HEAP - II

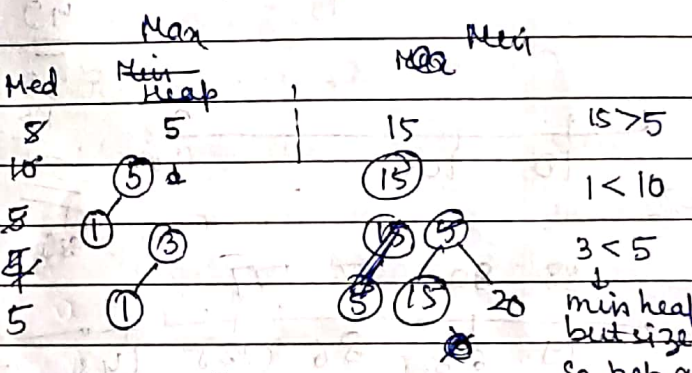
(Q) • HeapSort
n times do heapify.

(Q) Given an array which is k sorted.
6, 5, 3, 2, 8, 10, 9
each element is moved at most k places k=3
• build a min heap of size k+1,
move the window pop one and push one.
~~O(k) + O(n lg k)~~ (n-k) deletions
O(k) + (n-k) lg k

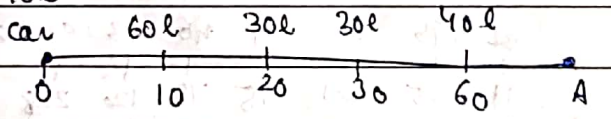
(Q) Given an array →
1, 23, 12, 9, 30, 2, 50.
Find 1st k ~~min~~ max elements
i) Build max heap → O(n) + k lg n
optimize
② Build k-sized min heap.
if we encounter greater than top then replace. otherwise ignore.
Finally we will get k largest elements.
intuition → keep track of k elements and whenever a greater element comes replace with smallest.
O(k) + (n-k) lg k.

Online algo → gives answer on each input.

(Q) Median of Running Stream of Integers
Here we are only bothered about middle elements ~~as a set~~ (ordering of rest are not matter).
• largest of lower part (Max Heap)
• Smallest of the upper part. (Min Heap)
|size of left − size of Right| <= 1

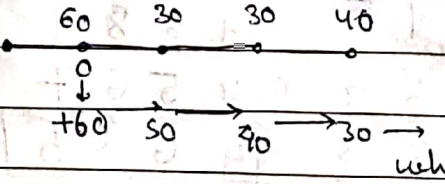


compare with current median and push to the reqd. heap accordingly.

(Q)


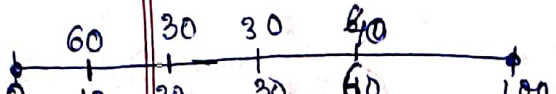
← 100 m →

Pos = [10, 20, 30, 60]
F = [60, 30, 30, 40]
1 liter = 1 unit distance.



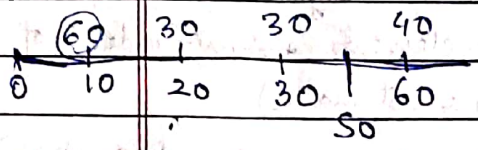
when becomes 0 then we can get the largest among the visited options

max heap



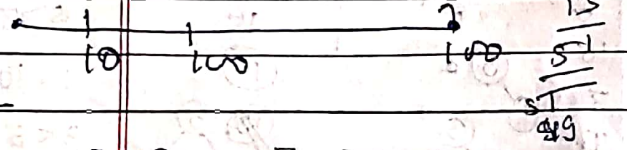
st=0 0 50 40 10

sum 10 20 30 60

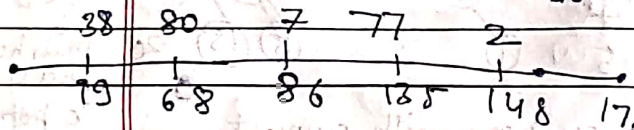


st=10 0 50 40 10

sum=10 10 20 30 60

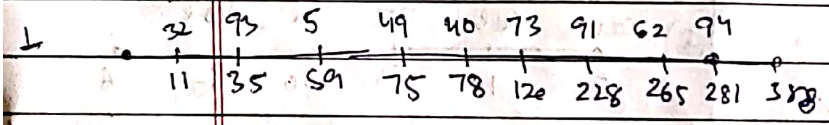


st=2



st=32 13 2

19 68



st=14 3 11

sum 11 35

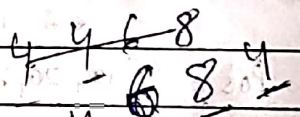
1 2 3 4 5

1 1 2 2 3

3 3 4 4 5

1 1 2 2 3

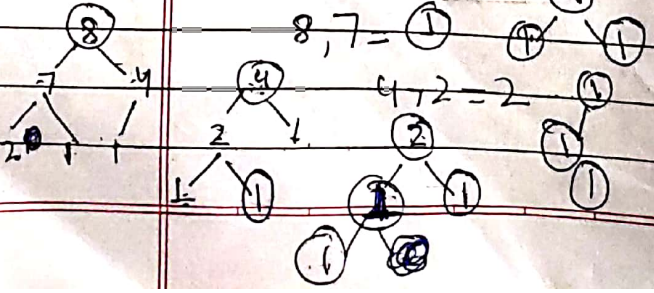
5 4 4 3 3



4 6 8 4

4 5 6 7

4 6 6 5



GREEDY

Date
Page

(a) k^{th} minimum in a sorted matrix.

→ do in $k \times k$ will never lie beyond k

10	20	30	40
15	25	35	45
24	29	37	48
32	33	39	50

k=3 → 20.

My approach → go to required row and then use min heap

k=3 → row = 3/4 min = 3 v. 4.

Suppose k=7 → row = 7/4 min = 7 v. 4

✗ ↑ wrong. 3

create heap of first row.

and pop k times whenever you pop → push the next column element in the heap.

$O(n) + O(k \lg n)$

~~further modified~~

$O(k) + O(k \lg k)$

- create the heap by first k elements of the 1st row then do the above.

(or)

push (10) pop (10) → contends 20, 15

(15) → pop → (15) → push (25, 24)

(25) (20) (24) 25 → then pop 20 and push the next contends

~~most~~ more optimal Sol^n

-1 -2 -3 0 1 2 3 | 4 7 8 12

(Q) Given array
$-2, 0, 5, -1, 2$
$K = 4$
do K negations
after K ops → maximize Sum.
we can negate same element twice.

- Myst app → min heap → negate top and push back to min heap. ✓ AC
  ↳ $K \lg n$

in $O(1)$ space. but $n \lg n$.
keep negating and keep track of min element.
↳ keep negating the min element. (if +ve encounter)

(Q) • Minimum cost of Ropes
• A, B → K min sum pairs.
• Min. possible max after K ops.
↳ giving immediate benefit
Greedy approach
↳ if local optimum is global optimum → then greedy would work.

(Q) Activity Selection.

S:	10	12	20		→ (2)
E:	20	25	30		

at any time t → you can do 1 activity.
find maximum activities you can do.

S: [1 3 0 5 8 5]
e: [2 4 6 7 9 9] → (4)

Date
Page

• sort according to end times and select if curr start >= Prev end time.

(Q) given an array →
$-1, -1, -2, 4, 3.$ = -24
↳ Min subset product.
$-5, 2, 1 \rightarrow -10$
$-5, 0 \rightarrow -5$
$6, 0, -2 \rightarrow -12.$

① • odd negatives and all +ves.

② count -ve elem
odd → multiply except 0
even → multiply leave the max -ve element
else,
↳ all positive then min value

(Q) Job Scheduling :- ↳ most frequently asked.
n jobs with their deadline and profit

Job	deadline	Profit
a	4	20
b	1	10
c	1	40
d	1	30

each job can be done in 1 day.
but before deadline day.
Maximize the profit.

- sort according to pair <dead, profit>

- sort according to pair <profit, dead>

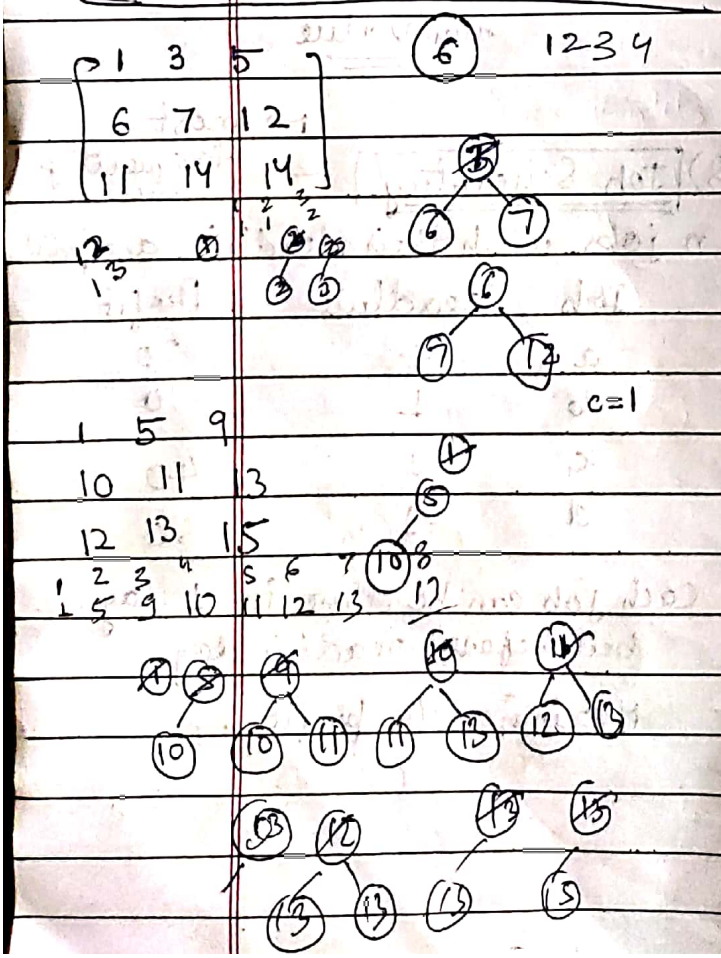
fill on the deadline and search for next available slot

- $O(n \log n + \frac{d^2}{2})$ → can be optimised using sets
  d is no. of day.
  for each day you need to search for an empty slot.

- more optimized → use disjoint data set

~~maxi~~

Set → maximum value is lesser than the given value.



Max Xor Pair in Array

first move left when 0 comes, move right when 1 comes
So trie node would basically contain *left and *right.

```
struct trieNode{
    trieNode* left; *right;
};
void insert (int n, root)
*cur = root
for (int i = 31; i >= 0; i--)
    b = (n >> i) & 1;
    if (b == 0)
    { if (!cur→left)
        { cur→left = new trieNode()
        }
      cur = cur→left
    else
    { if (!cur→right)
        {cur→right = new}
      cur = cur→right;
    }.
```

How to find max xor for given n.
find it's j^{th} bit and try to traverse to it's opposite bit if exists.

```
find max( head, a, n)
max_xor = INT_MIN
for(i = 30; i < n; i++)
{ int val = a(i)
  *cur = head;
  for(j = 31; j >= 0; j--)
      b = (val >> j) & 1;
```

```
if (b == 0)
{ if (curr->left){
      curr_xor += pow(2,j);
      curr = curr->left
  else
      curr = curr->right
else
{ if (curr->right)
      curr_xor += pow(2,j);
      curr = curr->right)
  else
      curr = curr->right)
if (curr_xor > max_xor)
      max_xor = curr xor
return max_xor
```

1) A. → N
2) AB. → Y
3) ABB → N
4) A.BB → Y
5) A..BB..B → N
6) A.B..BBB. → Y
7) AB. .... → N
8) A.B.. BBBB. BB → Y

4) AB.B ✓
5) ~~AB.B~~ AB.B..B.. ×
6) AB.B.B.B. ✓
7)
8) AB.B.B..B..B..

count of B >= ceil (n-1)/2
&& <= (n-1) → Yes

A........
A.........

lent
45 20 4 50 18 45 7 19
x==0 → return sum.
S → if xor becomes bigger.
P → if xor becomes smaller.
S → sort in descending
P → sort in ascending.

- Min swaps to bring all 1's together
get the indices of all 1's
and find the (population center)
of all 1's.
then moves all ones to that group.

