eg: [3,5] cost → 1, 4, 25 ⇒ (1day, 7day, 30 days)



cost 1 ⇒ 2
cost 7 ⇒ 4
cost 30 ⇒ 25
} min = 2 ✓

TC ⇒ 3^n

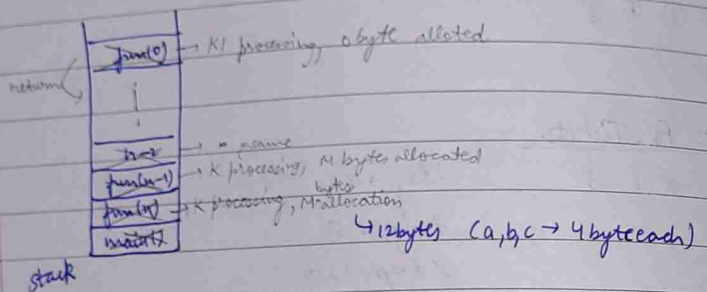
SC ⇒ O(n)

Time and Space complexity of recursive solutions [recorded]

T(n) → Time taken as a f" of input 'n'.

```
main() {
    fun(n);
    return 0;
}
```

```
fun() {
    // base case → if(n==0) return;
    // processing
    int a,b,c;
    fun(n-1);
}
```



↳ 12 bytes (a,b,c → 4 byte each)

eg: print Array (linear traversal of an array)

```
void printArray(int a[], int n) {
    if(n==0) return;                    → K1
    cout<<*a<<" ";                      → K
    printArray(a+1, n-1);
}
```

⇒ F(n) = K + F(n-1)

TC: T(n) = K + T(n-1) (M1) Formula based method

T(n-1) = K + T(n-2)

T(n-2) = K + T(n-3)

⋮

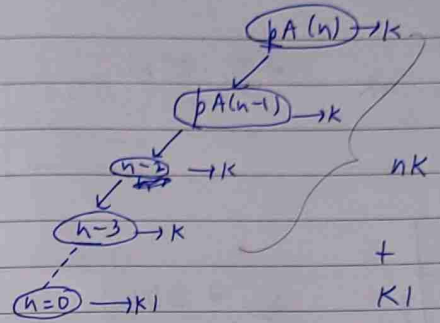
T(1) = K + T(0)

add I(0) = K1

T(n) = nK + K1

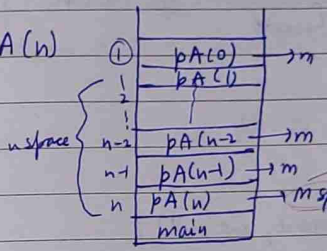
O(nK+K1) ⇒ O(n)

(M2) Recursive tree method ↳ visual / better method



TC: O(nK+K1) = O(n)

SC: pA(n)



SC: O(n+1) = O(n)

eg: Factorial RE

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

F(n) = n × F(n-1)

TC: (M1) T(n) = K + T(n-1) → equation

T(n-1) = K + T(n-2)

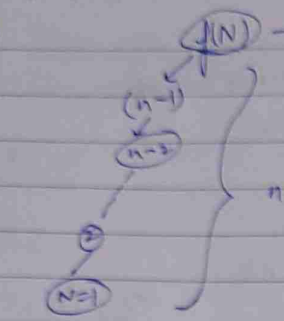
⋮

T(1) = K

T(n) = nK

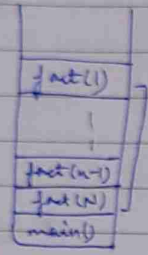
O(Tn) = O(nK) = O(n)

M2



⇒ TC : O(n)

SC :



NXm
↳ constt

SC ⇒ O(Nm)
SC : O(n)

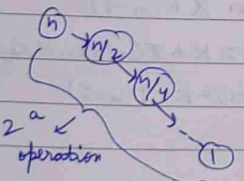
eg: Binary Search RE

```
int BS(int a[], int k, int start, int end) {
    if (start > end) return -1;
    int mid = start + (end-start)/2;
    if (a[mid] == k) return mid;
    else if (k > a[mid]) return BS(a, k, mid+1, end);
    else return BS(a, k, start, mid-1);
}
```

⇒ F(n) = K + F(n/2)

TC : T(n) = K + T(n/2)
T(n/2) = K + T(n/4)
T(n/4) = K + T(n/8)
⋮
T(2) = K + T(1)
add T(1) = K
T(n) = aXK

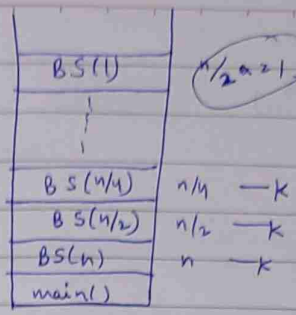
TC : O(logn)



$\frac{n}{2^a} = 1$

a = logn

SC :

