Amoeba

After it being born, it takes 1hr to gain mature

After being mature, an amoeba produces 1 new amoeba for every 1hr.

How many amoebas will there be after 5hrs?

Sol) $t=0$  // born here & it is immature

$t=1$  I is mature

$t=2$ 

Let F_n be # amoebas after n seconds

$F_1 = 1 \quad F_2 = 2$

at $t=2$, 1 mature — 1 immature

$t=3$ 1 mature, 1 immature — 1 mature

$t=4$ 1 mature, 1 immature, 1 mature, 1 mature, 1 immature

$t=5$ 1 mature, 1 Im, 1M, 1m, 1IM, 1m, 1IM, 1M

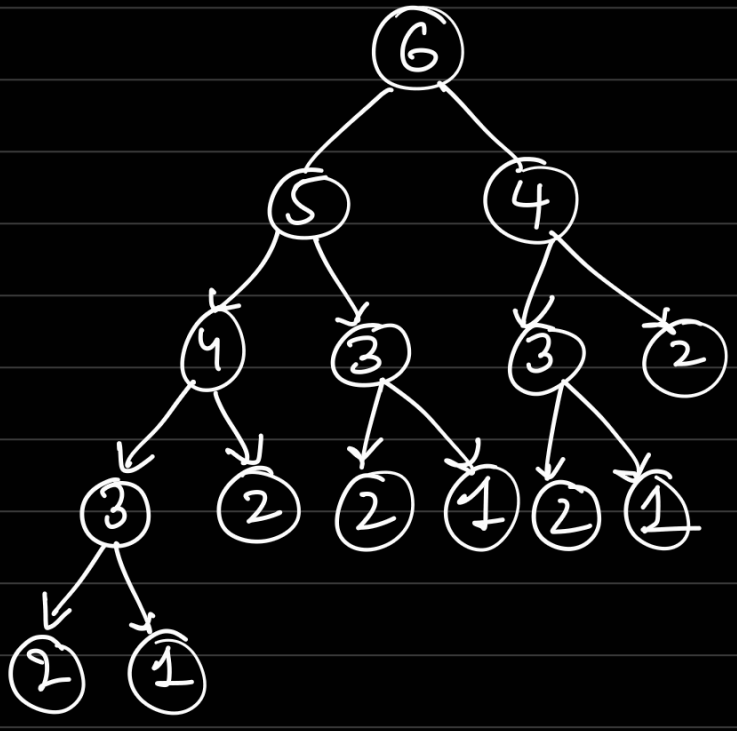
$F_3 = 3 \quad F_4 = 5 \quad F_5 = 8$

amoebas at time $t=n$ = # mature + # immature

mature $\rightarrow F(n-1)$; # immature $\rightarrow F(n-2)$

Thr. before to see how many are there.

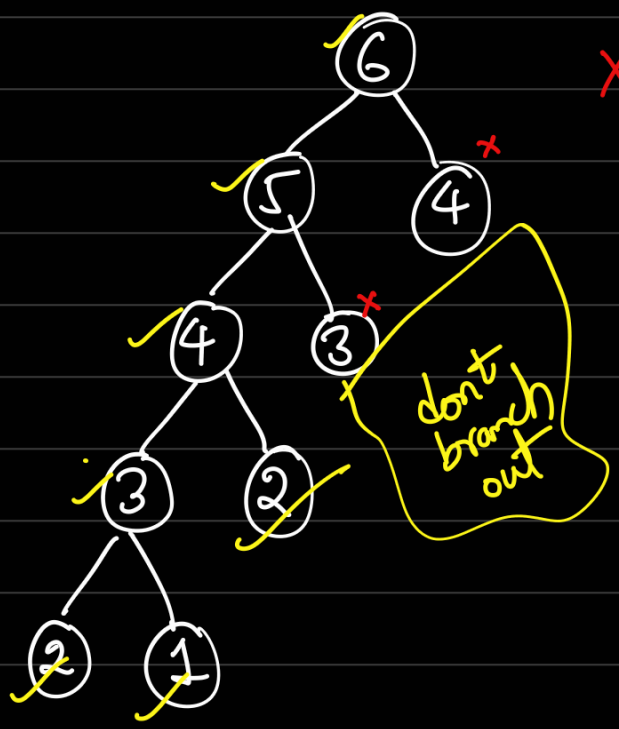
```
RecFibo(n)        // its recursive funct. to compute
                     nth Fibonacci number.
{
  if (n==1) return 1;
else if (n==2) return 2;
else return RecFibo(n-1) + RecFibo(n-2)
}
```



we compute F(3), 3 times.
F(4), 2 times

```
MEMFIBO(n)
{
    if (n==1)  F(1)=1;
    if (n==2)  F(2)=2;
    if (F(n) is undefined)
    {
        F(n) ← MEMFIBO(n-1) + MEMFIBO(n-2)
    }
    return F(n);
}
```

```
MEMFIBO(n)
{
    if (n==1)  F(1)=1;
else if (n==2)  F(2)=2;
else
{
    if (F(n) exists)
        return F(n);
    else return F(n-1)+F(n-2)
}
return F(n)
```



X → not computed.

Initialise array with -1.

```
ITERFibo(n)
{
    if (n==1) return 1;
    if (n==2) return 2;
    for(i=3 to n)
    {
        F[i] ← F[i-1]+F[i-2]
    }
}
```

→ storing array

How its better than prev. prog. in terms of T.C

Functn is called just once no matter whatever n

Prev Program #Functn calls depends on n.

This prog. has same T.C as prev. prog

Better approach than MemFibo(n) is Iterative Fibo

```
DYNFibo(n)
{
    if (n==1) return 1;
    prev ← 1 , curr ← 2 ;

    for( i=3 ; i≤n; i++)
    {
        next = curr + prev
        prev = curr
        curr = next
    }
    return curr;
}
```

n	1	2	3	4
F(n)	1	2	3	5

prev curr next

prev curr next

Given n as input, output 2^n in binary
↳ 1 (n times zeroes)

Algo:
```
output 1
for(i=1; i≤n; i++)
  output 0
```

For $n=(8)_{10} = (1000)_2$ O/P: 1000

Given n as input, output 2^n in decimal.

```
c=1
for (i=1 ; i≤n ; i++)
{
    c= c*2 ;
}
```

#multiplication = n.

I/p : 873 o/p : 2^{873}

Program doing: $1 * 2 * 2 * 2 * 2 \cdots * 2$ (873 multiplications.)

Can u do better?

$$2^n = \underbrace{2 \times 2 \times 2 \times \cdots \times 2}_{n \text{ times}}$$

$$2^n = 2^{\lfloor n/2 \rfloor} \cdot 2^{\lceil n/2 \rceil}$$

n is in 2^k powers & k is even

For n (even) it takes $\log_2 n = k$ mult & multiplications.

C++ implementation:

```cpp
long long power(long long a, long long n) {
    if (n == 0) return 1;
    long long half = power(a, n / 2);
    if (n % 2 == 0)
        return half * half;
    else
        return half * half * a;
}
```

TC

$T(n) = T(n/2) + O(1)$

$n^{\log_2 1}$ vs $O(1)$

1 vs 1

$T(n) = \Theta(1 * \log n)$

$\boxed{T(n) = \Theta(\log n)}$

#multiplication in above program $= \Theta(\log_2 n)$

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

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

n : F(n)=factorisation

0 : 0

1 : 1

2 : 1

3 : 2

4 : 3

5 : 5

6 : $8 = 2^3$

7 : 13

8 : $21 = 3 \times 7$

9 : $34 = 2 \times 17$

10 : $55 = 5 \times 11$

11 : 89

12 : $144 = 2^4 \times 3^2$

13 : 233

14 : $377 = 13 \times 29$

15 : $610 = 2 \times 5 \times 61$

16 : $987 = 3 \times 7 \times 47$

17 : 1597

18 : $2584 = 2^3 \times 17 \times 19$

19 : $4181 = 37 \times 113$

20 : $6765 = 3 \times 5 \times 11 \times 41$

$[F(n)]^2$

0

1

1

+ 4

9

25

+ 64

+ 169

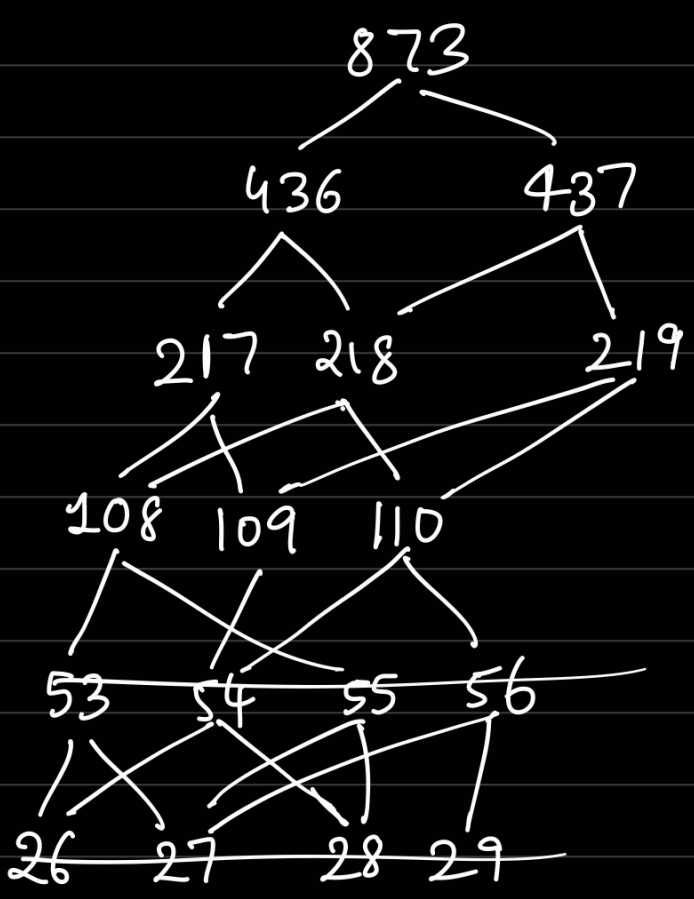
441

1156

$$F(n) = \left[F\left(\lfloor n/2 \rfloor\right)\right]^2 + \left[F\left(\lceil \frac{n}{2} \rceil\right)\right]^2 \quad \text{if } n \text{ is odd}$$

$$= \left[F\left(\frac{n}{2}-1\right)\right]^2 - \left(F\left[\left(\frac{n}{2}+1\right)\right]\right)^2 \quad \text{if } n \text{ is } \underline{\text{even}}$$