Q → (Seats) Greedy

(Q) Binary Strings
find the 1st 0 and flip next
K → then again. and count
↑ my approach

000 10110
11110110
11111000 →

- Bulbs
1001010

392 94693

A B.B. B..

4260958 8662

4141154468

1 1/3 < 1/2, 3/2 ; 1/2 < 1/3, 2

426095 ✓

Stacking Cubes

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

6 → 3, 1,2,3
7 → 1
8 → 3
9 → 2
10 → 3, 1,2,5,10

Zee + Soy + Color + Start

Date
Page

(a) distribute Candies

1 2 3 2 4
1 2 3 1 2

1 2 2 3 4 3 2 1
1 2 1 2 3 1 2

Challenge

N - items M wins

Terminal - either to contestant or guard.

- Stability → S_i of terminal i
- D_G → guard removing cost = W_i
- D_C → $S_u \cdot S_v$ (of contestants)

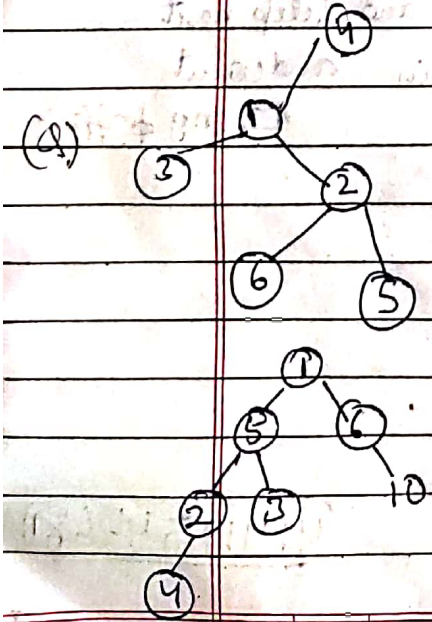
total cost of removing cables.

N M

S_i → States

W_i → cost

query of the cal
remove no - cables
read 0 ix new $S_i := x$
1 i y → $W_i = y$
2 → no change



INTRODUCTION TO DP 1D

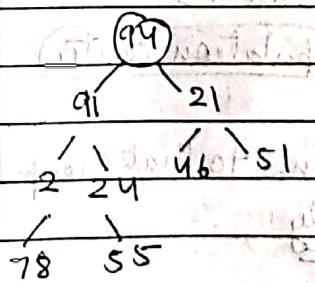
Those who can't remember the past are condemned to repeat it.

↳ Dynamic Programming.

1+1+1+1+1+ +1
=5
= 6 → because we remember the prev sum.

(a) Fibonacci - if we use recursion then we get $O(2^n)$ → exponential time
so go with dp.

1. Constructed a Recurrence Relation
2. Removing Redundancy.



(a) Stairs

in one step you can jump 1 or 2 step.

Find total ways to reach top.

$st(n) = st(n-1) + st(n-2)$

↑ because you can reach n^{th} step from $(n-1)$ and $(n-2)$

(a) Coin Change Problem →

n coins = [5, 10, 25]
Sum = 30

Min no. of coins of make the sum.

30 = 5+25

→ taking Max denomination first will not work

[9 6 5 1] → sum = 11
& ans = 2 [6,5]

Greedy approach will fail

$$f(30) = \min \begin{cases} f(25) \\ f(20) \\ f(5) \end{cases} + 1$$

(we know answer of these: f(20), f(15), f(0); min + 1)

⇒ Similar for all other

• now we can see Redundancy & come up with Recurrence and Remove Redundancy.

```
int minCoins(int s){
  if (s==0) return 0;
  int cmin = INT_MAX;
  for(i=0 ... M.len){
    if (M[i] ≤ s)
      cmin = min(cmin, Min Coin(s-M[i]))
  }
  return cmin+1;
```

(expon)

↓ to remove recursion use precalculated values and if it's present then return it else push the new value.

• Try to remove recursion by storing values.

Memoization (Building your cache)

(2) Tabulization

Start from small and build your table

(a) Given n

find min no. of squares to form this n

n=3 → $1^2+1^2+1^2 = 3$

n=100 → $10^2 = 1$

n=12 → $3^2+1^2+1^2+1^2 = 4$

$2^2+2^2+2^2 \rightarrow 3$ ✓

Search from 1 to $\sqrt{n}$ will become same as prev.

n=12 → [1,2,3]
→ [1,4,9]
∴ min no. to make 12

(a) Own a wine shop.

Sell 1 bottle per year.

Price = [1, 4, 2, 3]

each year - P(i) = y × A(i)
increase with year
because older the wine (costly).

only sell bottles which are at corners: ✗ ✓ ✗ ✓

[1 4 2 3]

$P = 1\times1 + 3\times2 + 2\times3 + 4\times4$
$= 1+6+6+16 = 29$

greedy will fail →

Eg → 2 3 5 14

$2\times1 + 3\times2 + 4\times3 + 1\times4 + 5\times5$
⇒ 2+6+12+4+25
⇒ 49 ✗ wrong

If $2\times1 + 4\times2 + 1\times3 + 3\times4 + 5\times5$
2+8+3+12+25
50 ✓

Recurrence Relⁿ

Zee Farub

P[0-4] = Max[P(0-3] + A(4)×y),
A(0)×y + P(1-4])

↱ Profit by selling (0-3)

P[0-3]

Max of each branch

A(0)×y + P(1-3)   P(0-2) + A(3)×y

A(1)×(y+1) + P(2-3)   P[1-2] + A(2)×(y+1)

⋮

P[0-N] = Max[A(0)×y + P(1-N],
P[0-(N-1)] + A(N)×y]

not 1D-DP

```
int Price (int s, int e) {
  if (e<s) return 0;
  return max( price(s+1, e, y+1) +
cache[s][e] = w[s]×y, price[s, e-1, y+1] +
    w[e]×y )
}
```

↑ same as the recurrence relation

How to convert to D.P.

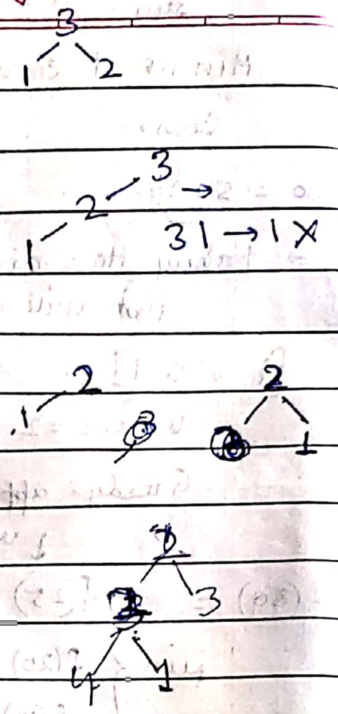
So keep a cache [ ][ ].

for price[s+1, e].
↳ check in cache [ ][ ]

else at the end

or

1 2 3 →
1 2 2
2 3 3
3 1 1
2 1 2
1 4 3



45 20 4 50 18 45 7
→ 27 34 50 50 36 45 49

328

DP-2 Dynamic Programming

(Q) very frequently asked →
given n houses → these are amount of money

[1] [2] [3] [4]

adjacent houses are connected

So if you rob house 1 then you can't rob 2. or if robbed 2 then you can rob 1 & 3.

Maximum amount.

My recur — a[s] + Max(a[s+2], a[s+3]).

M[i] — M[i] = Max[A(i) + M(i+2),
M(i+1)]

if you take iᵗʰ house → then skip one
if you don't take iᵗʰ then all the next houses.

Iterative approach

M[0] = a[0] → if only 1 house.

then by looking at the recurrence relation.

~~M(i) = Max(M(i-1), A(i)+M(i-2))~~

$M(i) = M(i-1) + M(i-2)$ ×

A(i) is already added

M(i, j) = (M(i-1), Max a(i) + M(i-2))

(Q) given 2 × n elements.

1	2	3	4
2	3	4	5

find max sum.

① can't chose V, H, D.

so ans → 8

My sol^n → choose greedily among the two no. of column. then create the recurrence rel^n.

global effect will be same

$$M(i) = \begin{cases} \text{Max}[\text{Max}(a[i], a[i+1])] + \text{Max}[M[i+2]] \\ M(i+1) \end{cases}$$

or.

find max from each column then new question of robbin.

2 3 4 5.

then do what you did in previous question

(Q) Given an encoding (ways to decode). A → 1, B → 2, C → 3 ... Z → 26

1 → A → L

12 → AB → 2.
↳ L

121 → ABL → 3.
↳ LA
↳ AU

12 → single
↳ double sum of both.

- take 2 at a time and keep checking single digit and double digit

count[i] = count[i-1] + count[i-2]

take the one if both valid (single and double)

↑ take 2

(Q) Party → can celebrate in two way → single or a pair.

Total no. of

n = 3 → 1, 2, 3 / (12), 3 / 1 (2,3) / (13) 2 = (4)

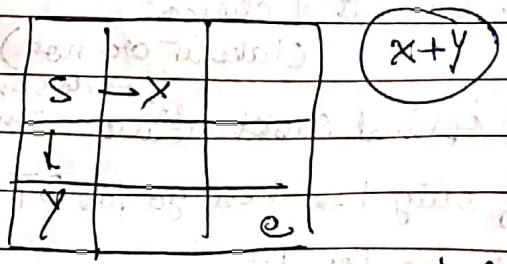
if you are alone

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

if you are pairing (you can select any one of the n-1 people)

(Q) total ways from (0,0), (n-1, n-1) if we use backtracking

BT two sums one starting from right and one starting from down



dp[i,j] = dp[i,j-1] + dp(i-1,j)

from bottom

(Q) Variation of previous

Obstacles in the given matrix.

if A[i][j] = 0 if obstacle is found

(Q) Now the grid has some values (cost matrix).

So, return the path with min cost.

1	2	3
4	8	2
1	5	3

Ans[i,j] = A[i,j] + Min(A(i+1,j), A(i,j+1))

(Q) intersecting Chords

1 — 1

2 → 2

3 → 5

4 → 14

5 → 42.

6 → 132

DP-3 → Dynamic Programming 3

Vivek is back (Yeah ...)

Kshitiz was also very good.

Symptoms — propes to Recursion

1. Element of choice (take it or not)

2. Optimal Substructure — do something on smaller to get larger problem

if only 1 → then go for BT

else go for DP.

1) Hardik ✓
2) GT82LC ✗
3) GT82LC ✓
10) Training ✓
11) A ✓
14) A ✓
15) E ✓
17) D ✓
19) A ✓
21) C ✗
22) E ✗
23) A ✗
24) B ✗
40) Self ✗

1) Yes ✓
2) yes ✗
4) note-given ✓
9) H ✓
10) F ✓
11) A ✓
18) A 18) A
19) C 19) C
20) A 20) A
20) D 21) D
22) E 22) E
23) F

2 1

00 01

10 11

000 001

100 101

000

100 001

010 101

next = nex

9

Listening

1 Hardik ✓
2 19 ✓
3 GT8268 ✓

21 A ✗ B ✓
22 B ✓ E ✓
23 B ✓ A ✗
24 D ✗ E ✓

30 A ~
3k competition ✓
40 worm ✗

18) A ✓
19) D ✓
20) E ✓
21) F ✓
22) J ✓

38) False ✗
39) Fals ✗
40) False ✓

18 19
20 21 2

A B D E
B D E

B C 18
A B → 0
B C
E
jubil

1 yes → no ✗
2 no ✓
5 yes ✓
7 no → yes ✗

18 A A ✓
19 D B
20 B C
21 F D ✓
22 J E ✗
B ✓
29
30 E ✓

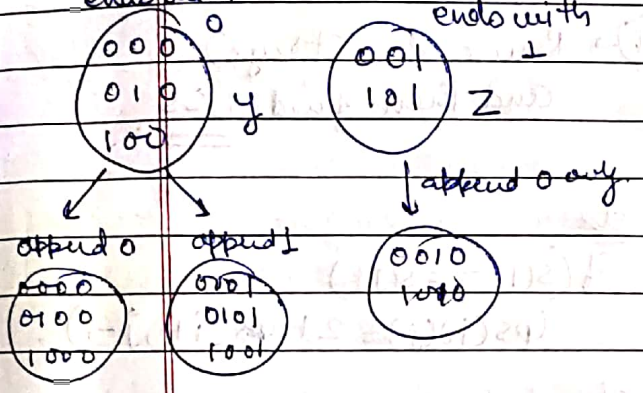
(Q) Given an integer 'N' return count of bit strings of length N without consecutive 1's.