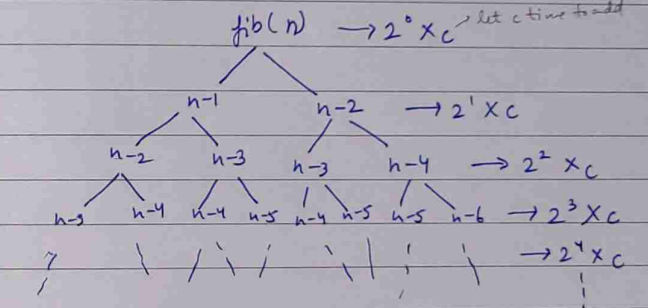


→ a = logn

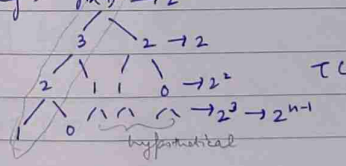
SC ⇒ O(n) = O(K logn)
SC : logn = O(n)

eg: Fibonacci Series RE

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

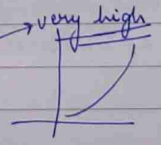


eg: fib(4) → 2^0

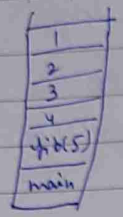


T(n) ≤ 2^0 + 2^1 + 2^2 + 2^3

T(n) ≤ c[2^0 + 2^1 + 2^2 + ---- 2^(n-1)]
T(n) ≤ c[2^n - 1]
T(n) ≤ 2^n - 1
TC : O(n) → 2^n
TC : O(2^n)
↓
exponential TC



SC : 4:N=5



SC : O(n)

BACKTRACKING and DNC

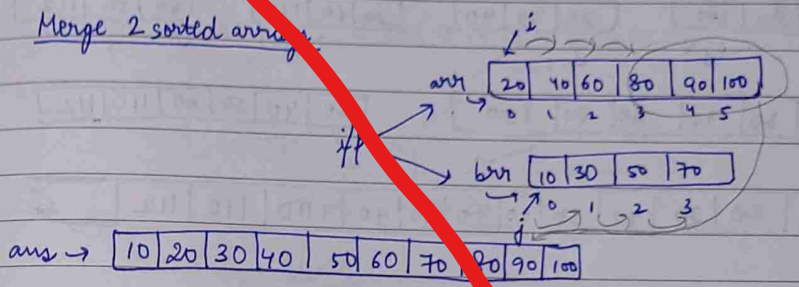
DnC Class-1 (Live)

26 June 2024

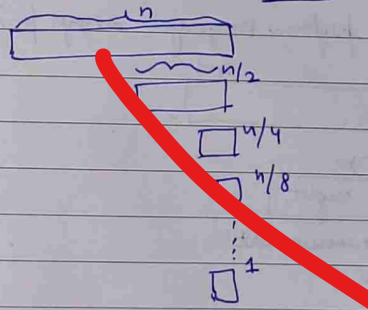
(Divide and Conquer)