for $F(0)=0,\ F(1)=1,\ F(2)=1$

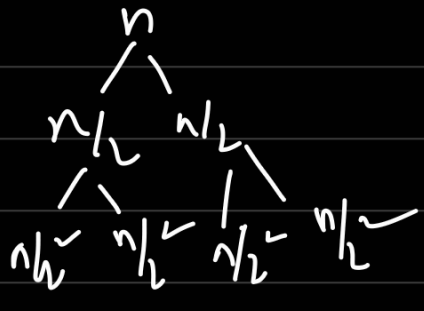


$$F_{2n-1} = F_{n-1}^2 + F_n^2$$

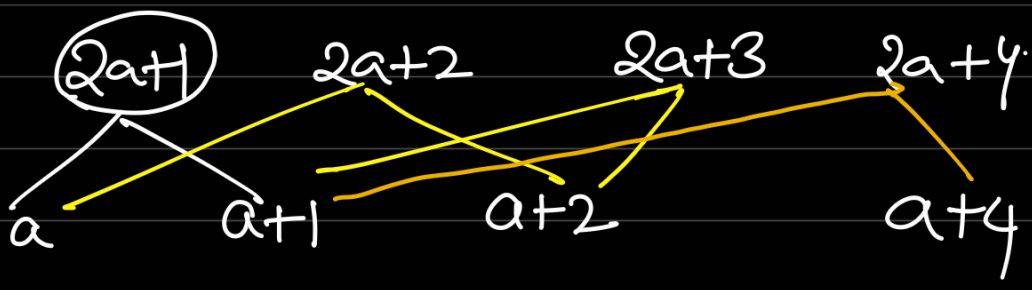
$$F_{2n} = F_n\left(F_{n-1} + F_{n+1}\right)$$

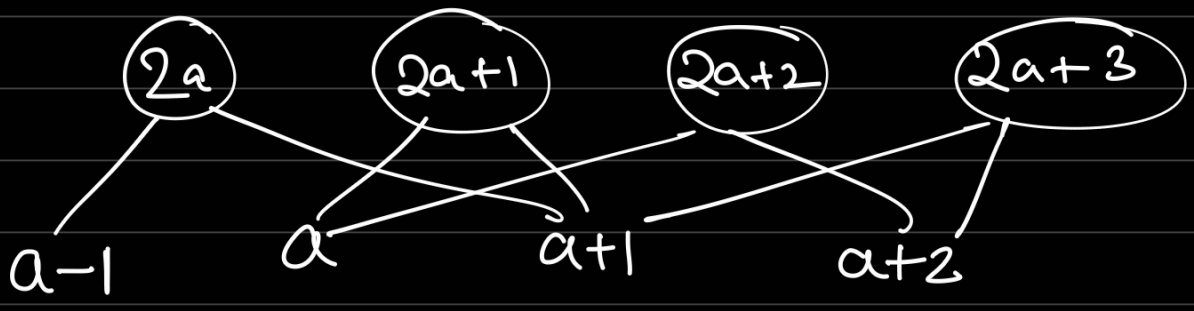
(WKT: $F_{n+1} - F_{n-1} = F_n$)

$$= F_{n-1}^2 - F_{n+1}^2$$



$$\underbrace{2 * 2 * 2 * \cdots * 2}_{\log_2 n} = 2^{\log_2 n} = n$$





To compute nth fibo, we require $4 * \log_2 n$ operation

width height

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$F(t) = 2F(t-1) - F(t-4)$$

$$A \times \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 7 \end{bmatrix} \qquad A \times \begin{bmatrix} 2 \\ 4 \\ 7 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 13 \end{bmatrix} \qquad A \times \begin{bmatrix} 4 \\ 7 \\ 13 \end{bmatrix} = \begin{bmatrix} 7 \\ 13 \\ 24 \end{bmatrix}$$

$$A^2 \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 13 \end{bmatrix}$$

$$A^t \begin{bmatrix} F(0) \\ F(1) \\ F(2) \end{bmatrix} = \begin{bmatrix} F(t) \\ F(t+1) \\ F(t+2) \end{bmatrix}$$

IMP. A^t can be computed in $\log_2 t$ operation

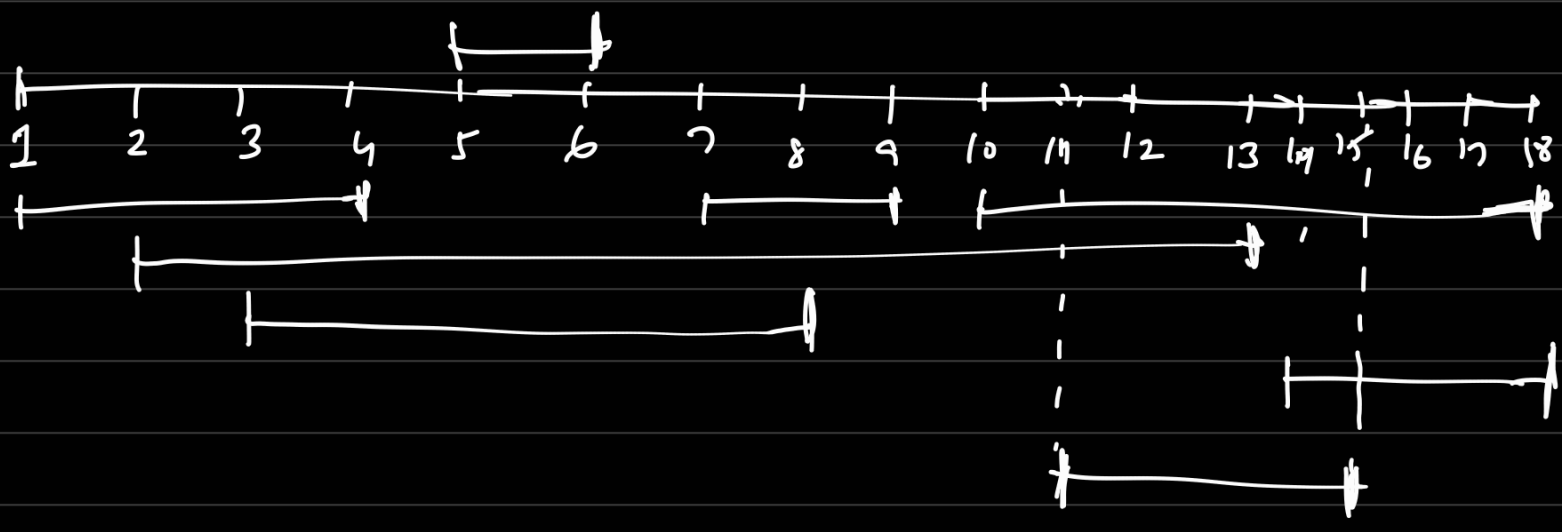
A^t has $2\log_2 t$ matrix multiplications.

F(t) found in $O(\log_2 t)$, to be precise, ($2\log_2 t$ matrix multipl)

↳ $2 \times 27 \log_2 t$ multipl

Matrix (3×3)

↳ 9 entries × 3 (normal mult) = 27



Interval Scheduling

- The time period for each class is given by its instructor.
- What is the maximum number of classes that can be scheduled in one classroom in such a way that there are no clashes?

Ans) 4

Instructor	Start time of class	End time of class	Duration of class
Prof. Anu	1	4	3
Prof. Ben	7	9	2
Prof. Cam	10	18	8
Prof. Dan	2	13	11
Prof. Eva	3	8	5
Prof. Fez	14	19	5
Prof. Gus	5	6	1
Prof. Hal	11	15	4

L3 7 Aug

[1,4]	[1,4]	[1,4]
[5,6]	[5,6]	[5,6]
[7,9]	[7,9]	[7,9]
[11,15]	[14,19]	[10,18]

Correctness, Optimality

2 things to worry about

Crctness:- Is set of intervals non-overlapping?

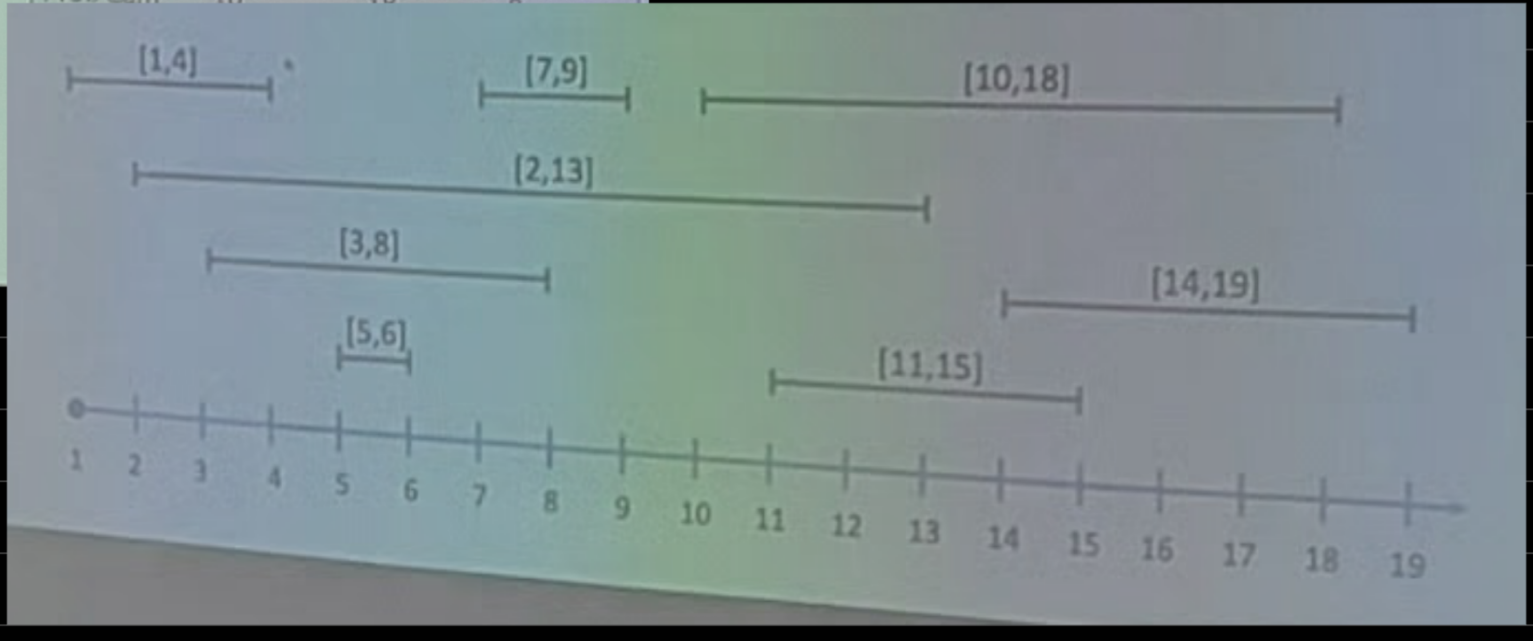
optimality:- Is the size of the set the maximum possible?