N=2 → [00, 01, 10] → 3.

N=3 → [000, 001, 010, 100, 101,] = 5

you have ans for (n-1) ends with

N=3



ans[N] = 2y + z

keep track of y & z.

N-1

y ← ends with 0

z ← ends with

(N) temp1 ← y+z

temp2 ← y

ans = temp1 + temp2

y ← temp1

z ← temp2.

(Q) Given a Rod of N.

arr = contains prices of pieces <= N

Find max value obtained by cutting rod into pieces & selling.

N=8

L =	1	2	3	4	5	6	7	8
Price =	1	5	8	9	10	17	17	20

5 + 17 = 22

2 + 6

• My approach → coin change problem with max soln.

price[1] + ans(N-1)

price[2] + ans[N-2]

:

price[N] + ans[0]

} Max.

using table

for i ← to N

ans = 0

for j ← 1 to i

ans = max(ans, P[j] + Ans[i-j])

ans[i] = ans.

return ans[N].

(Q) Given 2 strings of size 'M' and 'N' Length of longest subsequence. [LCS].

S_1 → A B C D G H

S_2 → A E D F H R

ADH → 3

S_1 = ... a

S_2 = ... a → if i, j matches.

LCS[i][j] = 1 + LCS[i-1][j-1].

else

LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1])

because this would be the current running max answer.

i<0 || j<0 → return 0.

using Memoization

LCS

```
if i<0 || j<0
    return 0.
if (LCS(i,j) != NULL)
    return LCS[i][j].
else if s1[i]=s2[j]
    LCS[i,j] = 1 + LCS(i-1,j-1)
    return LCS(i,j)
else LCS(i,j) = max(i-1,j
                    i,j-1)
    return LCS(i,j);
}.
```

termination condition of Recursion is initialization of Tabular.

	"	A	B	A	D
"	0	0	0	0	0
B	0	0	1	1	1
C	0	0	1	1	1
A	0	1	1	2	2
D	0	1	1	2	3

Start from last
if mat ch → Push to stack
and move to i-1,j-1
else move to max (i-1,j)
(i,j-1).

Reduce space to O(N)

to print all the LCS
move to both max because max's are same.

(Q) given a string find the length of longest Palindromic Subsequence.

Ex.
BBABCBCAB
O/P → 7

① → Reuse the string and then find LCS.

else
```
if (s(i)==s(j))
    lps[i][j] = 2 + lps(i+1,j-1)
else
    lps[i][j] = max(lps(i,j-1),
                    lps(i+1,j))
```

initials → if i==j → return 1
or i>j → return 0.

then use memoization.

	0 B	1 B	2 A	3 B	4 C	5 B	6 C	7 A	8 B
0 B	1	2	2						
1 B	0	1	1	3					
2 A		0	1	1	1				
3 B			0	1	1	3			
4 C				0	1	1			
5 B					0	1	1		
6 C						0	1	1	
7 A							0	1	1
8 B								0	1

	"	a	b	b	c	d	g	f
"	0	0	0	0	0	0	0	0
b	0	0	1	1	0	0	0	0
b	0	0	1	2	2	2	2	2
a	0	1	1	2	2	2	2	2
d	0	1	1	2	2	3	3	3
c	0	1	1	2	3	3	3	3
g	0	1	1	2	3	3	4	4
f	0	1	1	2	3	3	4	5

i=2
j=
j+1=3

(Q) build cities $1 \to N$

$N = 5$

4		1
3		2
1		3
2		4
5		5
↑ random order		↑ Sequential order

Max bridges connecting same cities such that no bridges intersect

Find the LCS

Since other is sorted LIS would also be sufficient.

(Q) Find the length of Longest Palindromic Substring.

overhead

if $S_i == S_j$ and $Pal(S[i+1 \ldots j-1])$

$lpsstr[i][j] \leftarrow j - i + 1$

So, Keep a isPalin[N][N].

isPalin[i][j] → 1 / 0 (i,j) is a Palin or not

if $S_i == S_j$

if (isPalin[i+1][j-1] == 1)

isPalin[i][j] = 1.

	0 A	1 B	2 C	3 B	4 b
0	1	0	0	0	0
1	1	1	0	1	0
2		1	1	0	0
3			1	1	0
4				1	1

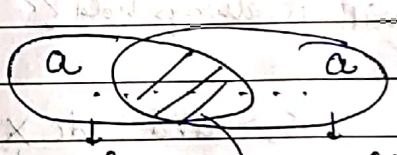
→ Pal(S[i+1][j-1]) will be done in O(1)

(Q) Sum of non-empty Palindrome substrings

Ans Sum of the isPalin Matrix

(Q) given a string 'aba' find total no. of non-empty palindromic subsequence.

$num[i][j] = num[i][j-1] + num[i+1][j] - num[i+1][j-1] + (S_i == S_j) * (num[i+1][j-1])$



num(i,j-1) num(i+1,j)

2 times so − num(i+1,j-1)

and if $S_i == S_j$

then add x

a ____ a

Pal

all pal b/w i+1, j-1 will also be pal if

a ____ a

x pal

DP-4 Dynammic Programming

(a) Given 3 strings S_1, S_2, S_3 find whether S_3 is formed by interleaving S_1 & S_2

interleaving - inserting S_2 in spaces of S_1

① find $LCS(S_1, S_3) \rightarrow S_1$ } if S_2 is interleaved
$LCS(S_2, S_3) \rightarrow S_2$

will it always hold??

$S_1 \rightarrow$ aab d
$S_2 \rightarrow$ ab dc
$S_3 \rightarrow$ aabd badc ✗ not

but
$LCS_1 = S_1$
$LCS_2 = S_2$

$S_1 \rightarrow$ ..... (i)
$S_2 \rightarrow$ ..... (j)
$S_3 \rightarrow$ ..... (k) → contenders of k are i and j

if ($S_3[k] != S_1[i]$ && $S_3[k] != S_2[j]$)
return false.

if ($S_3[k] == S_1[i]$ && $S_3[k] == S_2[j]$)
either $(i-1, j)$ or $(i, j-1)$

if $S_3[k] == S_1[i]$
$i-1$

i, j, k remove k as $(i+j)$
2 state dp

define $IL[i][j]$ → 0 / 1
$S_1[0 \ldots i]$ Date Page $[\ldots j]$
denotes
is $S_3[0 \ldots i+j+1]$
is interleaf of i, j

if ($S_3[i+j+1] != S_1[i]$ && $S_3[i+j+1] != S_2[j]$)
$IL[i][j] = 0$;
else if ($S_3[i+j+1] == S_1[i]$ && $S_3[i+j+1] == S_2[j]$)
$IL[i][j] =$ interleaf $(i, j-1)$ ||
interleaf $(i-1, j)$;
else if ($S_3[i+j+1] == S_1[i]$)
$IL[i][j] =$ interleaf $(i-1, j)$;
else $IL[i][j] =$ interleaf $(i, j-1)$;

Termination condition
if ($i == -1$ && $j == -1$) interleaf of empty str
ret 1
else if ($i == -1$) interleaf of empty str and $S_2(j)$
return 1 if $S_3(j) = S_2(j)$ $(0 \ldots j) = (0 \ldots j)$
else ($j == -1$)
ret 1 if $S_3(i) == S_1(i)$
$S_3[(0 \ldots i)]$ $S_1[0 \ldots i]$

↳ but it $O(n)$
So precompute Longest Prefix for S_1 S_3 and S_2 S_3

this was top down recursive approach.
Now do bottom-up tabular approach
and initialise using terminating conditions

		a	a	b	c	c
		1	a[0]	a1(1)		
a	a1(0)					
b	a2(2)					
b						
c						

a1 → contains boolean val whether ($S_1 == S_2$)

a2 → " " ($S_2 == S_3$)

(Q2) Given 2 strings

'word 1'

'word 2'

Find min no. of operations

word 1 → word 2.

ops → ① insert char at any pos

② delete at any point

③ Replace any char.

① horse

② ros.

horse → rosse → rose → ros

data rep del de

(a) S_1 → ..... (i)

S_2 → ..... (j)

if $S_1(i) == S_2(j)$

ans[i][j] = ans[i-1][j-1].

else ⌐ replace

min(1 + Ans[i-1][j-1]

{ 1 + ans(i)(j-1) → insert }

1 + ans[i-1][j] → delete

Terminate conditions ⌐ 0 ops reqd to conv. empty to empty

if i<0 & j<0 ret 0.

if i<0 & j>0 → j+1 insert ops.

return j+1

if i>0 & j<0 → i+1 insert ops.

return i+1

converting S_1 to S_2 in this manner is called Edit distance

do both Recur + Memoize

		"	r	o	s
	0	1	2	3	
h	1				
o	2				
r	3				
s	4				
e	5				

then fill by recurrence relation

Live examples of Edit distance

in case of addresses → if edit distance < threshold. then valid

in git → code changes

in general Text Processing.

(Q) Given Two strings S and T. count no. of distinct subseq of S which equals T.

S = rabbbit

T → rabbit

o/p → 3 → pair up all b's.

S = abacde

T = ac → (2)

abcde

S = ... (i)

T = ... (j)

if $s_i \neq T_j$ → move s back

ans(i)(j) ← ans(i-1)(j)

if $s_i == T_j$

element of choice → ans(i)(j) ← move s & T back
ans(i-1)(j-1) +
ans(i-1)(j)
↳ move only s.

Termination conditions →

if (j == -1) → T is vanished ret 1; so we found

if (i == -1) → S is vanished but T not found ret 0;

```
int getDSS(i, j)
if (j == -1) ret 1;
(i == -1) ret 0;
if (DP[i][j] != NULL)
    ret DP[i][j];
if (s[i] != T[j])
dp(i)(j) = getDSS(i-1, j)
else dp(i)(j) = getDSS(i-1, j)
              + getDSS(i-1, j-1)
```

		r	a	b	b	i	t
	⊥	0	0	0	0	0	0
r	⊥						
a	⊥						
b	⊥						
b	⊥						
i	⊥						
t	⊥						

↳ move forward by recurrence relation

Wildcard Matching

(Q) Given a String S and pattern P.

S = {a-z} matches with seq of ch

P = {a-z, ?, *} ↳ substr

? → matches with any 1 char

S = a d c c b

P = *a *b → True

" d c c

S = c b

P = ? a → False

S = a c d c b

P = a * c ? b

S → ..... (i)

P → .... (j)

if P_j is lowercase

ans(i)(j) = ($S_i == P_j$) && ans[i-1][j-1]

else if (P_j == ?)

Ans(i)(j) = Ans(i-1)(j-1).

else ↑ don't eat and decrement

Ans(i)(j) = Ans(i)(j-1) ||

Ans(i-1)(j) || Ans(i-2)(j) ||

↓ eat and be there ↓ eat 2 and be there

Termination

i == -1 && j == -1

ret T

i == -1 → return O(1) by precomp

ret T if P[0...j] = '*'

else 0

j == -1 → ret F

(Q) Wildcard II

now
$S = \{a-z\}$

$P = \{a-z, *, .\}$. matches any 1 char.

↑ matches preceding chars.

$S = aa$

$P = *a \rightarrow T$

$S = aab$

$P = c*a*b \rightarrow T.$

0c 2a b

Preprocess the String

$a*.* \rightarrow A$

$.* \rightarrow \$$

after Preprocessing

$a*.**bc*.*d$

$A\$bC\$d$

S . . . . ①

P . . . . ②

if (P_j is UpperCase)
$S_i - P_j == 32$
↳ $(i-1, j)$ || $(i, j-1)$

Ans(i)(j) ← Ans(i)(j-1)

if ($P_j == \$$)
take and stay
dont take and move.

Termination → $i == -1$ && $j == -1$
ret T

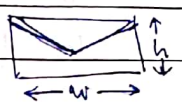
$j == -1$ → ret 0.

$i == -1$ → P(0 . . j) contains capital or $

Date
Page

DP-5 → Dynamic Prog-5

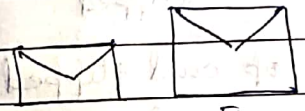
(Q) Given 'n' envelopes.