→ Merge Sort

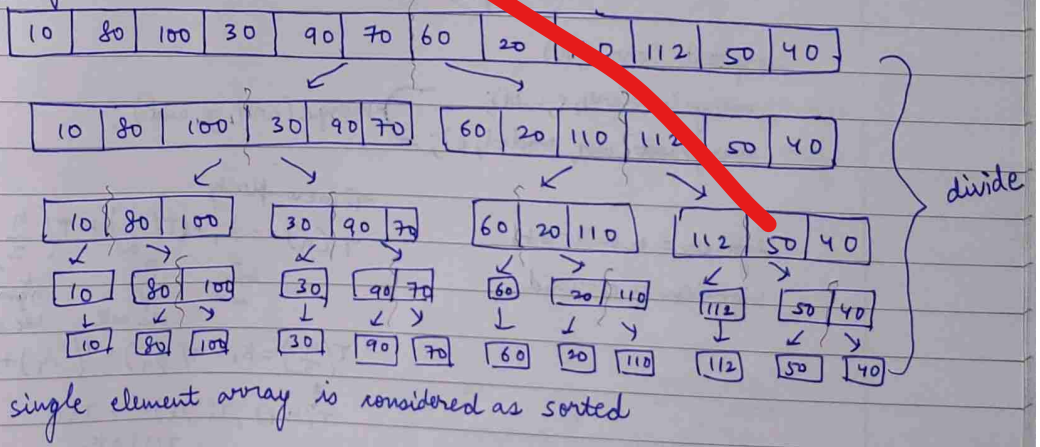
Q. Merge 2 sorted arrays



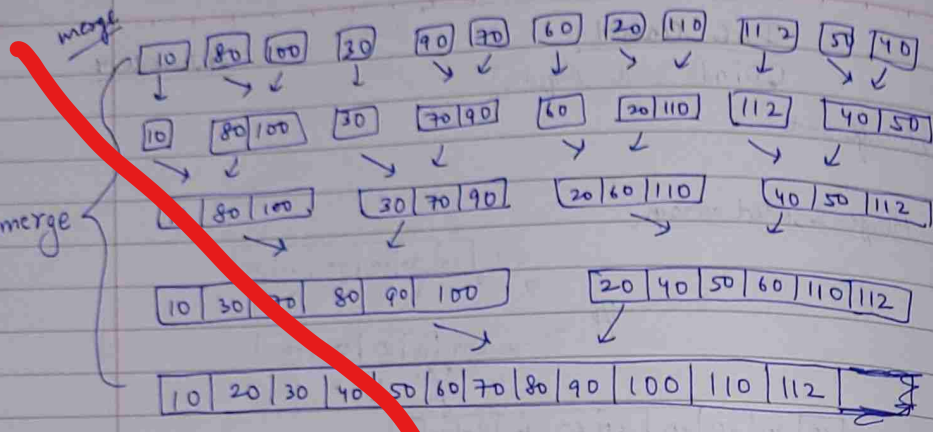
→ Merge Sort is based on DnC (Divide and conquer)



→ i/p array



single element array is considered as sorted

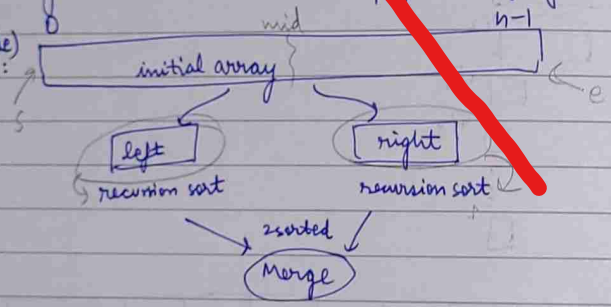


TC: O(nlogn)

↳ same in all cases.

(HW) Q. merge sort linkedlist pe acha perform kregi ya arrays pe? and why?

logic (for code) :




```
mergesort (arr, s, e)
left ← mergesort(arr, s, mid);
right ← mergesort(arr, mid+1, e);
merge(arr, se, mid);
```

leftlen = mid - s + 1

rightlen = e - mid

recursion

↓ divide ↑ conquer

Time complexity

$T(n) = k_1 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + (n*k)$

base case · left rec call · right rec call · merge

$T\left(\frac{n}{2}\right) = T(n/4) + T(n/4) + (nk/2)$

$T(n/4), T(n/8) \ldots T(1)$

$T(1) = k_1$

Q. HW → inplace merge sort (without create/copy array)

$T(n) = k_1 + 2T\left(\frac{n}{2}\right) + (n*k)$

$2T\left(\frac{n}{2}\right) = 2k_1 + 4T\left(\frac{n}{4}\right) + nk$ (multiply by 2)

$4T\left(\frac{n}{4}\right) = 4k_1 + 8T\left(\frac{n}{8}\right) + nk$ (multiply by 4)

$8T\left(\frac{n}{8}\right) = 8k_1 + 16T\left(\frac{n}{16}\right) + nk$

⋮

$T(1) = k_1$

(done in Binary search)

$a = \log n$

$T(n) = a \times (n*k)$

complexity $T(n) = n \log n$

Page No. Date / /

(Recording)

Quick Sort (End Element as Pivot)

start ↓ , end ↓

7	2	1	8	6	3	5	4
0	1	2	3	4	5	6	7 ← pivot

i j → Pivot

→ To place pivot such that the element to the right of pivot > a[pivot]

→ Element left of pivot element < a[pivot]

↳ new way to do partition

① pivot = end

⇒ pivot = 7

② i = start-1;

j = start;

a[pivot] = 4

~~while(j < pivot)~~
```
if (a[j] >= a[pivot])  j++;
else { i++;
     swap(a[i], a[j]); j++;}
```

```
or while(j < pivot){
   if (a[j] < a[pivot]) {i++;
        swap(a[i],a[j]); }
   j++;
}
```

TC: $O(N^2)$ → big O
worst case

avg case → $n\log n$ ⇒ Ω

best case → O(n) or $n\log n$ (already sorted) ↳ confirm it

2|7|1|8|6|3|5|4
i j

2|1|7|8|6|3|5|4
i j j j j j → out of bound

2|1|3|8|6|7|5|4
i

⇒ 2|1|3|4|6|7|5|8
i Pivot

Rec. quicksort (start → i-1)

Recursively call quicksort (i+1 → end)

Page No. Date / /

Recursion Marathon (Recording)

eg:

2	4	1	6	8	5	9
0	1	2	3	4	5	6

max sum of non adjacent element (houserobber) (done in recursion class-3)

inc → ans = 2 + f(i+2, n-1)

exc → ans = 0 + f(i+1, n-1)

→ max → finalAns

→ TLE

Leetcode 213. House Robber II

4 house

↳ money → {1, 2, 3, 1}
I II III IV

first ✓ ✗ last → ✗ and vice-versa

I → II → III → IV → I

I. | 1 | 2 | 3 | ~~1~~ | include 0 … n-2 → max

II. | ~~1~~ | 2 | 3 | 1 | 1 … n-1 → max

(gfg)