Any Algo. problem, do a visulation.

Instructor	Start time of class	End time of class	Duration of class
Prof. Anu	1	4	3
Prof. Ben	7	9	2
Prof. Cam	10		

min val = 1

max val = 19



Algo1 :- Earliest-Start-time

From left to right, select the interval with the earliest start time.
delete intervals which overlap with selected interval

Q) Why this algo. give optimality ?
↳ size of the set is max. possible.

↳ Ans) No.

Counter Eg 1 :-

Q)

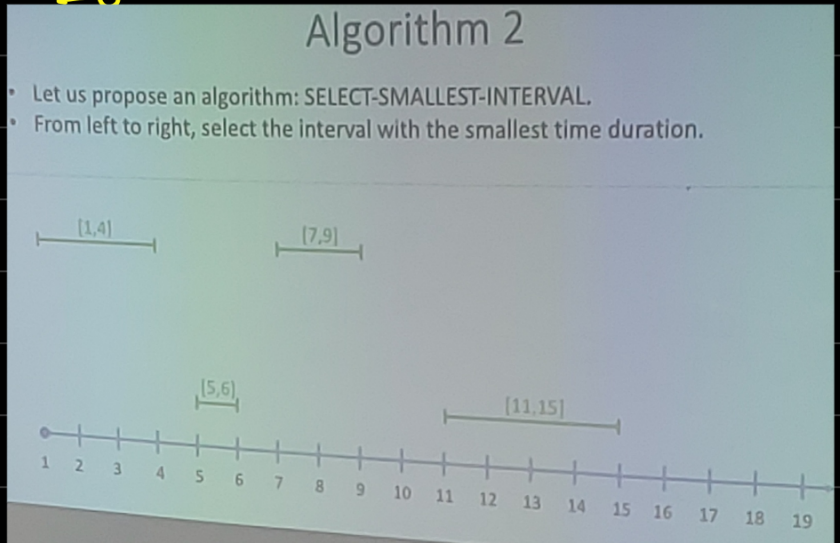
Algo1 :- it select 1 & doesn't select 2, 3.

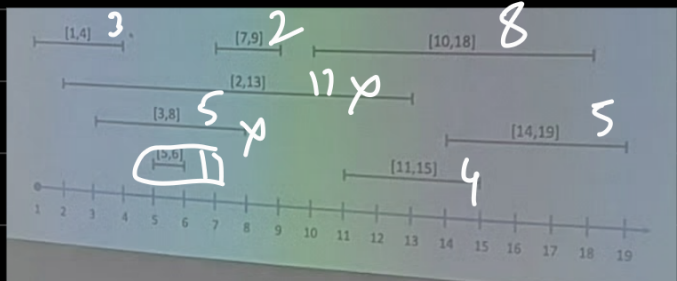
Algo 1 given any 

But optimal sol. (by Common sense) is 

Q) Does algo 1 gives interval which don't clash each other?
A) Yes.

Algo2 :- SELECT-SMALLEST-INTERVAL

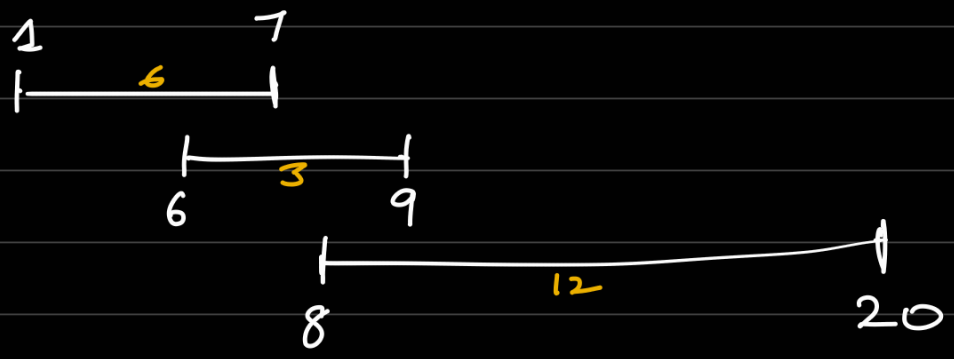




Correctness :- Yes. (Bec. as algo. follows, no overlapping intervals)

Optimality :- No.

Counter eg :-



Algo 2 : o/p

optimal :-

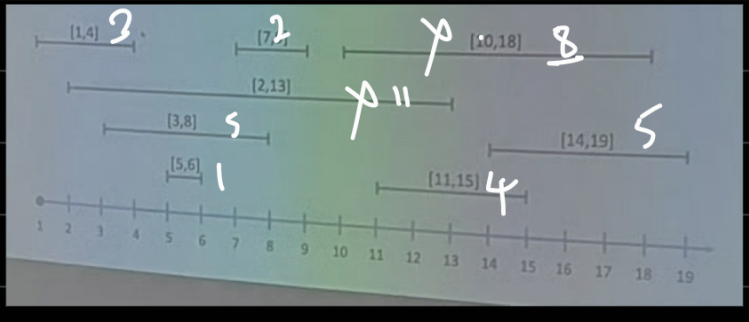
Algo 3 :- DELETE-LARGEST-INTERVAL

↳ from L to R, delete interval with largest time duration

Correctness :- Yes.

optimal :- No.

take Same above counter Eg.



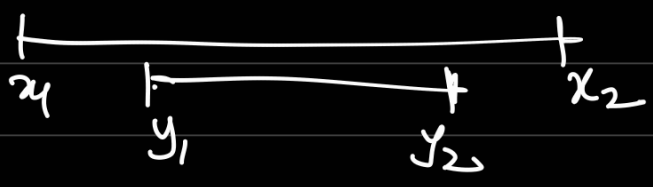
Algo 3 : o/p

optimal :-

Dominating Intervals

An interval $[x_1, x_2]$ is said to dominate an interval $[y_1, y_2]$

if $x_1 \leq y_1$ & $x_2 \geq y_2$



Algo:

From L to R, delete all intervals that dominate another interval

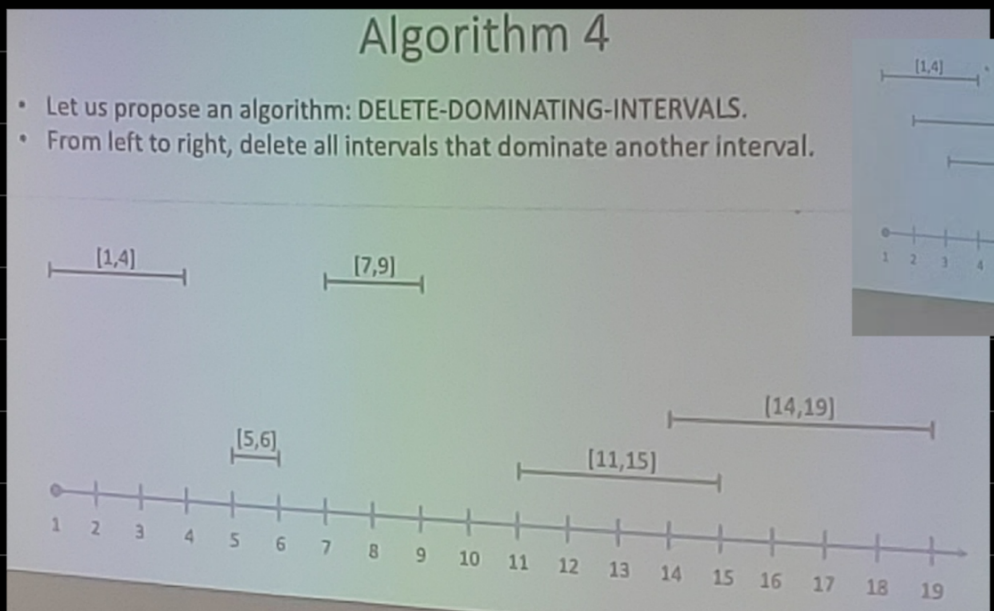
Correctness: No.

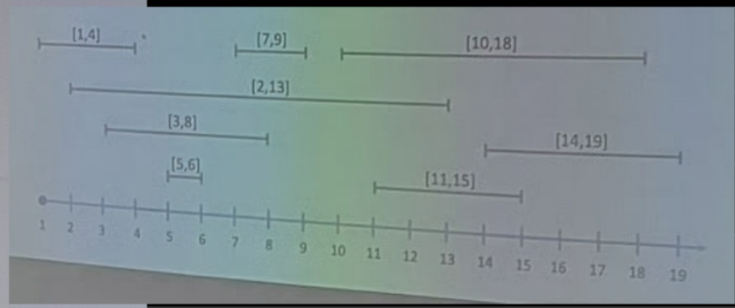
if correctness is no then we dont talk about optimality

(9)

[3,10] is dominating 4,6

so delete [3,10] & take [4,6]





Given:

After Alg. 4, we remove all dominating intervals.

Suppose 2 intervals $[x_1, x_2]$ & $[y_1, y_2]$ st. $x_1 \le y_1$

then $x_2 \le y_2$

Proof:

Not possible (possible)

bec. $[x_1, x_2]$ is dominating the $[y_1, y_2]$

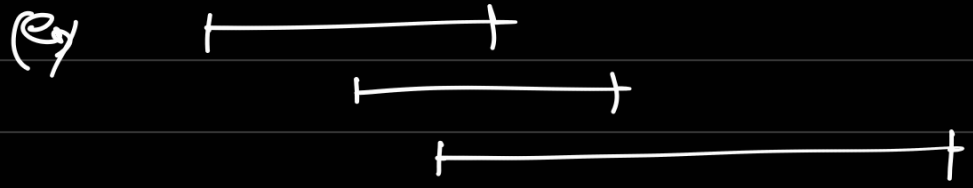
Observation :-

After dominating delete then sort by start time is same as sort by end time

---

ALG 4 then ALG 2 : Not works.

ALG 4 then ALG 3 : Not works.



Algorithm 5 : ALG4 + ALG1

First delete dominating intervals
Then, Earliest start time.

ALG4 then ALG1 :— It works. Why? Let see proof

Let OPT be optimal (max) # of non-overlapping interval

ALG4 ≥ OPT → why?