W[N] = { }

H[N] = { }

$E_i = [W_i, h_i]$



E_1 fits E_2 if $W_2 > W_1$ && $h_2 > h_1$

Find max no. of envelopes that can be put in one-another

envelopes which can be stacked inside.

$E_1, E_2 \ldots E_5$

W	5	6	6	2
H	4	4	7	3

I/P -

O/P → 3

$\frac{2}{3} \Rightarrow \frac{5}{4} \Rightarrow \frac{7}{6}$

$W_1 < W_2 \ldots < W_x$

$H_1 < H_2 \ldots < H_x$

} Variant of LIS in 2-D

↓
Something like LIP
↳ longest increasing pairs.

LIP[i] = max [1 + LIP[j]]
j < i and $h_i > h_j$
and $w_i > w_j$

To get the best of LIP
Sort the envelopes on the basis of area

(Q.) Given E's

	E_1	E_2	E_3.
W	1	9	5
H	3. 4		10

Now you have the freedom to flip the envelope.
So, flip 2nd.
ans → 3

LIP[i][0] → top and not flipped

LIP[i][1] → top and flipped.

so, it will be $\boxed{2 \times N}$.

$$LIP[i][0] = \max \begin{cases} \max(1 + LIP[j][0]) & j<i,\ w_i > w_j,\ h_i > h_j \\ \max(1 + LIP[j][1]) & j<i,\ w_i > h_j,\ h_i > w_j \end{cases}$$

↓ unflipped compared with previous flipped and unflipped

LIP[i][1]
↓
$h_i \times w_i$ and rest is same

or a simpler version
generate both the version and then sort them according to area then apply LIP as the previous qust.

→ easy to implement.

(Q) Given 'N' boxes (cuboid).
L[N] = { }
B[N] = { }
H[N] = { }.
Create a stack of boxes which is tallest.

• Note - same box (li, bi, hi) can be used multiple times

Gen → 6×N conf.

$$MSH[i] = \max_{\substack{j<i \\ l(j)<l(i) \\ b(j)<b(i)}} [MSH[j] + \ldots$$

if you generate only 3 configuration then you have to also check for the flipped version
so, in short (don't do with 6) do with 6)
ultimately space will be same

(Q) Given 'N' Matrices.
$M_1, M_2, M_3 \ldots M_n$
Then dim are such that $M_1 \times M_2 \times M_3 \ldots \times M_n$ can happ
$(M_1 \times M_2) \times M_3 \rightarrow op_1$
$M_1 \times (M_2 \times M_3) \rightarrow op_2$

find the series of Multiplication such that no. of op's is minimised?

ANS(0 . . . . 4).

(0-0)(1..4) + x (0-1)(1-4) x'. (0 2) (2 4) + x''

Date
Page

ANS(i...j) = 0 if i == j

else min [Ans[i...k] + Ans(k+1...j) + M[i].r * M[k].c * M[j].c]

i ≤ k ≤ j

int mult(i,j).
{ if (i==j) ret 0;
  else if (ANS[i][j] != NULL)
    return ans(i)(j).
  ANS(i)(j) → ∞.
  for k = i → j-1)
    Ans(i)(j) = min [Ans(i)(j), mult(i,k) + mult(k+1,j) + m(i).r * m(k).c * m(j).c].
  retun Ans(i,j).

O(N^3)

(2) Palindrome Partitioning

Min. no. of cuts so that each portion is a palindrome.

a|b a b b b a|b|b|a b a b a

min cuts = 3

Ans(a a b a)

Ans(a) + Ans(aba) + 1

Ans(aa) + Ans(ba) + 1

Ans(aab) + Ans(a) + 1

termination if (i...j) is Pal ret 0; ↳ do in O(1) time

→ Ans[i][j] ← 0 if S(i...j) is Pal.

else Ans(i)(j) = min [Ans(i)(k) + Ans(k+1)(j) + 1] i ≤ k < j

O(N^3) → same as Matrix Chain Multiplication.

(3) Given 'N' balloons, each balloon is associated with some no. of coins.

A[N] = { }

if you burst i^th balloon reward = A[i-1] * A[i] * A[j+1] and i is gone.

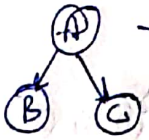
Find max profit by bursting them. consider 1 at both ends.

i/p → 5, 10

① ② → 50 + 10 = 60

② ① → -50 + 5 = 55

i/p → 1 2 3 4 5 → 110



not a connected component

Date
Page

↳ you can't reach from B to A or C to A

$A[k] * A[i-1] * A[j+1] + func(i, k-1) + func(k+1, j)$.

$$Ans[i\ \ j] = \max \begin{cases} a[k] * a[i-1] * a[j+1] \\ + \\ Ans[i\ \ k-1] \\ + \\ Ans[k+1\ \ j] \end{cases} \quad i \le k \le j$$

Termination

- $i > j$
- $i == j == 0$
- $i == j == N+1$

GRAPHS - I

data structure in set of (V, E)

$G[(B, C), BC)$.

edge can be bi/uni-directional

edge can be weighted or un-weighted.

N- nodes → Max edges → $^{N}C_{2}$

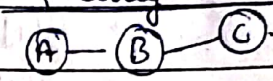
graph can have 0 edges

↳ (disconnected graph)

Connected Comp → you can reach to any node from any other node.

Representation

① Adjacency Matrix

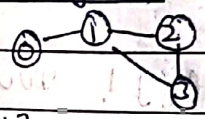


	A	B	C
A	0	1	0
B	1	0	1
C	0	1	0

Symmetric because graph is un-directed

space → $O(N^2)$.

② Adjacency list



0 = [1]

1 = [0, 2, 3] → more efficient

2 = [1, 3]

3 = [1, 2]

↳ vector of vectors > 2.

Graph Traversals

```
void DFS (int i)
{ print i;
  vis[i]=true;
  for all j in adjlist(i)
  { if !vis[j]
    { vis[j]=true;
      DFS[j];
    }
  }
}
```

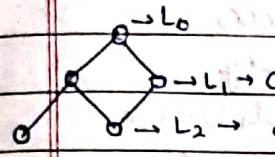
$O(V+E)$ or $O(\max(V, E))$

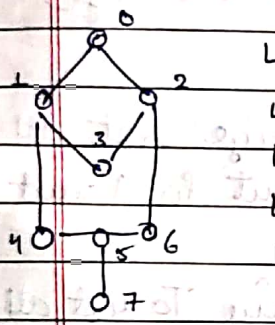
Now using stack →

(Q) given an undirected graph. check whether it's a connected graph.

- Run DFS once → and after that if any node is not visited means disconnected else connected.

← BFS Traversal → it traverses level by level.



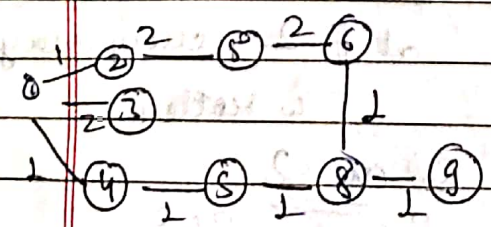


L_0 : 0
L_1 : 1 2
L_2 = 3 4 6
L_3 = 5
L_4 = 7

do this using a queue → good practice and better visual

- useful to find shortest distance in undirected graph.
However if the graph is weighted with different weights then BFS wouldn't work.

(Q) Given an undirected graph. $w = \{1, 2\}$.



modify BFS by inserting dummy nodes where d = 2.

Remember this → to create dummy nodes to make d=1. Then simply apply BFS to get the shortest distance.

(Q) Grid Problem →
Cell → 0 → water
→ 1 → land

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

Total 5 islands

- just a matter of connected components

(Q) 2D Board.
contains 'X' and 'O'.

X X X X
X O O X
X X O X
X O X X

if continuous streak of O's is surrounded by X's from NSEW then those O's would be captured.
- those O's will not be captured whose one of the O is at boundary so traverse only on boundary and trigger DFS on those O's.

(8) Given MXN matrix.
S: source → only 1 occurrence
d: dest. → only 1 occurrence.
* → safe
0 → unsafe.
moves NSEW

I/P →
$$\begin{bmatrix} 0 & * & 0 & S \\ * & 0 & * & * \\ 0 & * & * & * \\ d & * & * & * \end{bmatrix}$$

Find shortest Path from s to d = here d=6

first apply a simple BFS do create isSafe[]

(8) given NXM Matrix (Boolean).
for every cell (i,j)
find the distance of cell (i,j) from the nearest 1.
(Find the Manhattan dist.)

I/P →
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$
⇓
$$\begin{bmatrix} 3 & 2 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

insert all 1's in a queue and start the bfs.
(Multi Source BFS)
- check if not already visited
- check if it's 0.
- check if it's a safe.

(8) Given a Matrix
cells → 0 → empty
1 → fresh orange
2 → Rotten Orange.
NSEW
A rotten orange rots all it's adjacent in 1 unit time.
Find min Time To rot all oranges if Possible, else set -1.

$$\begin{bmatrix} 2 & 1 & 0 & 2 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 2 & 1 \end{bmatrix} \quad t=0$$
⇓
$$\begin{bmatrix} 2 & 2 & 0 & 2 & 2 \\ 2 & 0 & 2 & 2 & 2 \\ 1 & 0 & 0 & 2 & 2 \end{bmatrix} \quad t=1$$

at t=2 every orange is rotten
So ans = 2.
do multi source BFS because rotting is happening level by level.
- insert all 2's in a queue

edit distance Tabular

0 1 2 3 4 5 6 7
A N S H U M A N

A N T I H U M A N
0 1 2 3 4 5 6 7 8

		0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7	8
0	1	0	1	.				•	
1	2		0						
2	3			.					
3	4								
4	5								
5	6								
6	7								
7	8								
8	9								

4322

1 3 4
1 2 2 2 3 3 3 3 1111 222 33 44
1+6+12 = 19/8 = 2.35

27 3

39 8

79 3

Word Break - imp
in Table Manner.
if ~~dp(i,j)~~ S(i,j) is present
dp(i,j) = true

else make dp(i,j) true
iff ∃ k such that S(i,k) and S(k+1,j) belongs to dict.
else dp(i,j) = false

i=1 → (0,0), (1,1) (2,2).

what we are basically doing →
start with l=1
and check len 1 strings in dict.
then for 2 → if present then true
else break them and them.
↓ break point
then l=3 → if present → true
else break and check
2 break points

↑ → O(n³)

can be modified
for every length 1 to l.
keep checking all prefixes from 0→i
and if present mark them.

#Matrix Chain Multiplication

i==j → 0.
for(k=i; k<j; k++)
{ c = min(i,k) + min(k+1,j)
+ v(i).x * v(k).c * v(j).c.
if (c<cMin)
cMin = c
}
return cMin

GRAPHS-2

(8) We have n no's partioned in 'S' sets.

[1 3] [5] [2] [4,7,9] [8,10,6]
S_1 S_2 S_3 S_4 S_5

Union (9,3)

[1 3 4 7 9] [5] [2] [8 10 6]
S_1 S_2 S_3 S_4

So, perform Union (x, y).

1) find set of x
2) find set of y
3) Merge.

Union (4,10)

1) find 4 → S_1
2) find 10 → S_4
3) Unite S_1 & S_4, say S_1

- Every set is a connected component
So union op is adding 1 edge to create a bigger connected comp.

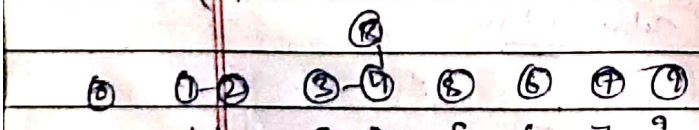
Union (1,2)

0 1-2 3 4 5 6 7 8 9

Union (3,4)

0 1-2 3-4 5 6 7 8 9
0 1 2 3 3 5 6 7 8 9

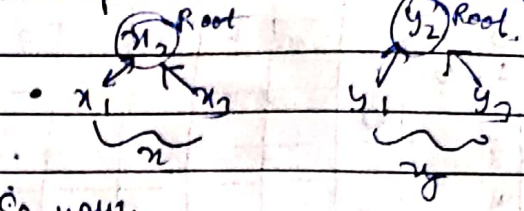
Union (4,8)



Union (x,y) → O(n) → exp
find (x) → O(1) → cheap

optimize union

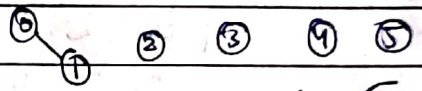
idea → we can have a representative element.