Q. Count Derangements (Permutation such that no element appears in its original position)

i/p ⇒ {10 20 30} → {30 10 20}
0 1 2 → 0 1 2

eg: n=1
{1} → 0

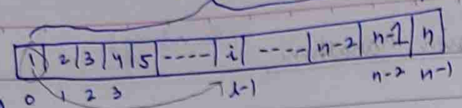
eg: n=2
{1,2} → {2,1} → 1

eg: n=3
{1,2,3} → {3,1,2}, {2,3,1} → 2

eg: n=4 → 9

0, 1, 2, 9 ---

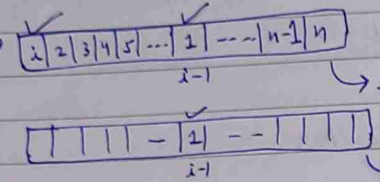
n element



$0 \to n-1$

$f(n) \to$

2 case



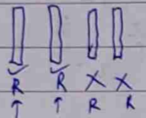
n-2 blocks left → f(n-2) → recursion
(+) → add kr do
n-1 blocks left → f(n-1)

(i not fixed)

(Bhot acha ques) → hard to think if doing for first time.
(gfg)

Q. Painting Fence Algorithm

n: posts
K: colors



dose jyada neighbour post me same color na ho.

CASES → same color
→ different color (last ke do post ke liye)

eg: K=3 (RGB)

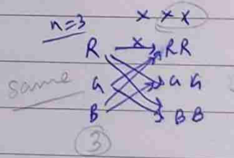
	n=2	n=3	n=4
same	RR, GG → [K], BB	RGG, BRR, RBB, BGG, GRR, GBB [K(K-1)]	RRGG, RRBB, ⋮ (18)
diff	RG, RB → [K(K-1)], BR, BG, GR, GB	RRG, RRB, GGR, GGB, BBR, BBG	RGR, BGR, RGB, BGB, RBR, GRG, RBG, GBG, BRB, GBRB, BRG, GBR

same × K(K-1)
+
diff × (K-1)

n = 4   x x (x x)

same + diff

$\Rightarrow \left[(K-1) \times f(n-2)\right] + \left[(K-1) \times f(n-1)\right]$



Imp ★★ → TLE
leetcode 72. Edit distance

convert word 1 into word 2 (using min no. of operations)

operations ↳ insert, remove, replace

word1 → h o r se
word 2 → h no s replace

5 operation ✓

word[i] }
word2[i] } → 2 case → matching word
→ not matching → perform operation.

match → age badh jao next char pe
base case → out of string

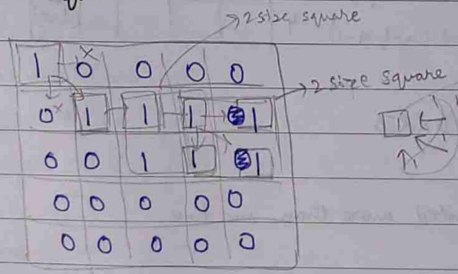
insert
w1 → n h o r s e (i)
w2 → n o s
j → j ⇒ j+1

remove
w1 → ~~h~~ o r se → i+1
w2 → n o s (j)

replace
w1 → h o r se ⇒ i+1, j+1
w2 → n o s (j)

→ TLE

leetcode 221 Maximal Square.



Imp → gfg: 0/1 Knapsack Problem (bhot imp) (dry)
leetcode: 1039, 1155

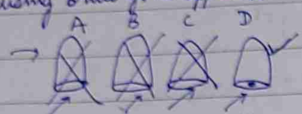
(Live)

DnC and backtracking class-2 — 8 July 2024

(20 min motivation)

Backtracking : variation of recursion.
- ↳ explore all possible ways
- ↳ way → already explored ↳ don't explore again

→ solve using brute force approach



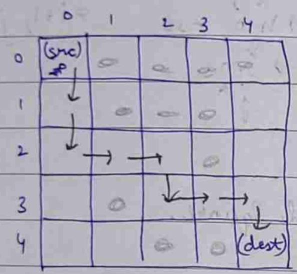
A me sikka mila, badme bhi nhi milega jab ek baar nhi mila toh, A ko dubara nhi uthayenge

→ discard

(gfg) (0 A me aa skta hai directly)

Q. **Rat in a maze**

- — → empty space
- ⊘ → closed space
- Rat → source location (src) → (0,0)
- destination (dest) → (n-1, n-1)



movements allowed

(U) ↑, ← (L), Rat → (R), ↓ (D)

DDRRDRRD

* no cell can be visited more than one time.

eg:

	0	1	2
0	1 (rat)	0	0
1	1	1	1
2	0	0	1 (dest)

0 → blocked

1 → empty space

src → (0,0)

dest → (2,2)

① out of bound — DRRD

(0,0) → D/R/U/L
↓
(1,0) → D/R/U/L
↓
(1,1) → D/R/U/L
↓
(1,2) → D/R/U/L
↓
(2,2)

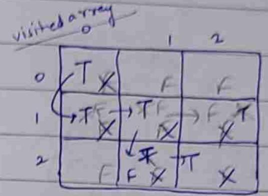
or → (0,0) → U/L/D/R
↓
(1,0) → U/L/D/R
↓
(0,0) → U/L/D/R
↓
(1,0) → U/L/D/R