ALG5 ≤ OPT — (1)

↳ bec ALG5 is cret Algo. (it has Alg 1)

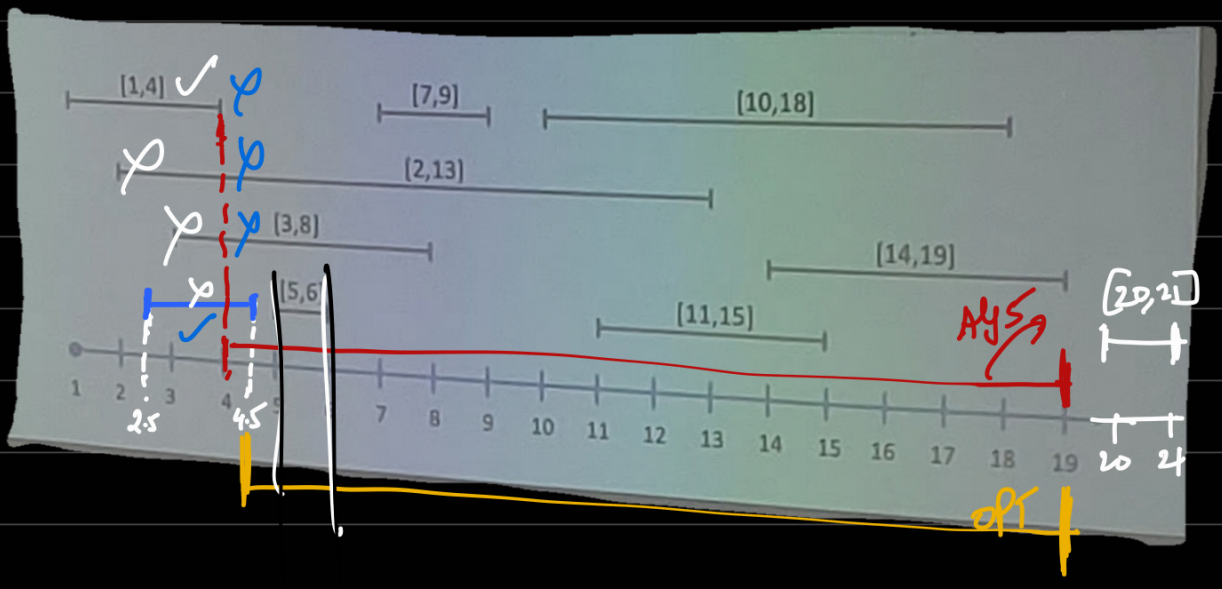
then
ALG1 ≤ OPT
ALG2 ≤ OPT
ALG3 ≤ OPT

These are numbers only.

Whenever ALG5 selects its ith first interval, it has all the non-dominated intervals available to itself that were available to OPT after OPT selected its first interval

↳ we can say ALG5 ≥ OPT — (2)

ALG5 performs atleast as well as OPT



ALG S	OPT
[1, 4]	[2.5, 4.5]
[5, 6]	[5, 6]
[7, 9]	[7, 9]
[10, 18]	[14, 19]

we need to prove that our algo same # of opt.
but not the same set as of opt

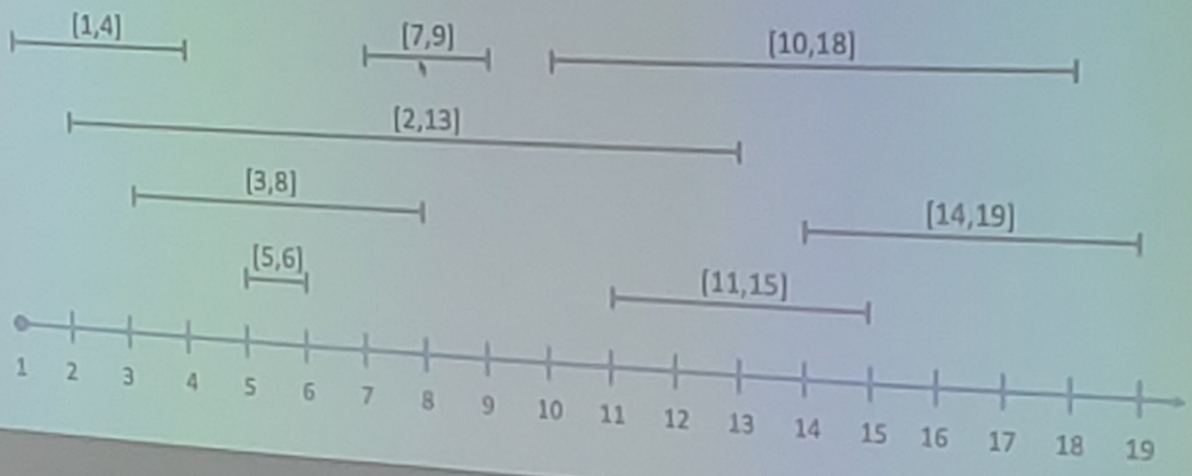
from ①, ②, we say ALGS = OPT

Student's idea:-
↳ some other proof.

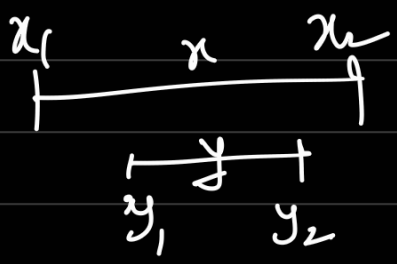
Algorithm 6 [Best Algo in Practice]

- Let us propose an algorithm: EARLIEST-END-TIME.
- From left to right, select the interval with the earliest end time.

Correct ✓
optimal ✓



(g1)

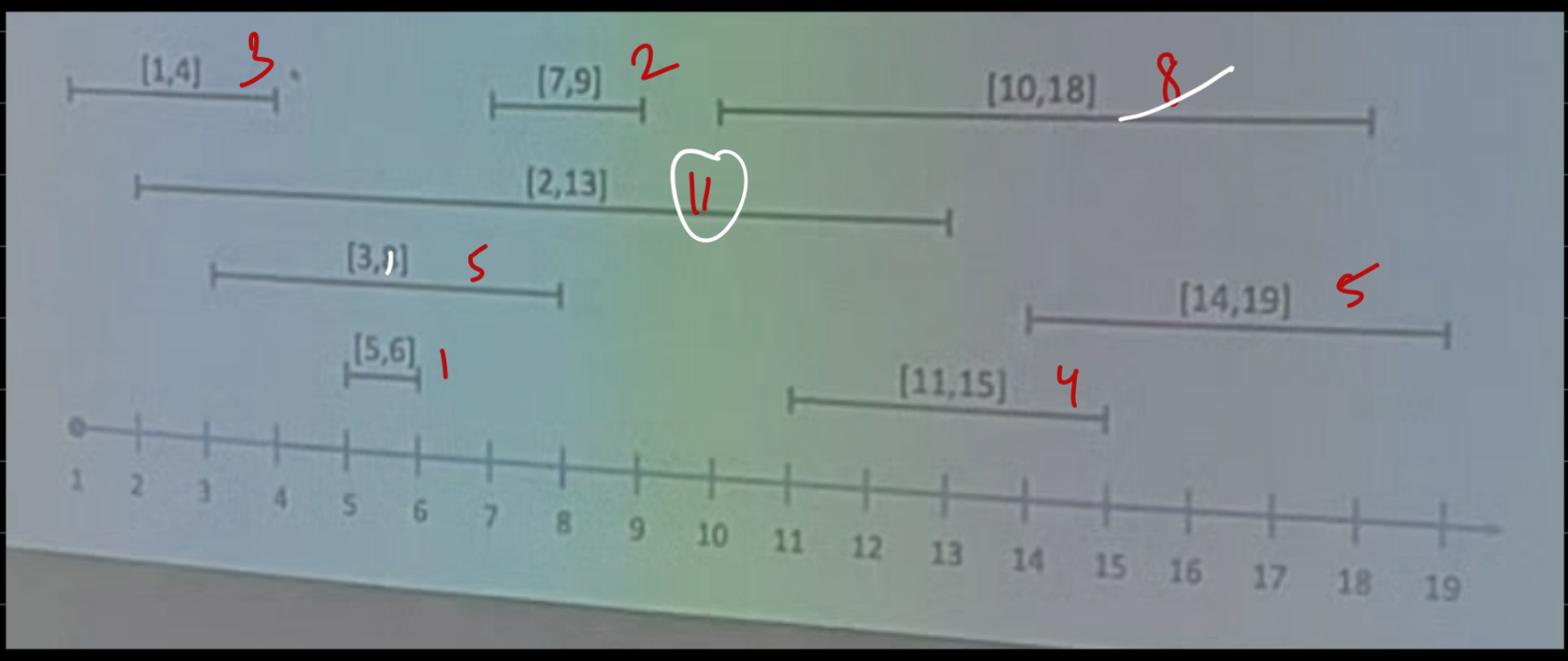


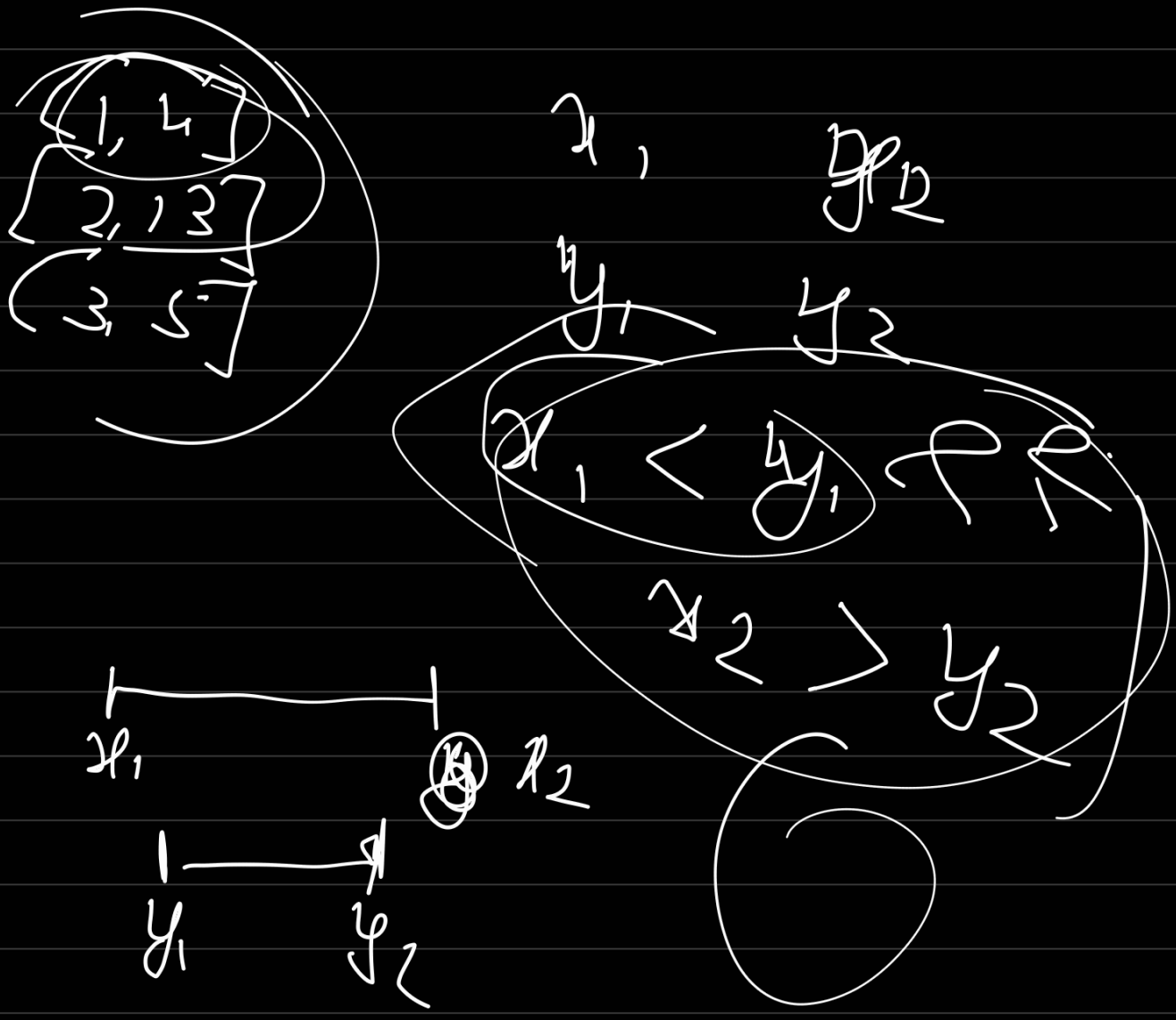
Alg 6; select y & not x.