So now.

0 1 2 3 4 5
own tree
Par 0 1 2 3 4 5

Union (1,10)

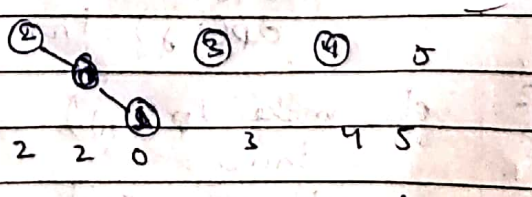


Par 0 0 2 3 4 5

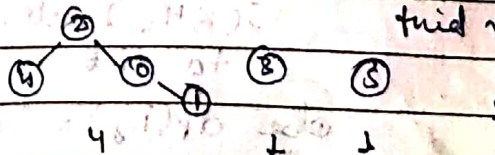
① Par(root(b)) ← root(a)
or
Par(root(a)) ← root(b)

- Smaller size tree will be merged with larger tree
This will be as balanced as possible.

How to find - keep finding parent until Par[i] = i
find α height
Union α find α height



Union (4,1) find root(4) → 2
find root(1) → 0
4 ≠ 2
So ...



```
int Par & size
int Par[A+1]
int size[A+1];
for (i=1 ← A)
{ Par[i] ← i
  size[i] ← 1;
}
int getRoot (i, Par).
while (i != Par[i]
{ i ← Par[i];
}
ret i;
```

an optimisation
↳ Path Comp.

Date
Page

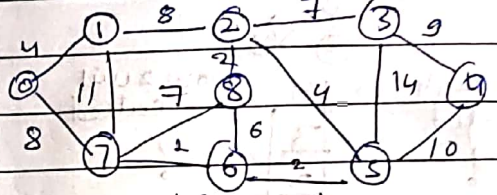
```
int find(i, j, Par).   → 1 → same set
                       → 0 → diff set.
ret getRoot(i, Par)
    == getRoot(j, Par);

void union (i, j, Par, size)
if (find(i, j, Par) == 1)
    return;
root i ← getRoot(i, Par)
root j ← getRoot(j, Par).
size i ← size[root i]
size j ← size[root j].
if (size i > size j)
{ Par[root j] ← root i;
  size[root i] += size j
}
else {
  Par[root i] ← root j
  size[root j] += size i
}
```

DSU → Disjoint Set Union.

Solves Tough Problems
Very Powerful DS,

(Q) Given 'N' cities
N = 9
[0 .... 8] Civil Eng

Plans to build roads→



bi roads

build the roads in such a way that all the cities can be reachable from each other and construction cost is min.

obs → NO cycles → useless
↳ means two paths

So the ans would be a Tree → no cycles and connected

Kruskal MST

- Sort the edges
- For edge (i)
  if it does not for cycle include it → ↳ use DSU

(Q) Given an undirected graph. Find whether there is a cycle

① DSU
② DFS →

7	→ x
6	if in adj(x)
1	then 7 is a
0	y and y != Par(x)

is Vis(y) ← true
↳ cycle

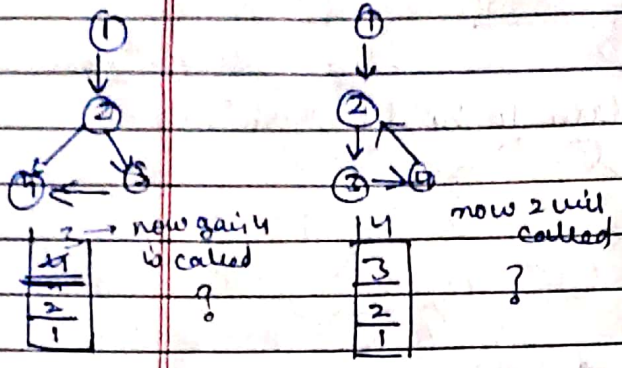
conclusion → not a parent but already visited →
parent → who called the given node.

in case of directed graphs.

DSU will fail

→ u

- DSU fails if graph is directed



back-Edge - pointing towards it's ancestor in DFS.

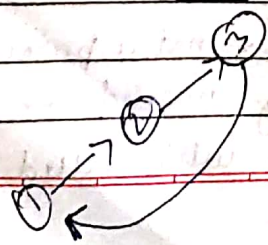
flag → 0 → not
→ 1 → visited
→ 2 → process (in the stack).

backedge if flag is processing

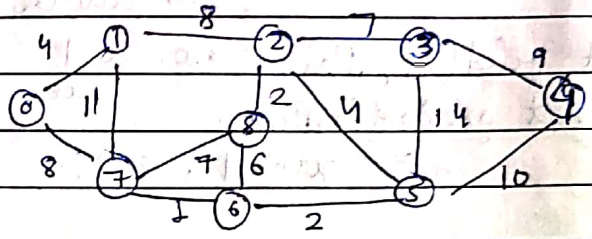
```
flag[N] ← 0
dfs(0);
int dfs(int i)
{  flag[i] ← 1.  // start processing
   for (j in adj[i])
   {  if flag(i) == 1 → return 1
      else if flag(j)==0
         dfs(j)
   }
   flag[i]=2;
}
```

BackTrack

do
Recursion
undo



(3) Given a graph → undirected
→ no self loop
→ only positive weight
given a node s:
find distance of all nodes from s → (Single Source Shortest Path) or Dijkstra



dist = [0 ∞ ∞ ∞ ∞ ∞ ∞ ∞]

extract min → j ← 0, dist[0] = 0

[0 4 ∞ ∞ ∞ ∞ ∞ 8 ∞]

extract min → j ← 1, dist[1] = 4

[0 4 12 ∞ ∞ ∞ ∞ 8 ∞]

4+11>8

extract min → j → 7 dist[7] ← 8

0 4 12

if done using simple array $O(V^2)$

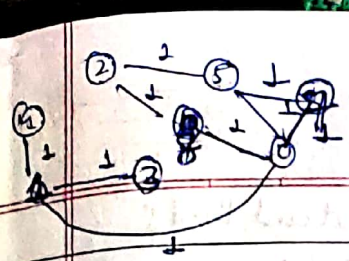
So painful part is extract min

↳ use heap

Heap → name
cur distance

also maintain position array

So that you can update the distances in the heap



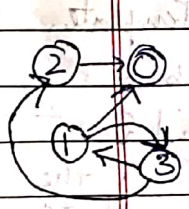
- no. of enter
- enter after Fill begin
- no use after C.
- only alphabet bold
- Table multiple of 3.
- bullet points

{ } { } { }

} } { { } { } { } { { }

Knapsack

recurrence → if weight of nth item > w
don't include
else → either include } max.
↳ not include

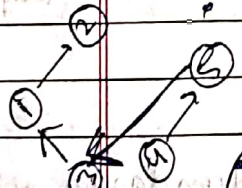


bold in Passage

audio → just

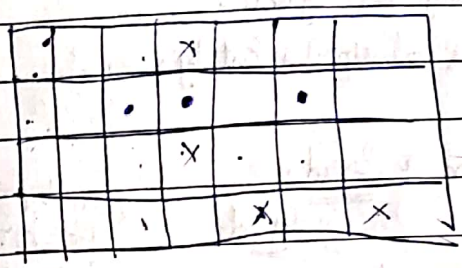
- bullet point probem align

justification in question



Knight on Chess Board →

8 cells.



5 → 2 $5/2 = 5$

$5 \% 1 = 5$ $4/1 = 4$

$4 \cdot 1 \cdot 2 = 0$

4 → 2 5 → 2

3 → 1 4 → 1

2 → 2 3 → 1

1 → 1 2 →

10 → 4

10 → 3

7 → 3 Ans

4 → 3

Date
Page

Scanned with CamScanner

LOW LEVEL DESIGN – I

LLD – I

Till now → Procedural Paradigm

↓ now Objected Oriented →

- everything is object.
- every Real World Entity.

(a) Design a Bird.

Object → Prototype / Blue print (general bird)

```
Public class Bird {
    double wt, ht;        // attributes
    String color;
    void fly()            // actions
    {
    }
}
```

eg. Hen

Object → instance / snapshot

```
Bird hen = new Bird()
Bird eagle = new Bird()
hen.fly();
eagle.fly();
```

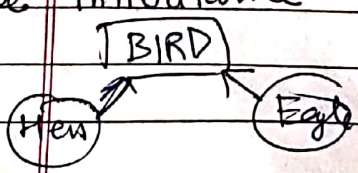
- Here same fly() fn can behave differently.

→ a naive way is to pass name of bird to fly fn.

↳ Problem → So many birds. Also code becomes dirty

So, How to restructure ??

use inheritance




```
class Eagle extend Bird {
    @override
    void fly() {
    }
}

class Hen extends Bird {
    @override
    void fly() {
    }
}
```

More better??

Flap wings → both → so move to parent → then call super.fly()

(common part is move to parent)

- Silent Killer

↳ a new dev didn't override fly() → everything compiled but unexpected behavior.

Think of this bird as Template.

now implement each bird

```
Peacock implements Bird {
    void fly() {
    }
    void sleep() {
    }
}
```

} the fn are already in temp

- but now code Redundancy is brought back.

How to remove Redundancy()

so now create a class flying util

```
public class flyingutil
{  public void common()
   {  Flap wings;
   }
}
void fly(){
   Flyingutil.commonfly();   // common to all birds
}
```

— bird specific

- Redundancy ✓
- No silent kill
- Not Dirty
- Poly Morphism

```
interface Bird {
   fly();
   eat();
}
class Peacock impl bird {
}
```

Bird is above Peacock

```
Bird b = new Peacock()
```

Correct and called Runtime Polymorphism

↳ comes handy → when handling all the birds.

↳ generic

Like if you want to sell birds.

```
class Seller{
   void Sell(Bird b)
```

↓ ↓ ↓

can be Peacock | can be eagle | can be albs.

still code is not Secure tampering with variables.

So use private → access specifier.

need getter, setter

Also called as Encapsulation (Data Hiding)

What is Business logic in code?

the if else conditions in the code is termed as Business logic → So data Hiding becomes really imp to make it run smoothly. It should not be tampered.

(Q) Given a codebase of 10,000 lines How can you explain it to a Tech/Non-Tech guy in 10 min ??

- Sample I/P (B) → Sample O/P (So-100 feet view of System)
- UML Diagram → Unified Modelling Lang.
  (1) Structural Explanation
  (2) Behavioral Exp.
- Structural → relation b/w classes. (Interaction)
- Behav → what it does. (I/P → O/P)

• ~~Structural~~ → use case → behavioral

use case → broader version of TC
↳ based on requirements

example → Library
(1) Search Book
(2) issue book.

• it defines the boundary of a system
↳ outside boundary will not be tested.
So, Test cases would be derived from use case

Search book → a string i/p

Flight
- Flight no : string
- dept. Airport, arr : Airport
- duration
- bookFlight()
- cancelFlight()

→ Attributes

Date Page

airport class → it will contain other information which will be encapsulated

airport
addr : address
code : string
name : String

address
addressline
• city
• state
• zip
• country

String

① Use Case Diagram

• Online Shopping System
• search product → custom
• Add to cart
• Check Out • Payment

include → Payment
checkout
search by user
search by category
search product
ext
Custo

parent use case

reused with extra fn

include → dependency
payment will not be done until payment is done.

(2) Flight Reservation System →

Physical → Customer, Flight

Flight
Flight No. → String
dep. : Airport
arriv : Airport
duration : int
List<FlightInstance>
GetInstances → returns list of instances

Airport
name : String
addr : Addr
code : String
GetFlights()

1-M
M-1

Return list of flight from only airport

different times of same Flight

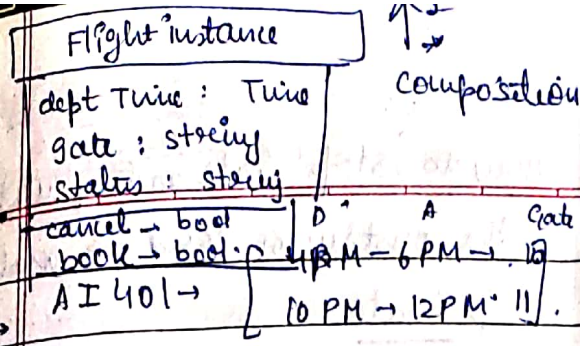
Associations

(Flight <-> Airport)

Many-to-Many

Dept Time
Gate
Status
→ Departure Info
↳ will change very frequently
So they can be modelled by in different class.