0 ↓↑ 0 infinite loop

∘ **Track → visit status**

→ make boolean array value
↳ for each cell

eg:

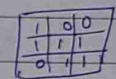


movement → empty
↳ vis → F

∘ vis[0][0] → True (rat betha hua hai src pe)

(0,0) → U/L/D/R
↓
(1,0) → U/L/D/R
↓
(1,1) → U/L/D/R
↓
(2,1) → U/L/D/R
↓
(2,2) → base case

(1,1) → (1,2) → U/L/D/R
↓
(2,2)



DRDR

DRRD

∘ change revert krke jana (T ko F krdena wapas jate time)

Brute force → extra cond^n ⇒ if (arr[0][0]==0)
↳ no answer (rat move nhi kr payega)

- up → (i, j) → (i-1, j)
- down → (i+1, j)
- left → (i, j-1)
- right → (i, j+1)

(or)

make array ⇒ dx → -1, 1, 0, 0

dy → 0, 0, -1, 1

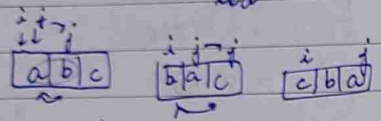
move → U, D, L, R

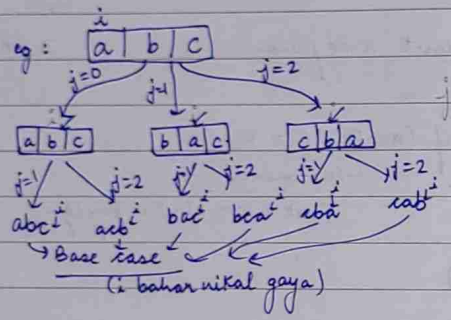
D&C and Backtracking class-3 (live class) 10 July 2024

leetcode 46.

eg: abc (3 length) → permutation → 3! = 6

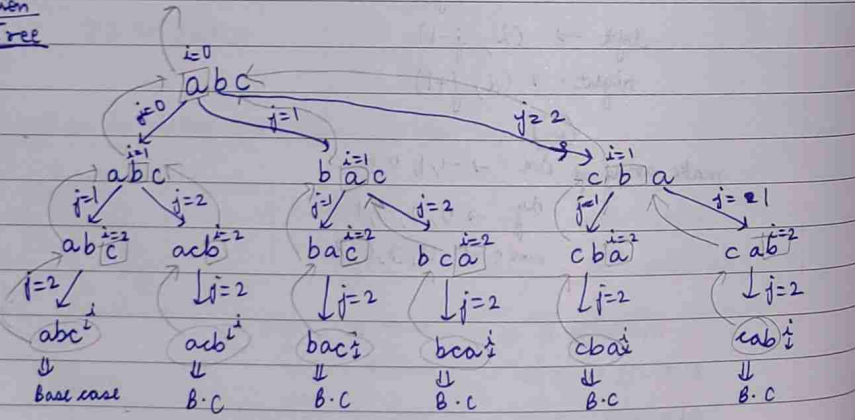
- abc
- acb
- bac
- bca
- cab
- cba





i ko i se lekar n tk chalana hai.

Recursion ^ Tree

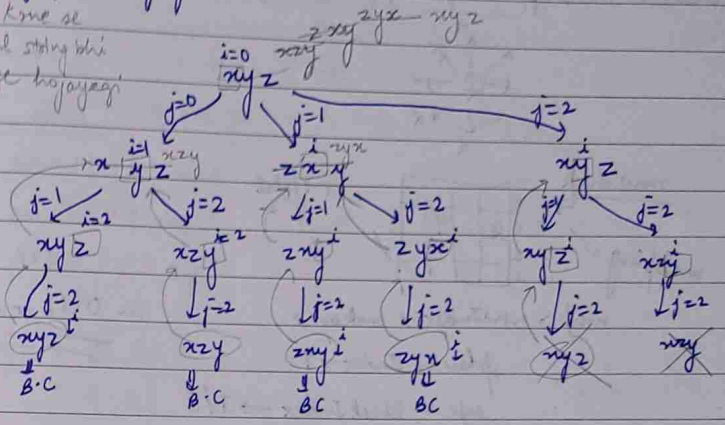


DBT class → STL method : sort(s.begin(), s.end());

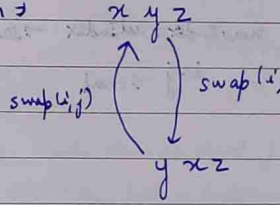
```
while(next_permutation(s.begin(), s.end())){
    cout<<s<<endl;
}
```

- if string passed by reference

swap krne se original string bhi change hojayegi



Solution →

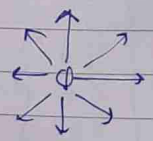


Wapas jaate time dubara swap karna

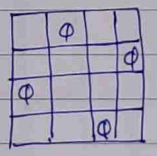
⇒ backtrack kro

leetcode 51. N-Queens.