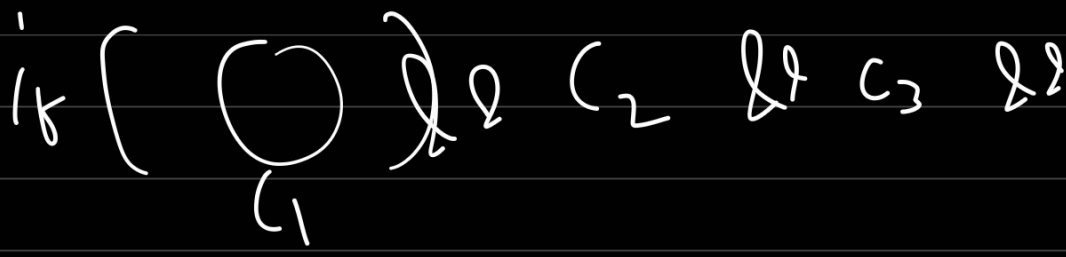
↳ it dont take dominating intervals · implicitly

↳ why means ? A) look above eg 1,

6


[1,4] 3
[7,9] 2
[10,18] 8
[2,13] 11
[3,9] 5
[14,19] 5
[5,6] 1
[11,15] 4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

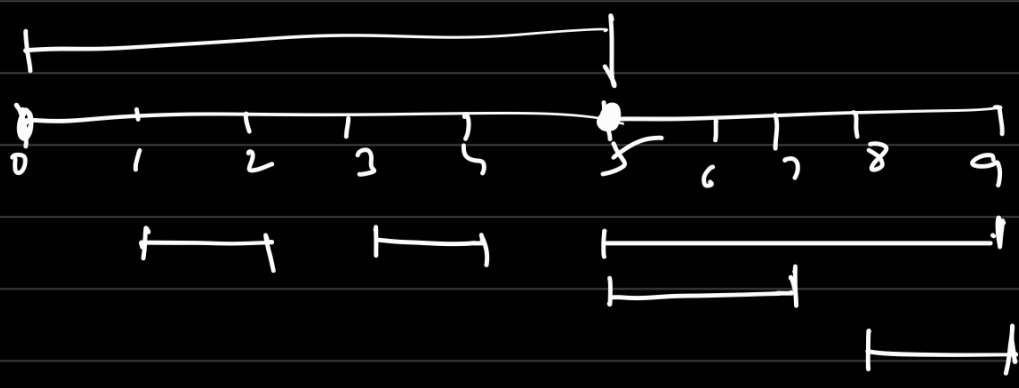

1, 4
2, 13
3, 5
x1, x2
y1, y2
x1 < y1
x2 > y2
x1
x2
y1
y2


if ( ) && C_2 && C_3 &&
C_1


x_1
x_2
y_1
y_2
y_1
y_2
y_1
y_2
x_1
y_2

```
sort( i.begin(), i.end(), [&](auto const& a,
                              auto const& b)
{ return a[1] < b[1]; });
```

```
sort(i.begin(), i.end(), [&](auto const& a,
                             auto const& b)
{ return a[1] < b[1]; })
```



$$\begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} f_{n+2} & f_{n+1} \\ f_{n+1} & f_n \end{bmatrix}$$

$$\begin{bmatrix} f_4 & f_3 \\ f_3 & f_2 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix}$$

n=1 n=2 n=3 n=4 n=5

0 1 1 2 3

f_0 f_1 f_2 f_3 f_n

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & n \text{ is even} \\ a^{n/2} \cdot a^{n/2} \cdot a & n \text{ is odd} \end{cases}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$

```
fun(double x, int n)
{
    if (n==0) return 1;
    if (n==-1) return 1/n;
    double temp = fun(x, n/2);
```

$$\begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{bmatrix} f_n & \\ & \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

L4 11 Aug

Divide & Conquer

$T(n) = 2T(n/2) + O(n)$

$n^{\log_2 2} = n^1$ vs n

$TC = \Theta(n \log n)$

$T(n) = 2T(n/2) + c \cdot n$

$= 2^2 T(n/2^2) + c \cdot n + c \cdot n$

$= 2^3 T(n/2^3) + c \cdot n + c \cdot n + c \cdot n$

$\vdots$

$= 2^k T(n/2^k) + (cn)(k)$

$(n/2^k) = 1 \quad (k = \log_2 n)$

$= n \cdot T(1) + (cn)(\log_2 n)$

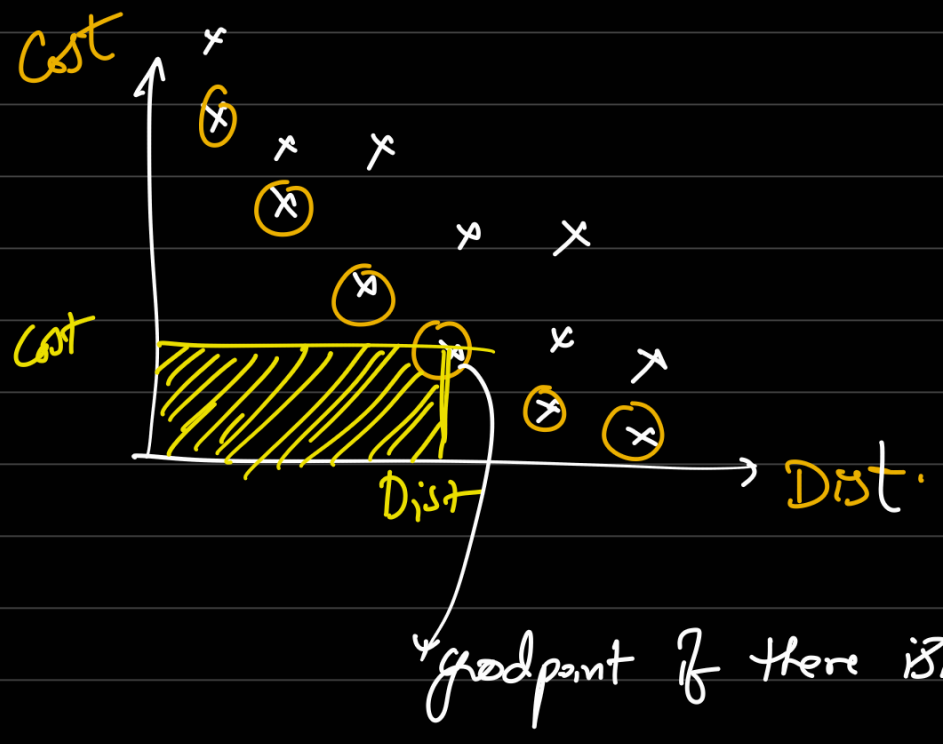
$= n \cdot O(1) + (cn \log_2 n)$

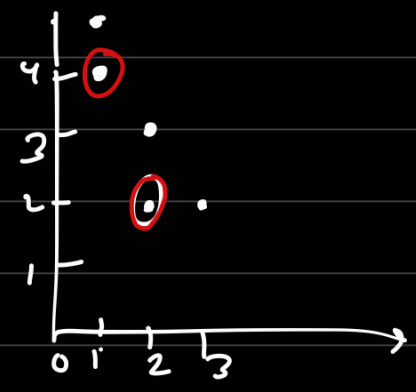
$= n \cdot k_1 + c \cdot n \log_2 n$

$= \underline{\Theta(n \log_2 n)}$

Hotels near sea shore

Given (x, y) x is cost of hotel
y is dist. of hotel from Seashore.





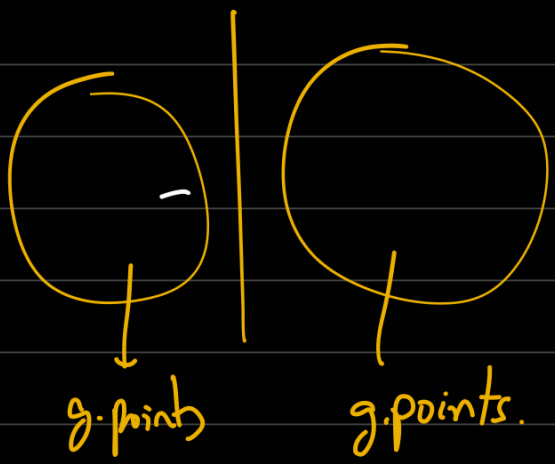
Goodpoint if there is no point in this region.