• Powerful structural → Class Diagram
• Static Structure of your system.
↳ will not change frequently.



Date Page MINE

Multiple instan → same flight

Many to one Relation

- if flight crashes → what would happen to flight instances.
- a customer books a flight instance.
- Pilot, Crew, Passenger

Crew *→* Flight instance → Many – Many

Pilot *→* Flight Instance. 2 → Many

bidirectional (Crew → get Flights, Pilot → get Flights)

ADMIN
- Add a Flight
- Add airport
- Add/remove Pilot

all are same forms of Person

all these will extends Person → name, phone, email, id.

Flight Reservation
Res No : String
Flight inst : FlightInst.
Seat Map
Status : Reservation status
getPassengerId() → list of <person>

1:58 Mid

LLD – 2

class Bird {
string col; int ht; int wt;
Public Bird (String color)
{ this.color = col
}
}

Bird b = new Bird ('Green');

Add 10 more attributes →

So you would do →

Bird b = new Bird (.....)

↳ very dirty.

↳ It's also backward incompatible change is not uniform along all previous version

↑ constructor overloading.

Still dirty → too many constructors. → 2^N combinations

How to solve??

- Telescopic constructor problem

How to solve??

① use of default values.

Static → uniform over all instances. remains same over all instances.

② create a nested class. (static)

```
Public class Bird {
    int ht, wt; string color;
    Public Bird (Builder b)
    { this.ht = b.ht;
      this.wt = b.wt;
    }
public static class Builder {
    int ht, wt ; string color;
    Public Builder() { };
    Public Builder ht (int ht)
    { this.ht = ht; return this }
    Public Builder wt (int wt)
    { this.wt = wt ; return this }.
    public Bird build ()
    { Bird b = new Bird (this)
      return b.
    }
}
```

↑ convert builder to build.

↳ instance of builder class

```
Bird b = new Bird.Builder().
    ht(5).wt(10).color('red').
    Build()
```

each dot is an instance of builder class.

a new dev comes →

who does

```
new Bird.Builder().
    ht(5), Build();
```

will work fine →

↳ cleaner code

(no issue of order mismatch)

. now if int area comes, I will add this to class but other dev know or not they will not face any issue.

How to stop interacting with the constructor Bird → make it private.

← Very popular design Pattern. (Builder Pattern)

How can we force client to pass certain variables must?

→ change the Builder constructor

```
Public Builder (ht, wt)
{ this.ht = ht
  this.wt = wt }.
```

Design Pattern → useful

- Stateful class / Stateless class

```
public class adder
{ public static
    int add (int a, int b)
  { return a+b;
  }
}
Adder.add(1,2)
```

↓

Stateless class

Every class will be same.

```
Public class add
{ int a, int b;
  Public int add()
  { return this.a + this.b
  }
};
Add a = new Add()
a.add()
```

↓

two objects will be different because of different state (different val)

So if we do lots different variable make them

Stateful → OOP. (you need to create obj)

stateless → procedural

Intro to

Industry Problem → Use case

① Address Validation →

```
public class AddressValidator {
    zipValidator z; StateVal s;
    public bool validate (Address a)
    { return z.validate (a.getZip())
        && c.validate (a.getCity())
        && s.validate (a.state())
        && l.validate (a.getAddrLine);
    }
}
```

• Validate each independently then move to next level.

• For each validator you'll need to establish connection with DB.
  ↳ Each time you do this
  ↳ too much time and poses a huge problem.

Maintain a ResourceInitialiser
  ↳ converts to a suitable DS.
and then create 1 instance and share it.

```
public class ResourceInit
{ private static ResourceTest
                INST = NULL;
  private static Resource getInst()
  { if (INST==NULL)
        INST = new ResourceInit()
    return INST.
  }
  private ResourceInit()
  { connect dB
    zips = read & parse zip
    cities = read & parse cities
  }
}
```

↳ will create only 1 instance

↳ now you have cached the reqd. info. → now no need to connect everytime

public bool checkIfValid(String s)
{ ret zips.contains(str); }

How to call this

ResourceInit.getInstance().checkIfValid(str).

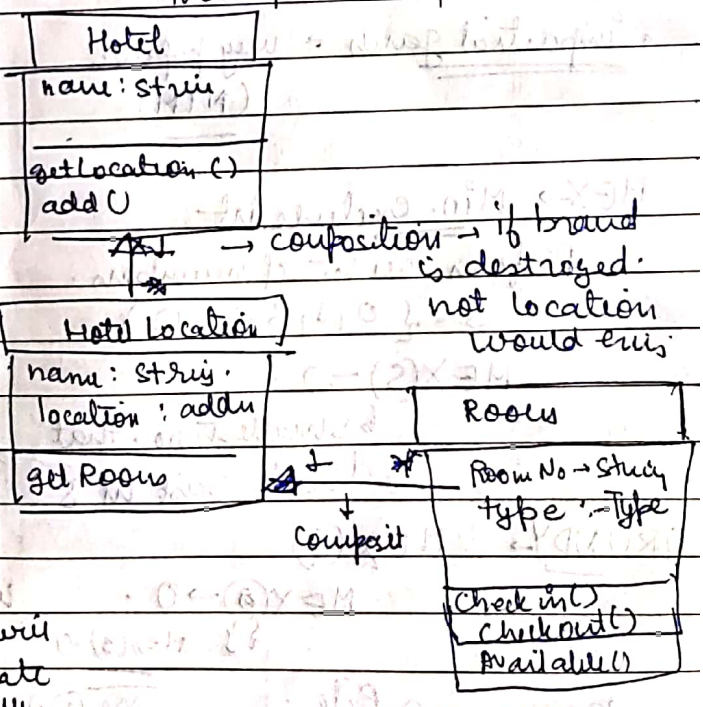
↳ 1 time connection established (everything cached)

• Singleton Pattern → offers 1 instance
  ↳ Singleton → only 1 instance
  imp → privatize the constructor.
  (Like memoization
  If NULL → create
  else use the already

② Hotel Management System

Imp → need to do

Objects → Hotel, staff, Room, Service, Address.



GRUNDY NUMBERS

if XOR(all set of positions)

↳ 0 → losing

→ 1 → winning

(8) Given 6 coins

Recursive.

Result(6) → XOR(gr(5), gr(4), gr(3))

XOR(1, 0, 3) → 2

↓

winning

Res(5) → XOR(gr(4), gr(3), gr(2)).

XOR(0, 3, 2) = 1

↓

winning

Res(4) → XOR(gr(3), gr(2), gr(1))

↳ XOR(3, 2, 1) → 0

↓

losing

• impartial games → very popular (NIM)

MEX → Min. excluded

Perform on set of numbers.

S = { 0, 1, 3, 8, 12 }.

MEX(S) = 2.

↳ smallest no. that don't exist in S

GRUNDY → MEX(S)

MEX(0) → 0.

{} Mex(0) → 0

Mex(1) → 1

10 coins in a Pile.

1 ≤ coins to take ≤ 3. Mex(2) 2

Grundy (10) = grundy(

Mex(gr(9), gr(8), gr(7)).

Find the grundy of base numbers on itself.

4 6 8 2.

Floor = { 2, 3, 6 }

G(2) = Mex{ 0, 0, 1 } = 2

G(4) = Mex(G(2), G(1), G(0)) =

= Mex(2, 1, 0) = 3.

G(6) = ~~Mex(3, 2, 1) = 0~~

Mex(G(3), G(2), G(1))

2. 2 1

G(3) = 2

G(3) = (G(1), G(1)

(1, 1, 0)

G(8) = Mex(G(4), G(2), G(1))

= Mex(3, 2, 1) = 0

So, XOR(2, 3, 0, 0)

→ (1) → so person who will start will win.

More would be to put opp. in losing

n = 10

k = 1 → D.

2 → A

3 → A

4 → D

5 → A

6 → A

7 → A

8 → A

9 → D

10 → A

if k = 1

even → D

odd → A

else

n = 9

k = 1 → A ✓

2 → D ✓

3 → A

4 → A ✓

5 → A

6 → A

7 → A

8 → D ✓

9 → A

10 → A

(Q) WORD Ladder

Find the min chain of transformation of words.

idea → use BFS.
adjacent words would be those who differ by only 1 char or 0 char

X——X——X

LLD-3

Till now → • Builder Pattern → for bulky
• Singleton Pattern.
↳ when need to initialize once.

• in different countries → different forms of address → Eg. Japan → Prefecture → Indian

```
public class INValidator {
    bool validate(Addres a)
    {

    }
};
```

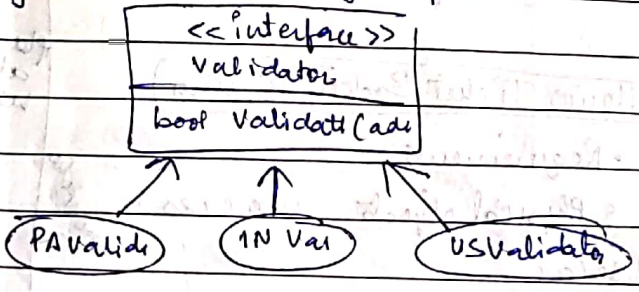
```
public class USValidate {
    bool validate(Addres a)
    {

    }
};
```

Date Page

```
public class ClientClass {
    bool validateAddres(Addres a)
    { if (a.getCountry == 'US')
        { return USValidator }
    }
```

(i) ↳ leads to so many if else condition → dirty.
(ii) client must know the name of all the classes.

• since all are validators → then we can use Polymorphism.



• use a map on my side.

```
public class X {
    private static map<string, Val> co
    public X() {
        Country Val.put('US', new USVal())
        Countval.put('IN', new INVal());
    }

    Public Validate getValidate(string c)
    { if !country.val.contains(c))
        { throw error 'not supported'}
      else return countryval.get(c);
    }
```

↳ this constructor is doing intensive job → may be some class is connecting DB (so it must be initialize once).
it must be singleton class.

```
Public static X getInstance()
{ if (INST == NULL) INST = new X()
    return X;
```

So the code will look like.

X. getInstance(). getValidator(a.getCountry). validate(a).

↳ must use try catch because there can be exceptions.

- Factory Pattern
  - ↳ provides different factory → just gives the name.
  - Singleton Factory Pattern.

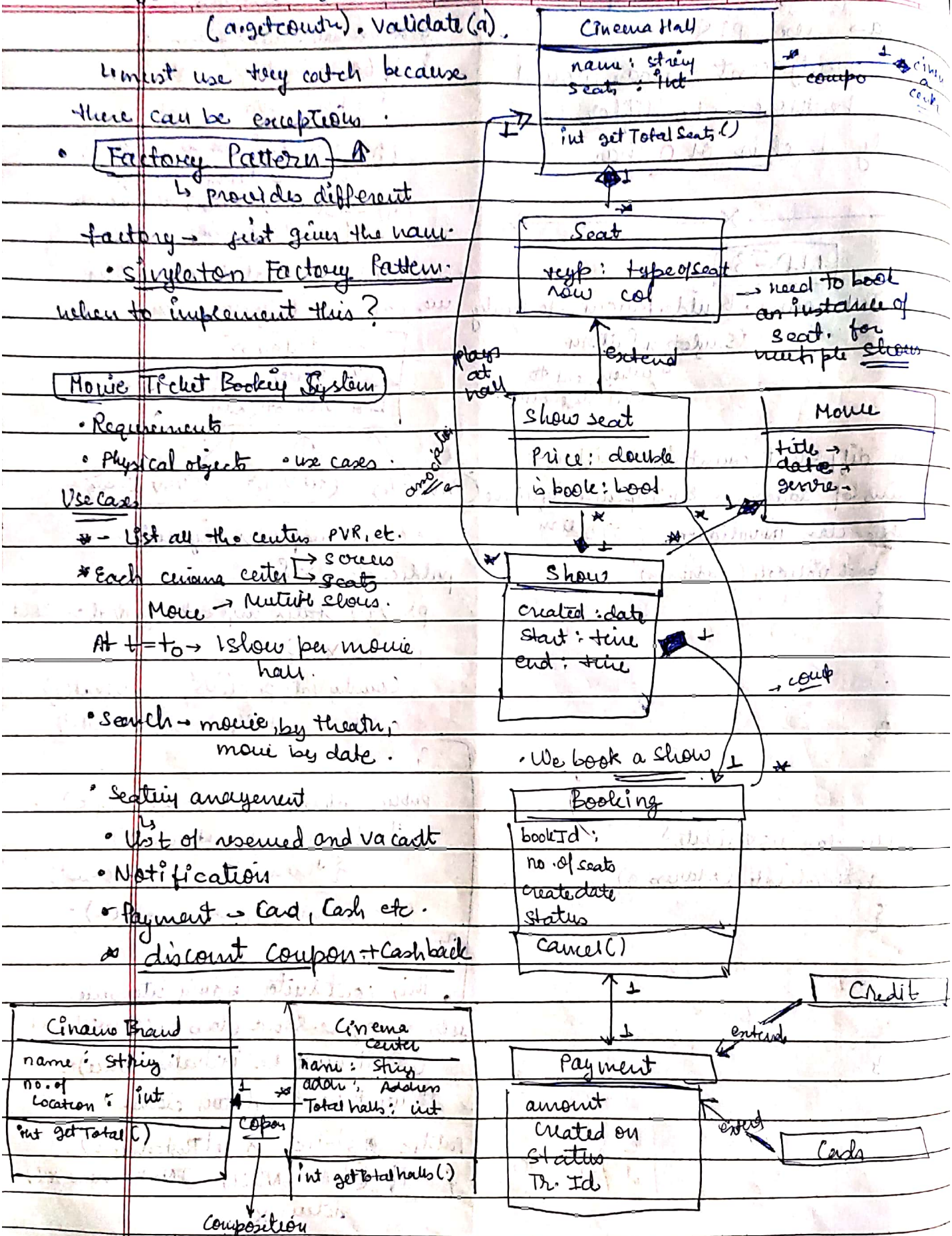
when to implement this?