i/p ⇒ n



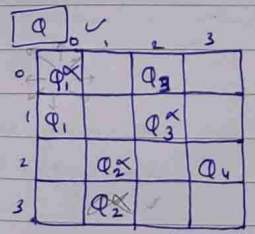
n x n

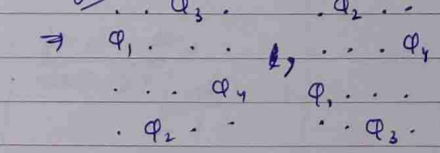


eg: n = 1

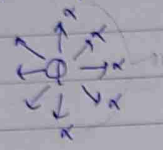
eg: n = 4

Q1
Q2
Q3
Q4



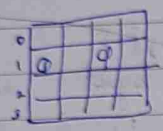


check → row, upperDiagonal, lowerDiagonal → isSafe()



move to check in this direction
kyuki right me queen hi nhi hai

row check →
mapping ✓

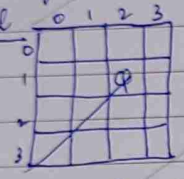


→ false

map < rowIndex → boolean >
place → Index → T
safe → checkIndex → T/F

in O(1) complexity

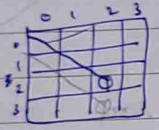
lowerDiagonal



rowIndex + colIndex → same
i+j → map

map < row + col → boolean >

upperDiagonal



i-j → same

map < row - col → boolean >

DOMS Page No. Date / /

Dnc and Backtracking class-4 (Live) 11 July 2024

(youtube motivation 20 min)

leetcode 37.

Q. Sudoku solver

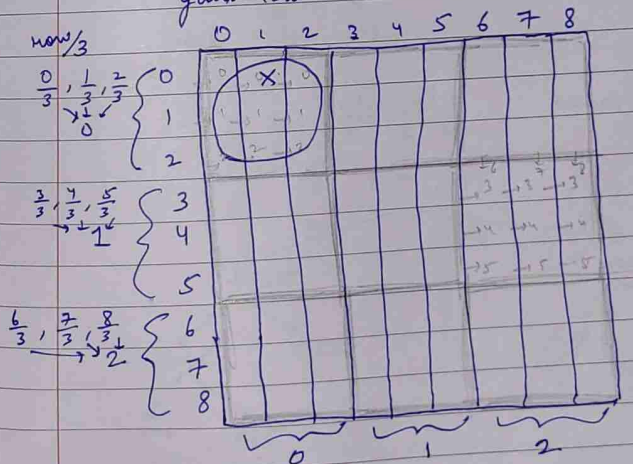
① row : 1 → 9 digits
② col : 1 → 9 digits
③ 3×3 : 1 → 9 digits

eg, 4	5	?	8	2	6	?	9	7
8	9	2	5	7	1	6	3	4
1	6	7	4	9	3	5	2	8

1 2 3 4 5 6 7 8 9
× × ✓

ek case hum baki case recursion → no need of base case (board fill hojayega fir recursive call jayegi hi nhi)

• agar kuch bhi na daal paao 1-9 tk matlb phle kahi place krne me galti kri hai → Backtrack kro.



row/3 → 0/3, 1/3, 2/3 → 0, 1, 2
3/3, 4/3, 5/3 → 1, 1, 1
6/3, 7/3, 8/3 → 2, 2, 2

row/3 ⇒ sub-matrix konsi row mai hai pata chal jayega

0+0 → 0th row
0+1 → 1st row
0+2 → 2nd row

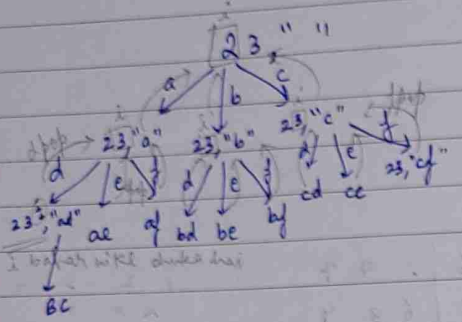
TC: exponential

need 9 times row index for each row-zone
⇒ 3(row/3) + (k/3) ⇒ 3 × row/3 + k/3

for col index
3 × [col/3] + k%3

☆ (asked in companies)

Leetcode 17. Letter Combinations of a Phone Number



2 → abc
3 → def
4 → ghi
5 → jkl
6 → mno
7 → pqrs
8 → tuv
9 → wxyz

LC 1079 → HW (Socrdem ki Kha hai par hard hai)

Recursion Mega Class (Live)