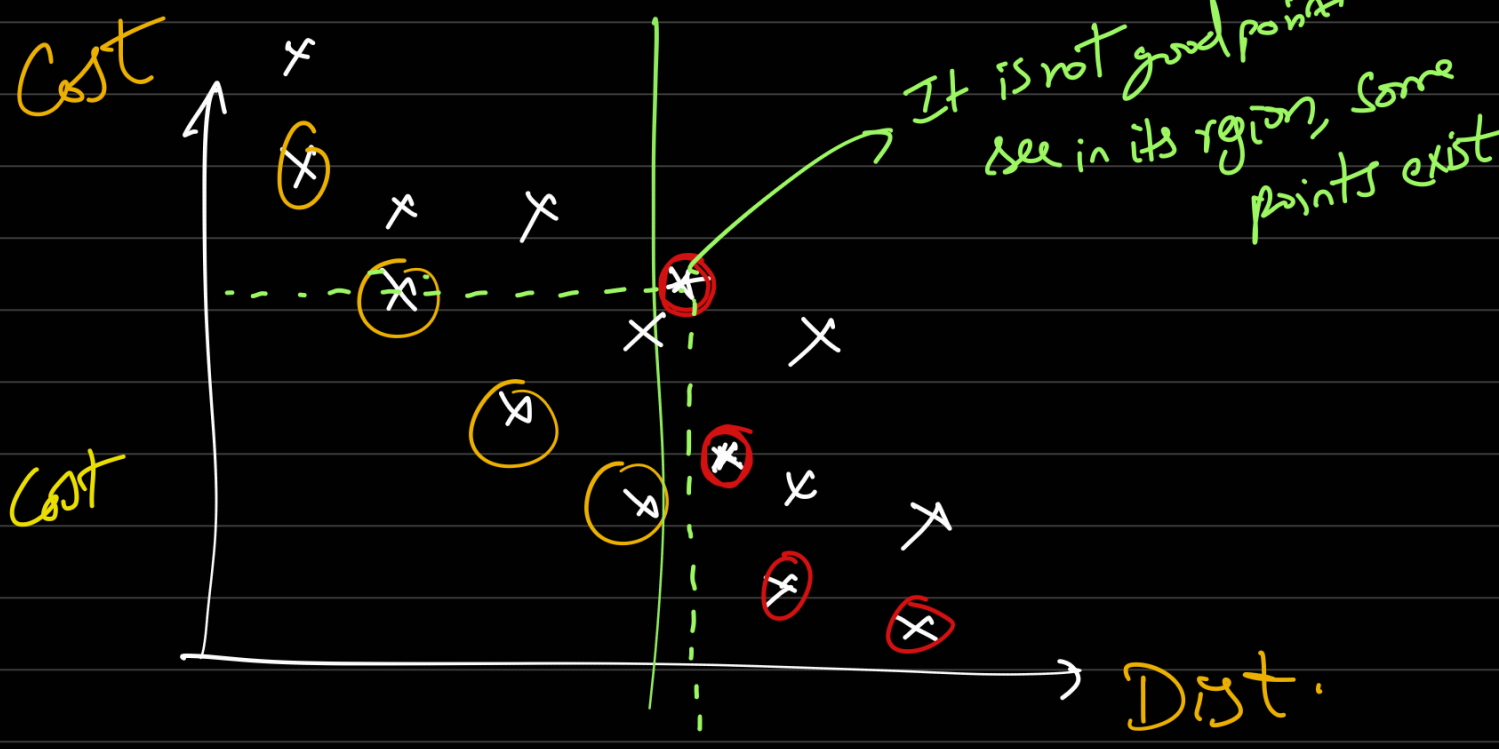
A pt $P(x_1, y_1)$ is a goodpoint if $\nexists$ a pt $Q(x_2, y_2)$
st. $x_2 \leq x_1$ & $y_2 \leq y_1$.

Problem :- Find good points in dataset

Sort with x coordinate & take half, half.

Prove that in comparision based algo., we can't design Sorting algo less than $n \log n$ T.C





Let say magically, I got ○ left good points

◎ right good points

If u see globally, ○ is good point.

All right goodpoints need not to be actual goodpoint

It can be badpoint

So do merge, take rightmost goodpoint in leftside

→ Take it as our y cordinate

Now scan rightside goodpoints;

remove if (y of Rightside goodpoint > our y cordinate)

O(n).

$T(n) = 2T(n/2) + O(n) \Rightarrow T(n) = n \log n$

total time ① sort → $n \log n$

② D&C → $n \log n$

overall TC = $n \log n$

Alternative sol :— TC $O(n \log n)$

① Let sy u did sorting

② Observation :- all good points will have dec. y cordinate

Peak finding

Given array. extreme ends are -1 (+ve integers are bounded by -1) & atleast +ve int. exists. & No. are distinct.

Where $P[i]$ is a peak if $P[i-1] < P[i] > P[i+1]$

H/w :- No. arent distinct & peak is $P[i-1] \leq P[i] \geqslant P[i+1]$

(eg) -1 2 7 1 8 10 15 3 6 -1

Observation :- Since no. are distinct, $\Rightarrow$ it has a peak.

Proof :- n +ve integers.

array size $= n+2$

Assume there is no peak in whole arr $\Rightarrow$ all ele. are not peak.

1st ele. is not peak. $\Rightarrow$ 2nd ele. $>$ 1st ele.
else 1st ele. is peak.

Now 2 ele. is not peak $\Rightarrow$ 3rd ele. $>$ 2nd ele.

⋮

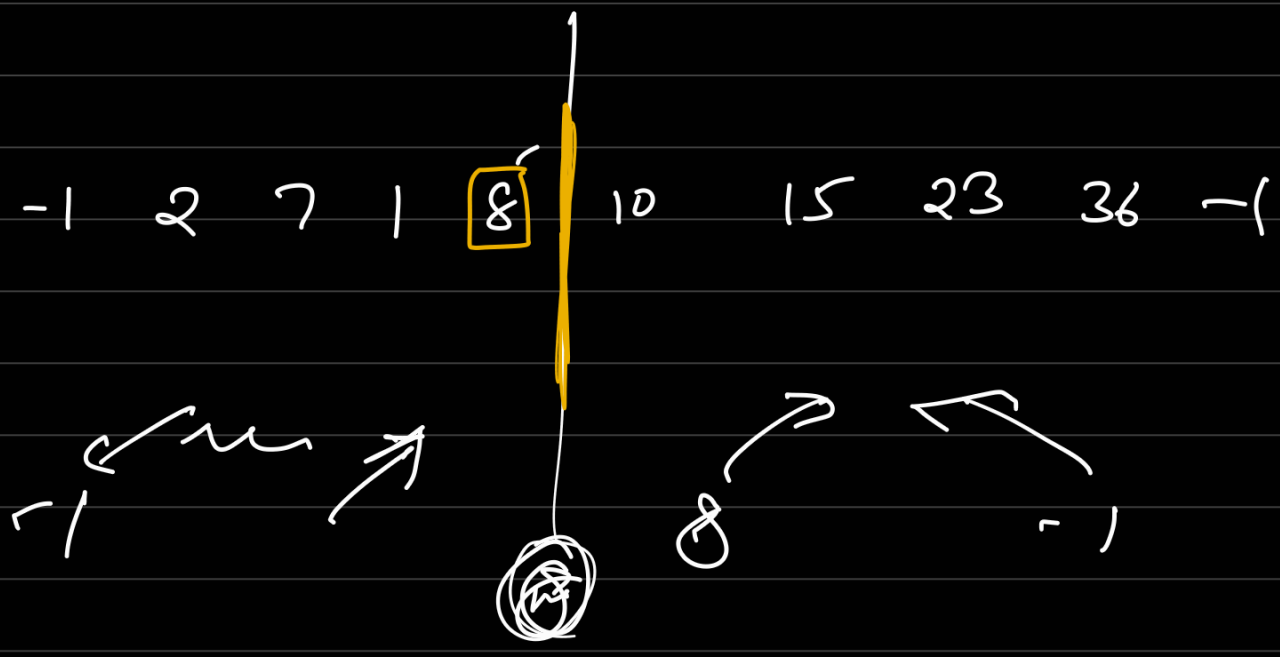
$(n-1)^{th}$ +ve int is not peak $\Rightarrow$ n^{th} +ve int ele. $> (n-1)^{th}$ +ve int.

n^{th} +ve int is not peak $\Rightarrow$ $(-1) >$ (n^{th} +ve int ele.) $\hookrightarrow$ ①
its beside ele. of n^{th} +ve int

But WKT: (+ve int > −1)

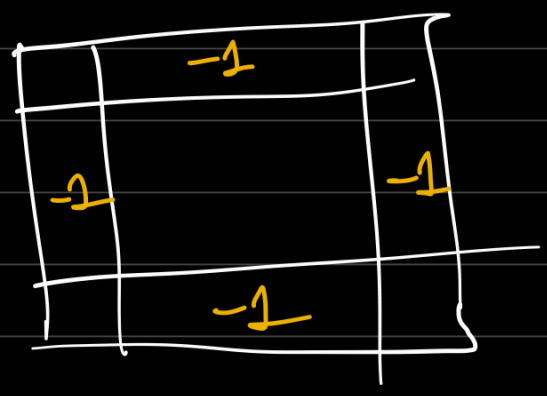
① contradicts fact

⇒ our assumption is wrong.



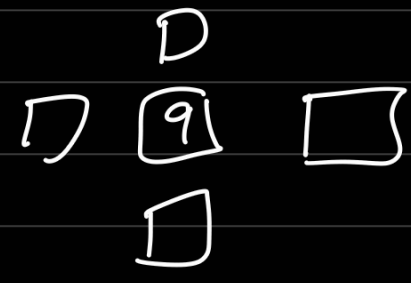
L5 14 Aug

2D Peak finding



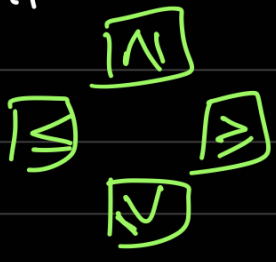
these are invisible -1 on 4 directions

I/P : n, $n \times n$ matrix given
O/P : report peak element



9 is peak if all 4 direc. ≤ 9
(Equal is also accepted)

WC TC :- $4n^2$ // n^2 elements present in it



n rows → find peak for each row.

And for n peaks, check 4 condition

→ This does not work.

bec. whatever u have peak in a row, that might not be in 2D array

Hussain: peak of row. When u get middle, do more.

→ analyse it.