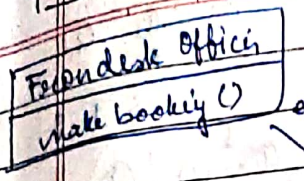
Movie Ticket Booking System

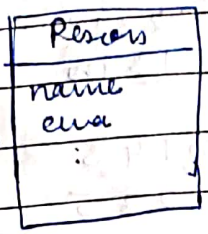
- Requirements
- Physical objects
- use cases.

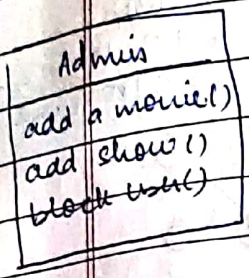
Use Cases

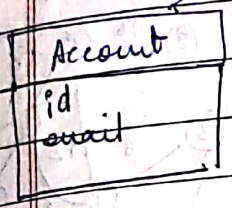
- List all the centers PVR, etc.
- Each cinema center → screens → seats
  Movie → Multiple shows.
- At $t = t_0$ → 1 show per movie hall.
- search → movie by theatre, movie by date.
- Seating arrangement
  - ↳ List of reserved and vacant
- Notification
- Payment → Card, Cash etc.
- discount coupon + Cashback

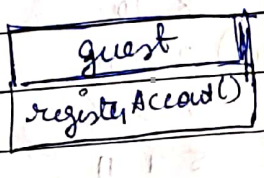


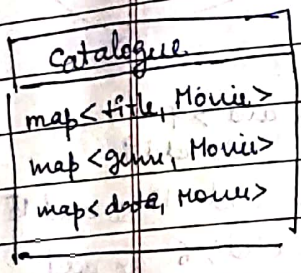


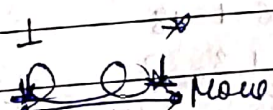












Car Rental System

- Requirements

① Support Renting of Car/Truck.
SUV Sedan

② Vehicle specific details → Veh. no.

③ charge late fee

④ Search and Reserve

⑤ Location of Vehicle

⑥ Pickup/Submit Notification

⑦ cancel.

⑧ ~~In car Servs~~
Extra services → in car wifi
→ extra driver

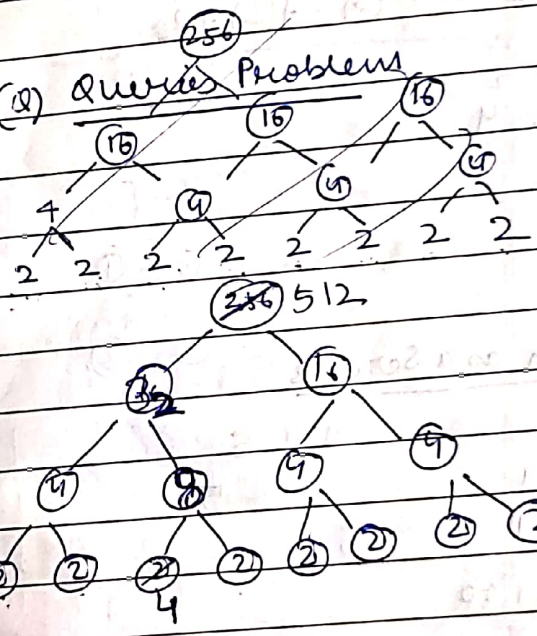
⑨ Retrieve
↳ which member took a particular vehicle
→ A particular vehicle was issued by which member

(Q) WORD SEARCH

C A A
A A A
B C D.

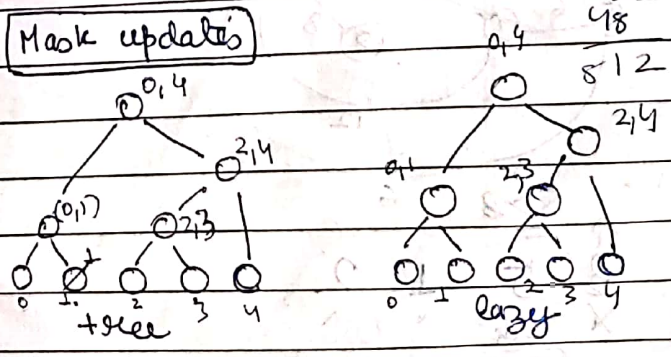
use backtracking
get 11 result from all 4 directions
and reset the markers for next.

(Q) Queries Problems

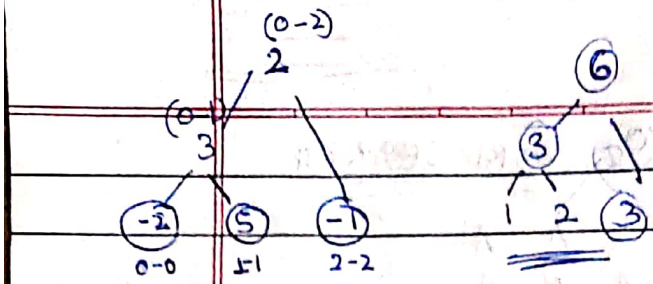


② → min X such that
a(i)^x .. a(n) - a(x+1). ∴ a(n) is ne

(Q) Mask updates



update (1,3)



$f(1) = f(2) = 1$

$f(3) = 3 \times f(1)$

$f(4) = 2 \times f(3) - f(2) + 2$

$f(4) = f(3) + 1$

$f(4) = 2 \times f(3) - f(2) + 2$

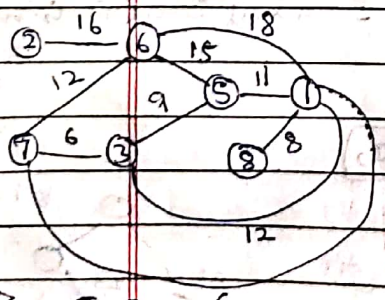
0 1 2
3 2 4

2, 1
3, 0
4, 2

• Graph as a Service

1 2 1
2 3 1
1 3 100

1 4 5
2 3 5



b> 5 > 3

3 5 1 2

X 2d (5 1 6) 2 > 3 + 1

A	B	B	B	A	B	B	B	B	A	B	A	B	B
a=1	1	1	1	2	2	3	3	3	4	5	5	6	6	6
b=0	1	2	3	3	4	4	5	6	6	6	7	7	8	9

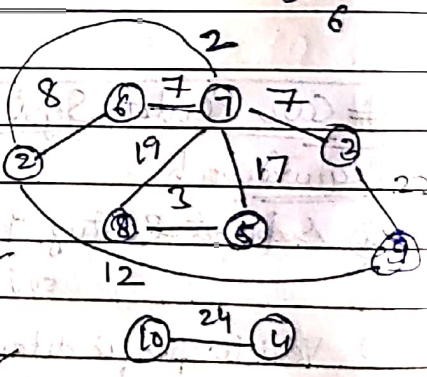
① 2 1 5 →

② 1 2 1
2 3 1
1 3 100

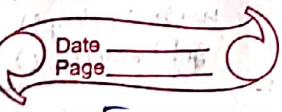
③ 1 4 5
2 3 5
4 2 4
8 10

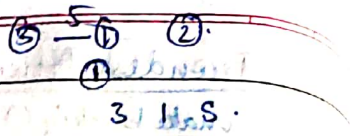
④ 2 6 16
7 3 6
5 1 11
7 6 12
8 1 8
6 1 18
1 3 12
3 5 9 → exists and > 9 so add 9
6 5 15 → exists and > 15 so add 15
7 1 6 → exists and > 6 so add 6
10 10

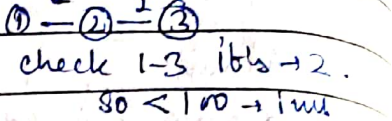
⑧ 6 7 7 ✓
8 5 3 ✓
3 7 7 ✓
16 4 24 ✓
8 2 8 ✓
8 7 19 ✓
3 9 22 ✓
5 7 17 → exists and > 17 so add 17
9 2 12 → exists and > 12 so add 12
27 2 ✓

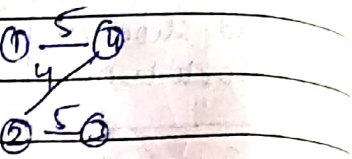


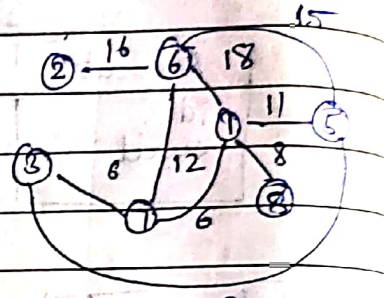
	A			
	B	A	A	B
a	0	2	3	B
b	0	0	0	1

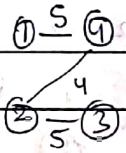












A A
A B B B B B
2+4+8 = 14

dp = 2 4 8 16 32

Date
Page

k=3 sets of 4

A B B B A B B B B B A B B

A B B B	B A B B	B B A B
B B B A	A B B B	B A B A
B B A B	B B B B	A B A B
	B B B A	B A B B

$2^6 + 2^7 = 64 + 128 = 192$

S 6 7

A B B B A B B B B A B A B B

k=3 sets of 3

A B B B A B B B B B A B A B B

A B B ✓ A B B ✓ B A B
B B B ✗ B B B ✓ A B A
B B A ✓ B A B ✓ B A B
B A B ✓

5 sets

k+1

B B B B B
A A A B B

③ $2^1 + 2^2 + 2^3$

A B A B B

128 + 256 = 384

A B B B A A B B A B A B B

B B B B B
5/2 → ②

k=2

Max Sum:

0	1	2
2	1	0
10	15	10
10	15	18
10	15	18
10	15	18

1 5 -3 4 -2

2
1
-1

bp =

A	B	B	B	A	B	B	B	B	B	A	B	A	B	B
1	2	3	2	1	2	3	2	3	2	1	2	1	2	1

Sum =

k=2

B	B	A	B	B
1	2	1	2	1

Sum = 2

k=1

B	A	A	B
	-1	0	1

B	A	A	B
0	-1	0	1

k=1

A	B	B	A
-1	0	-1	0

B

0	1	2	3	4
2	10	10	10	10

1 2 3 4 5

1 1 1 2 2 2 3 3 4 5

1 3 → 0 to 8 → length = 8

2 4 → 3 to 9 → 6

3 5 → 6 to 10 → 4

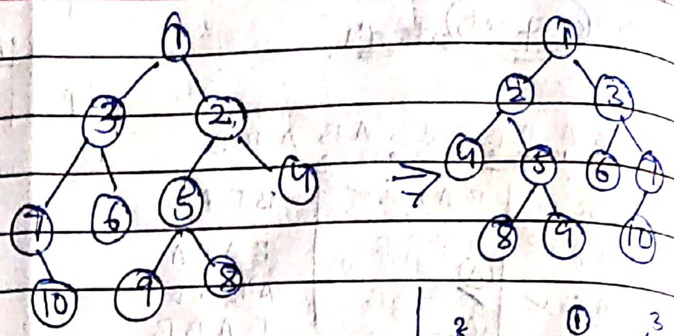
A B D   A C D

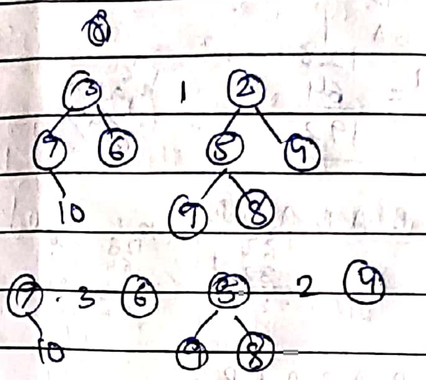
Changes to be done

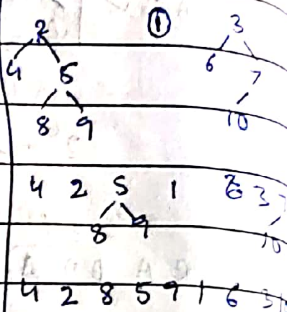
login
↳ $url → change.