14 July 2024

TLE

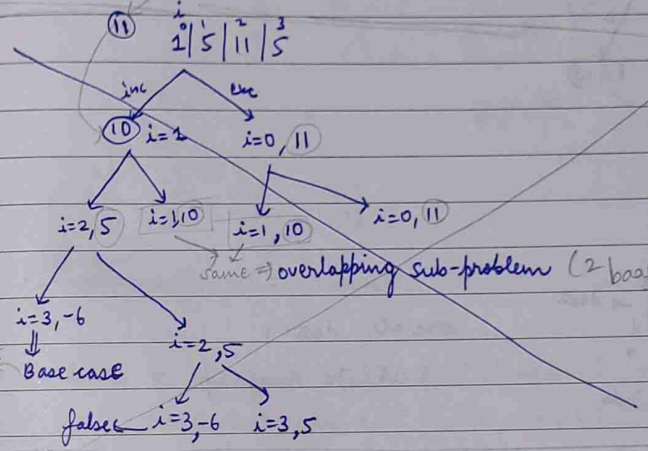
Leetcode 416. Partition Equal Subset Sum

eg: | 1 | 5 | 11 | 5 |
0 1 2 3

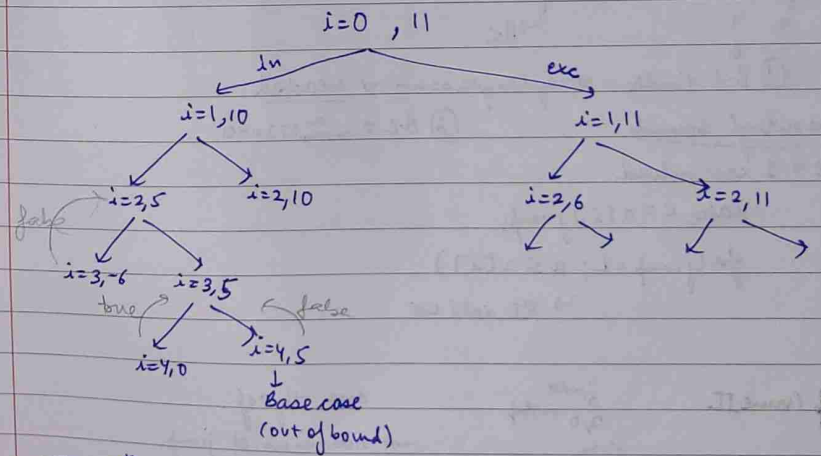
→ Include / Exclude

Target Sum = 11.

⇒ ① sum the elements
② if (sum % 2 ≠ 0) return false (odd)
③ target sum = sum / 2
④ find a subset with subset sum = sum/2.



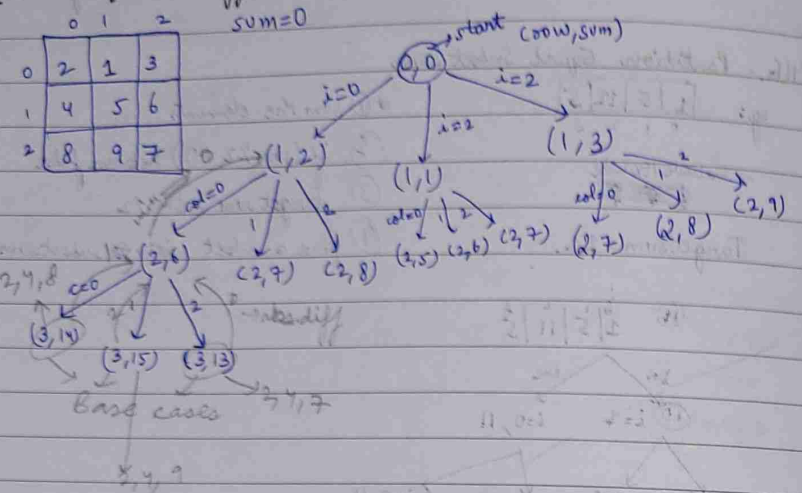
1 | 5 | 11 | 5 target = 11
0 1 2 3



TC: 2^n → exponential

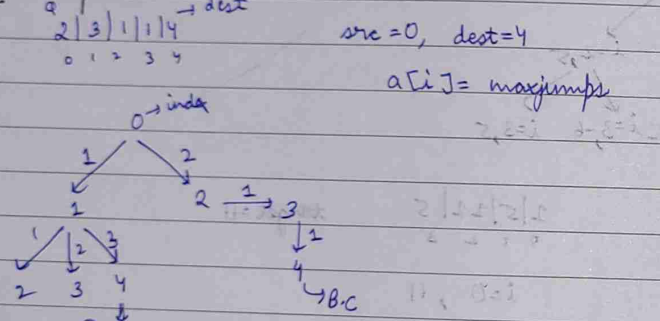
TLE

Leetcode 1981. Minimize the difference between target and chosen Elements



TLE

Leetcode 55. Jump Game



① B.C ⇒ index = arrayLength == n-1 ⇒ reached

② B.C ⇒ out of bound

③ B.C ⇒ a[i] == 0

RE ⇒ 1 case solved

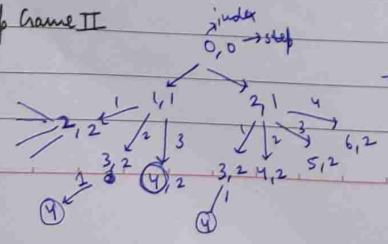
take c = a[i] jumps

for(jump = 1; c ≤ a[i])

↳ RE call

TLE

Leetcode 45. Jump Game II



ans → by ref.