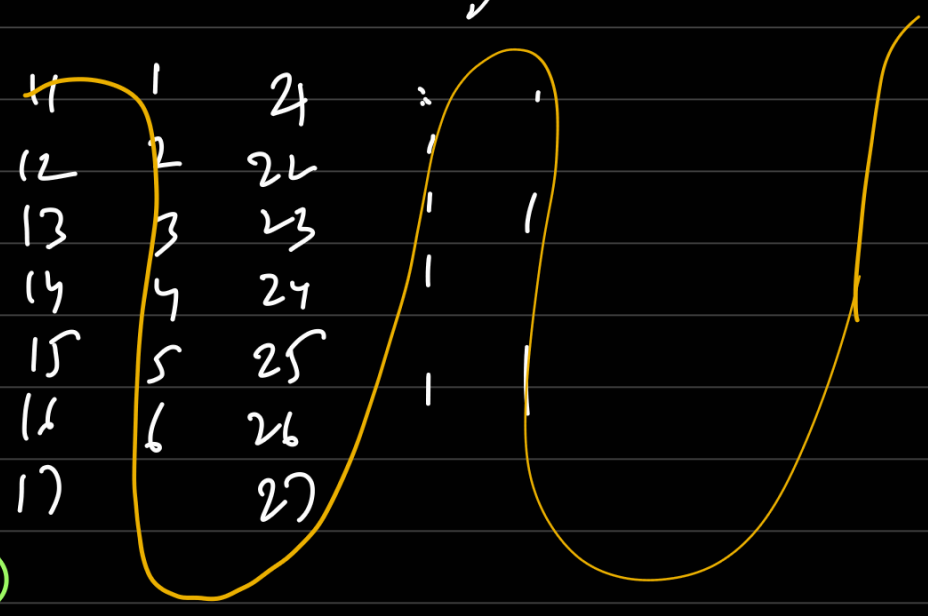
No

One person : (Step1) Take first ele. check if its 4 surro. are lesser

if yes then report

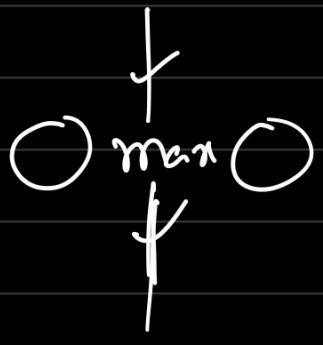
if no, take max. one & do step 1

→ Works but $(n^2/2)$



(Soustni Idea) Lini Mayn Sol.

Take middle col. & take max.



Check adj. ele., if max > adj then report it.

else go in direction of highest neighbour ele. (let say right).

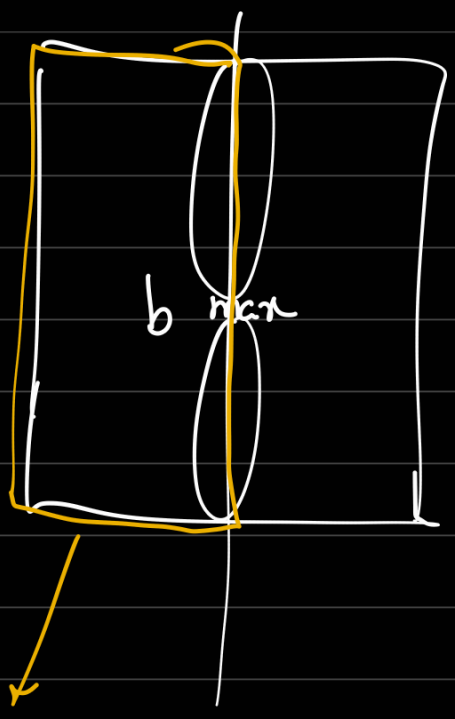
deft space is removed.

do same algo in right search space.

$(n \times \log n) = O(n \log n)$

find max in col.

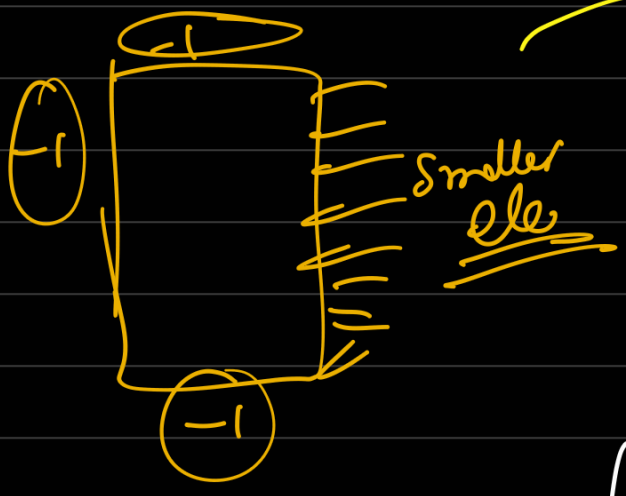
everytime half, half, ...

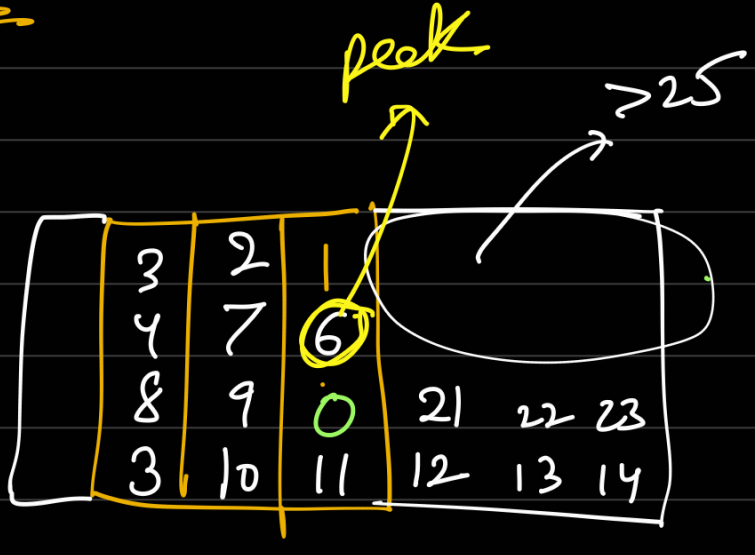


Let say $b > max$

$\Rightarrow b >$ all ele. in col.

$\rightarrow O(n \log n)$

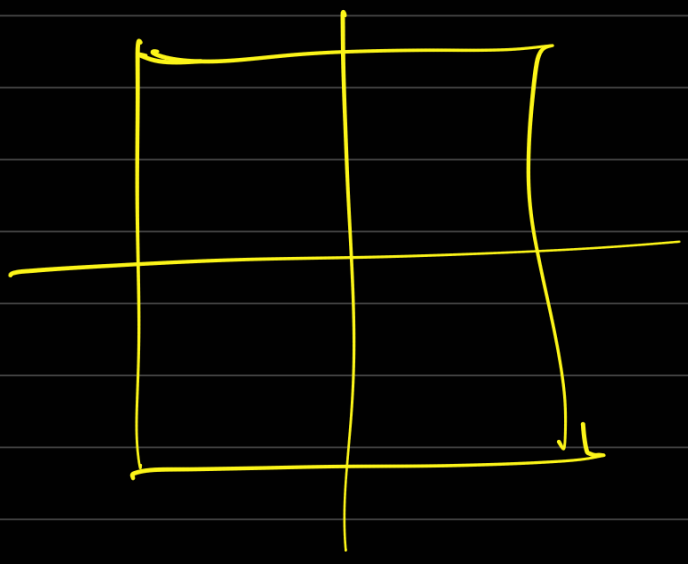




Leakage problem (if u take peak but not max)

Algo 2 :- exactly except peak ,

O(n) (Moving each quadrant)



Col. & row Bytmap → 2n

→ n

→ n/2

→ n/2^2

⋮

1

n [2 + 1 + 1/2 + 1/2... + 1]

n [(2)

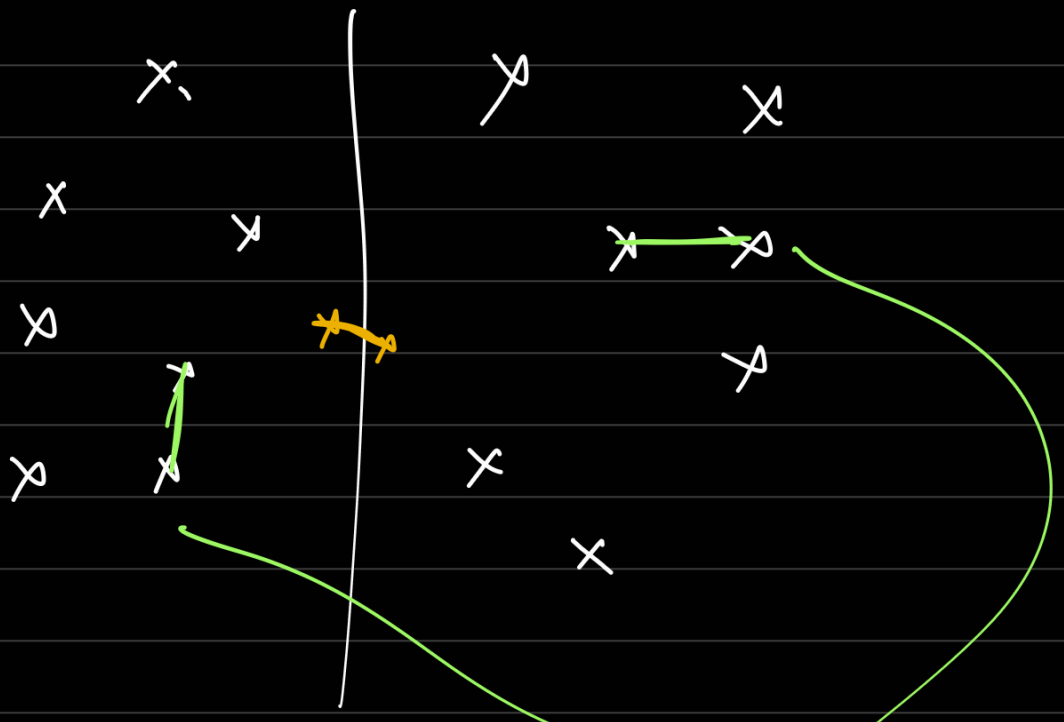
(What we guarantee that peak in that area?)

closest pair problem

Naive:-

1) Sort all points by x axis.

2) Take left side points & right side points in b/w x cordinate



use divide & conquer.

→ closet pair in LHS

→ RHS pair in LHS.

Compare among them & report smallest dist.

Q) Is it correct?

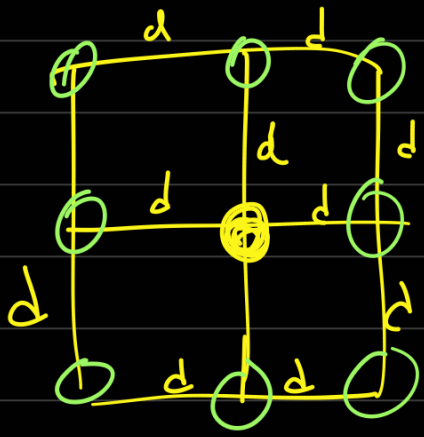
A) No

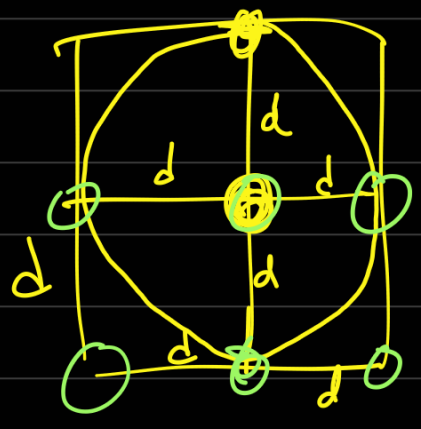
Bcc. There could be one pt LHS one pt RHS.

across pair must be $< \min(d_1, d_2)$ to be closest pair.