Check login
↳ set credentials for mysql.
change redirect locations.

sample
↳ at start change url.







- url renamer
- login id valid → 6 months

Student own → institute their url

① Home Page → Intro
TCYonline.com → IELTS
Student   Institute

ILS test

| 1 Free test Sectional tests Buy Test | | Buy acc to Student ID ↳ url customized |

About us / Contact us / Home / ILS Test → Test

⑨ 10 3 6 1 9 5 8 2 4
4 2 8 5 9 1 6 3 10 7

1 3 2 7 6 5 4 10 9 8
1 2 3 4 5 6 7 8 9 10

8805331 62

LLD - IV

IB academy

Instructor
- id
- name

Mentees
- id
- name
- inst_id
- batch

1:n → Batch.

Topic
_id
- name
- [ ]

class check2 = ...
name = check2

class reach check

Mentee Topic
- id
- mentee-id
- topic-id
- attendance
- start time } redundant
- end time

Batch
- id
- start time
- end-time

checklogin
$testname

COMPUTER NETWORKS

- Network → Machines connected in some sort of fashion in order to communicate.

How would they talk??
A protocol is defined which is abided by both parties.

(a) Army 1 — Army 2

Send
Food1
Food2
Food n

How?
Me → send one by one and get acknowledgement.

- If horses doesn't return in a threshold resend for the food of that day.
- If few boxes were missing
How to know → Maintain a checklist.

lead to TCP protocol-

Src ——— Dest

bunch of packets
- out of order → Sequence No. will solve.
- may reach / may not).

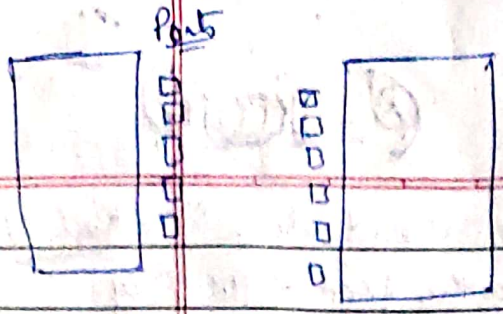
- UDP protocol
  - ↳ no acknowledgment → no guarantee
  - ↳ whether reaches on time or not
  - ↳ very light weight.
- Media streaming → UDP
  - ↳ if some data missed → let it be and move forward.

Working of TCP

① Establishes connection (3 way Handshake) (no conn in UDP)
- ① Ask
- ② get Reply
- ③ OK sending → starts here.

② Send packets → Seq No.
each has → checkSum.
wait for ack
if not in time → resend
ack is sent only if the packet is correct.

checkSum-
a very naive way.
checksum = 14 bits
and then check if 1 bits are 14. → but very inaccurate



Socket → IP + port

What happens when you type google.com

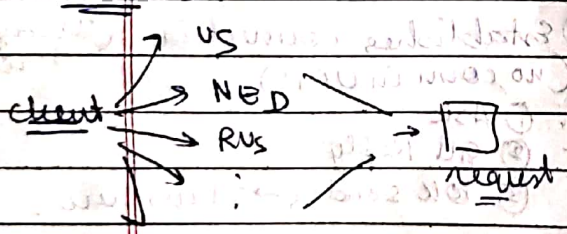
① DNS lookup → return IP
② go to that IP (TCP connection established).

ICANN → maintains legitimate info.

Browser DNS Cache → expires 1 day.
↳ no dns lookup if triggered again.

What was 2G, 3G scam?
Allocating a certain amount of frequency.

• TOR Browser



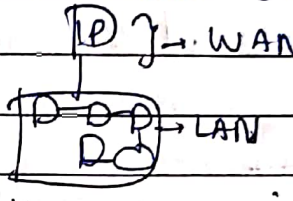
Slow because you are having multiple hops.

Date
Page

Bandwidths
• shared → Shared among many
• leased → only to 1 person

• ping google.com. our terminal starts sending seq packets to req site and make the stats.

• dig google.com.
• all internal connection CAN



• HUB vs Switch vs Router.
• HUB → broadcast to everyone that is their in it.
• Switch → Multicasts the message (list of IP's). Smart HUB.
• Routing → similar to switch but connects to IP's across networks.
• Gateway → local network to outside
Say N, of Google Realm wants to talk to Yahoo, it has to pass from Gateway.

Routing Algo

① Fixed Routing → centralised / Distributed
② Adaptive Routing
③ Flooding (Doesn't care)

① Fixed
• Done dijkstra at the beginning and it never changes.
So if one nodes dies then no path exists.

distributed Fixed Routing
↳ each Router stores it's next Router to go to.

② Adaptive
depends on state of the network
Recompute the tables.

③ Flooding → R1 sends to all it's neighbours and then so on.
to avoid cycles → hop count
hop → is each Router has hop count at each neighbour and when it reaches 0 → packet dropped.
always gets you shortest but more stress on Routers.

127 → localhost or loopback.

IP address ┌ → Network ID
└ → Subnet Mask.

IP: (192.168.32.128) /(0-32)
↳ denotes set bits of IP

Subnet Mask:
24 → 255.255.255.0 → CIDR
8 8 8

Network IP = (IP & Subnet Mask)

• Classes and Classful IP
↓ size are decided by production
↳ fixed class hosts.

• inside a network (LAN) internal nodes can talk to each other either by IP Internal or MAC.

Subnets → 2 # subnet bits.
Valid Host → $2^{\#\text{host bits}} - 2$ valid
000 all 0 network
255 all set broadcast

150.150.0.0 /30 → 30 set bits

Find the class → A, B, C.
then remaining 1's will be subnet bits and 0's would be Host.

places to change

IPaddresses

IPv4
0-255. 0-255. 0-255. 0-255
8bits after each.
32 bits
total 2^{32} bits
Run out of IPV4 addrs

IPV6.
xxxx:xxxx:
6 8 times
each is hexa
16×8 = 128 bit
2^{128} → IP address
Slowly moving to IPv6. First no. of IP.
Class A : 1-126
Class B : 128-191
Class C : 192:223
Class D : 224:239.
↳ First 8 bits → depends on 8 bits

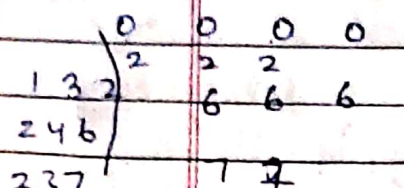
Codeforces

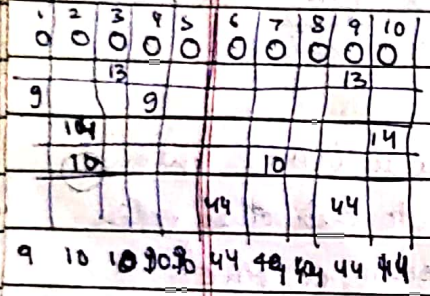
Q- Roads not only in Borland

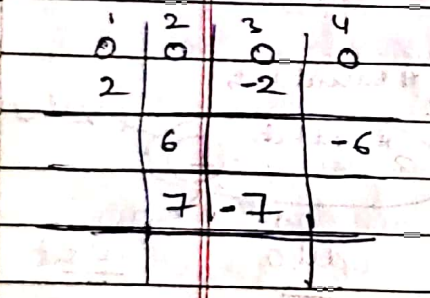
Spoj- Colorful Array

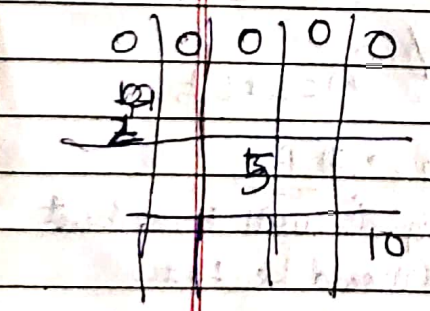
Q- Range queries to color the cells.

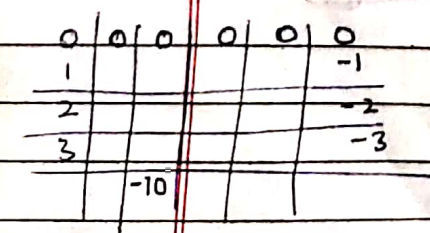
Find the color of each cell after end of queries.











Operating System - I

App

OS

Devices

Devices to talk to apps needs an OS in b/w (Middle level comp.)

code → compile → object code → Process (Main Mem) → Graphics card → Monitor

- Abstracts the hardware no need for us to configure the device.
- It does Resource Mgmt.

Processor

Power to run and execute instructions. compared by clock speed.

5 GHz → 5×10^9 instructions per sec.

RAM >>> HD.

capacitor & Transistor — Magnetic Disks

cache >>> 100 kB - 8MB.

- SRAM → Static → Transistor (used with cache)

DRAM → Dynamic → capacitor

used in RAM — leak current and are slow

- Multi-Programming:

if a process waits for I/O processor does a context switch to a new process.

How does processor know?

Status changes from I/p to Ready.

• Multi-Processing

Same scenario as before but now a new processor.

So, Multiprogramming + Process (Multi)

• Multitasking

You are processing 1 process at a single unit of time but when you look it over a span of time You'll find a bunch of process giving a view of simultaneous execution of processes → This is Multitasking (Time Slicing takes place)

• all these require Job Scheduling

(A) Print A-Z (B) Print 1-100

So in Multitasking → Since Time is sliced O/p → A B C 1 2 3 D E F G H . . 99

• Process is always run in RAM

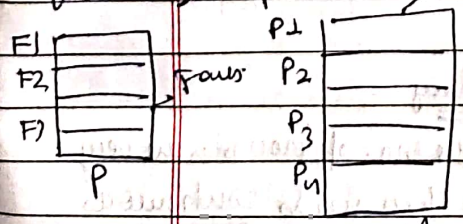
• Programs reside in HD.

#Fragmentation

You have available space but not contiguous amount of space → Ram is split into fragments of free memory.

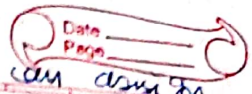
How to solve?

Divide the process into equal sized frames. → Page



Frame Size = Page size. RAM

So now we can assign frames non-contiguously.



to keep track of frames and page we have.

Processor Page Table

# Frame	# Page

→ For Process A

each Process will have it's own Process table

All this task is done by Kernel (OS in Admin Mode) (creating Frames assigning Pages etc)

Windows is susceptible to viruses because it's a private OS and has to wait for their own dev to fix and release Security updates.

However Linux is Open source. So whenever a virus comes the dev automatically patches them. So in general virus are created for windows so that they would pay them to get the virus removed.

• no need to load the all the frames at once. You can load some subset into RAM.

#Virtual Mem & Demand Paging

When is demand paging used →

• Either your process is too big to fit inside the RAM

Or

• There are only few pages that can accomodate some of the frames. ↔ Leads to demand Pg