⇒ return no. of jump

TLE

Leetcode 120. Triangle.

⇒ minimum Path Sum.

row 0 → 2 → 0 → i

row 1 → 3 4

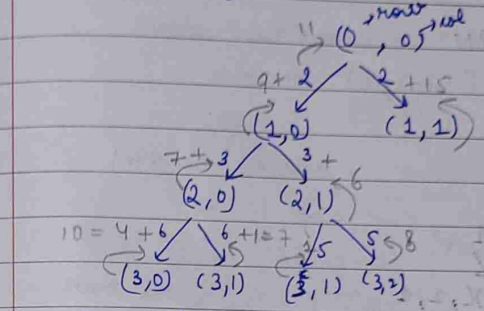
row 2 → 6 5 7

row 3 → 4 1 8 3

i = col

i, i+1

i



Base case (row == n rows - 1)

1 case ⇒ current value + RE(col, row+1) → A

⇒ current value + RE(col+1, row+1) → B

return min(A, B);

TC: 2^n

SC: O(n)

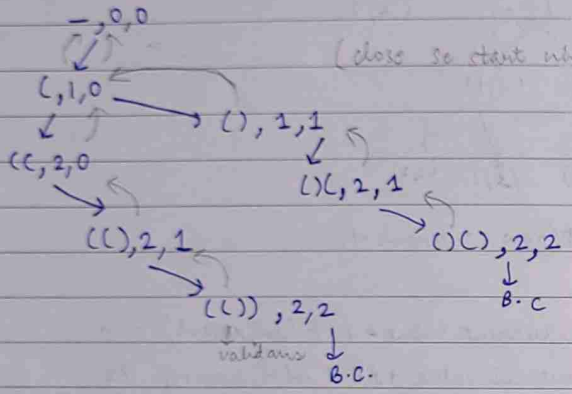
D&C and Backtracking Mega class (LIVE) — 21 July 2024

Leetcode 22.

① Generate Parentheses

eg: N=2 ⇒ total=4 ____

- open=0, close=0
- Base case ⇒ open + close = 2N



(close se start nhi kar skte)

① if (open < N) → '(' push
↳ open call — pop

② if (close < open)
↳ close call → ')' → push
pop

restriction

Leetcode 240.

② Search a 2D matrix II

→ Binary search Iteratively
- find Movement to Row or col

Leetcode 77.

③ Combinations

⇒ N, K

eg: N=4, K=2

[1, 2, 3, 4]

K _ _
K0 K1

→ K pick

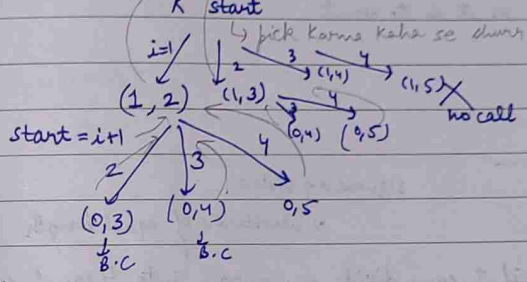
[1, 2 - - - N]

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

B.C ⇒ if (K==0) {ans.push(curr)}
if (start == n+1) [not needed wise]

vector<int> curr; → make combinations in it

curr → [1,2,4], [2,3,4], [3,4],



↳ pick karna kaha se aur kha tak

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

TC:

Leetcode 79.

④ Word Search

A	B	C	E
S	F	C	S
A	D	E	E

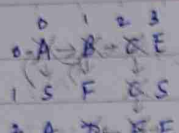
word = (ABCCED)

```
for (rows) {
    for (cols) {
        → har position se start krega
    }
}
```

B.C

```
if (row < 0 || row >= B.size() ||
    col < 0 || col >= B[0].size() ||
    B[row][col] != word[i])
```

visited : → 2D matrix
or
↳ Board to visit



⇒ ABCCED

→ handled by Base case

word[i] == B[r][c]

⑥ if i == word.size() ⇒ return true

① B.C

② visited mark

③ RE down → row+1, i+1

④ unvisited mark

leetcode 52

⑤ N-Queen-II

⇒ use N-queens I as it is to count

→ TLE (aise hoga yega)

leetcode 473.

⑥ Matchsticks to Square

eg: [1, 1, 2, 2, 2]

square ⇒ 4 side = 4

⇒ all sides are of equal length

↳ I can make square if I can divide my array into 4 equal subsets.

① arraylength < 4 ⇒ [2,2,2]

return false

② sum % 4 != 0 ⇒ sum not divisible by 4. ⇒ [1,2,2,2]

⇒ 7 % 4

return false

③ sideSum = sum/4;

Try to divide array into 4 equal subsets with sum = sideSum)

⇒ array ⇒ [sideSum, ss, ss, ss]

↓ A ↓ B ↓ C ↓ D

↙ Include / Exclude

↓

at the end → [0, 0, 0, 0]

To remove TLE ⇒ sort in decreasing order (large no. pehle aajayega, toh base case jaldi hit hojayega)

eg: [1, 1, 2, 2, 2] → 8

([2,2,2,2], 0)