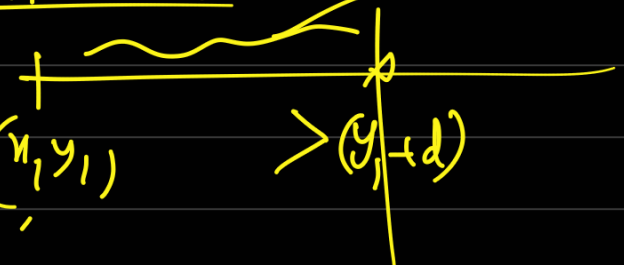
Let $d = \min(d_1, d_2)$

In $d \times d$, atmost 4 points

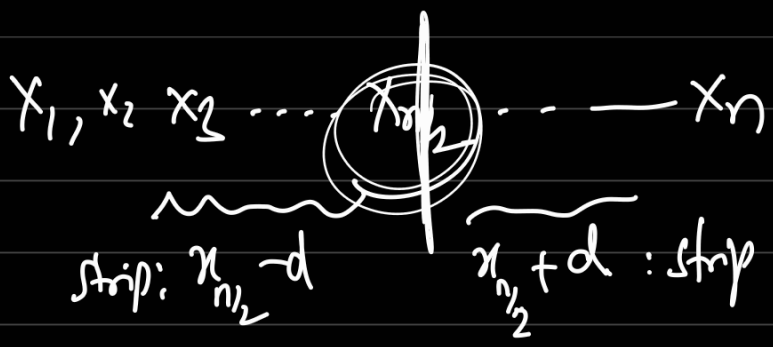




Sorted in order of y_i → atmost 5 points are there



$T(n) = 2T(n/2) + c \cdot n$



Keep points in both side strip.

↳ Now u take all points & sort by "y" coordinate

L6 18-Aug-25

Coin Denominations Problems

Currency Coins : 1, 2, 5, 10, 20, 50, 100.

Given an amount A, what is the min. num. of coins needed to totalup to exactly A?

↳ Greedy strategy works.

(P1)

A = 93	largest coin u can select
43	50
23	20
3	20
1	2
0	1
	-

Take always highest one which is just less than Total Sum.

Greedy Alg. → immediately gain (short term goal)
In long run, it might be harmful.
Can not harmful.

Greedy Alg.
- D&C
- D.P
- Recursion

} Make use of a previously computed smaller instance of the same problem to compute larger instance of the problem

Currency Coins : 1, ~~2~~, 5, 10, 20, 50, 100.

↳ Can greedy give optimal sol.? Ans) Yes.

Correct Solution →
Optimal Solution → producing with min no. of Coins

~~X~~ 2, 5, 10, 20, 50, 100.

~~51 = 50 + 1~~ Lec coin(1) not present.

A = 51 using greedy, it doesn't give correct sol.

51 = 20 + 20 + 5 + 2 + 2 + 2 (It is crct)

Q) Currency Coins : 8, 9.

Is A ≥ 50 always possible?

A) No.

50 = 8 + 8 + 8 + 8 + 9 + 9

51 = 8 + 8 + 8 + 9 + 9 + 9

55 ≠ 8x + 9y

Q) Is A ≥ 56 always possible?

A) Yes.

Prove urself

x, y numbers . gcd(x, y) = 1

max. impossible amount $A = xy - x - y$

If u have 1 rupee coin, then every amount is possible to be created.

Q) Suppose min. deno. is always 1. (1, 5, 11)

A) A = 15

by greedy

$15 = 11 + x$

$x = 4$ not possible coin.

by intuition

$15 = 5+5+5$

A = 15, by greedy, we can't have sol

Q) Suppose min. deno. is always 1.

1, 4, 6, 9, 14.

What is best strategy for any amount A with min # coins?

A) Q) Is Greedy give optimal sol? A) NO.

A = 26 Greedy = 14+9+1+1+1 > greedy fail

By optimal = 14+6+6 (intuition)

A = 31 Greedy = 14+14+1+1+1 > greedy fail

optimal = 9+9+9+4

→ Greedy Approach got failed, so go for D-P

I/P

Q) Design an Algo. which takes no. n
& $c_1 < c_2 < c_3 \cdots < c_n$
& Amount A

O/P: min # coins that addup to exactly A.

Assume lowest denomination coin $(c_1) = 1$

Eg, 1, 4, 6, 9, 14

Let $M[i]$ be min # of coins needed to addup to exactly i.

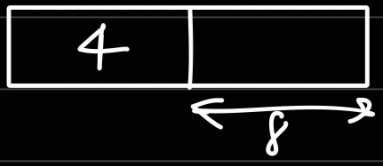
i	$M[i]$
0	0
1	1
2	2
3	3
4	1
5	2
6	1
7	2
8	2
9	1
10	2
11	3
12	2
13	2
14	1

Well, I will take coins from box till it will sum right? (Yes)

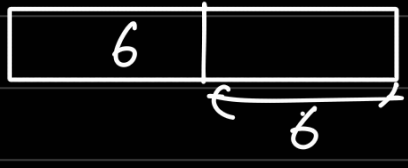
So last coin, u choose can be 1/4/6/9/14.

find m[12]

1+m[11]



1+m[8]



1+m[6]

1+m[3]

$$m[12] = \min\{\ 1+m[11],\ 1+m[8],\ 1+m[6],\ 1+m[3]\ \}$$

$$= \min\{4, 3, 2, 4\}$$

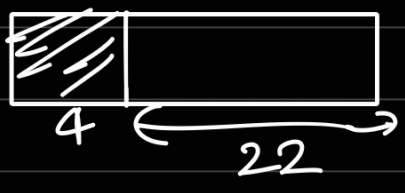
$$= 2$$

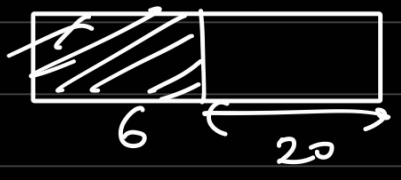
Lets find $M(26)$

last coin, u choose 1/4/6/9/14

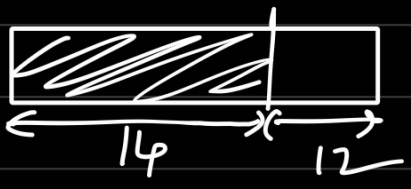
Sum = 26

$M(26) = \min\{$ 5 $1 + M[25]$
4 $1 + M[22]$ $\}$
3 $1 + M[20]$ 6+14+6
5 $1 + M[17]$
3 $1 + M[12]$ 14+6+6





$= \{5, 4, 3, 5, 3\}$



---

for this ex

$$M[i] = \min\{1 + M[i-1], 1 + M[i-4], 1 + M[i-6], 1 + M[i-6], 1 + M[i-14]\}$$

max. value $M[i]$ can take is i

(bec. amount i = $1 * i$)

I/P: $n, A, (c_1, c_2, \ldots c_n)$

Let $M[i]$ be min # of coins needed to add up to exactly "i"

```
M[0] = 0;
for(i=1; i≤A; i++)
{
    M[i] = i;
    for(j=1; j≤n; j++)
    {   if(i-cj < 0) break;
        else M[i] = min(M[i], M[i-cj]+1);
    }
}
```

→ $O(nA)$

Running time = T.C = $O(nA)$

Suppose $(c_n)^3 < A$ (and $c_1 < c_2 < \ldots < c_n$); Assume $c_1 = 1$

Devise algos. with a better running time than $O(n \cdot A)$

~~Ans) $O(n \cdot \lceil A \rceil)$~~

~~Osama~~

~~Do Algo., until remaining amount is multiple of c_n.~~

$c_1 = 1, c_2 = 4, c_4 = 9, c_5 = 14$ ← 26, 31 don't use coin "14"

$A = 18367$

$\left\lfloor \frac{18367}{14} \right\rfloor$

$18367 = 14(x) + \boxed{r}$

$r \leq 13$

$c_n(\ )$

Claim: Every amount $A > C_n^3$ uses atleast one coin of denomination (C_n).

Proof: W/o the C_n coin, the max-denomination is C_{n-1}

$$\left\lfloor \frac{A}{C_n} \right\rfloor = (C_n^2 + C_n - 1)$$

$$A > C_n^3.$$

$$\frac{A}{(C_n - 1)} > \frac{C_n^2 \cdot C_n}{(C_n - 1)}$$

$$> \frac{(C_n - 1)(C_n + 1) C_n}{(C_n - 1)}$$

$$> (C_n^2 + C_n)$$

# coin of deno. 14 $\leq (13)$	C_n use of deno. 14
3 coins $\leq 3 \times 13 = 39$	42
≤ 2548	2744

9, 12 $\quad A = 9x + 12y$

$$= 3(3x + 4y)$$

$$= 3[3x + 3y + y]$$

$$= 3[3(x+y) + y]$$

$A = ax + by \quad a < b \quad b = a + k$

$= ax + (a+k)y$

$= a[x+y] + ky$

L7 21Aug

Knapsack Problem

*) You are a thief

*) There are certain items that are available for you to steal.
Each item has a value and weight.

*) You are carrying a knapsack with a limited maximum weight capacity.

Goal : Maximize the total value of the items that are within the weight capacity of your knapsack.

(4)

Item	Value(V)	Weight (w)	V/w
Ball point pen	2	3	0.67
mattress	5	50	0.1
Gold.	1000	2	500
TV unit	100	500	0.2

maximize value, minimize weight ⇒ So take V/w ratio

Greedy :- Select item with the maximum V/w ratio & keep doing so until knapsack reaches its capacity.

Correct :- giving valid solution. (total weight < max. capacity)

optimal :- valid sol. && maximum total value.

Greedy gives correct but not optimal.

Counterfe:

Item	V	W	V/W
1	7	4	1.75
2	5	3	1.66
3	5	3	1.66

W = 6

GREEDY = 100

OPTIMAL = 011

Knapsack = {1}
by greedy
↓
Value = 7

(Common sense) optimal

Knapsack = {2, 3}
↳ Value = 10

Notation :- Suppose there are n items and their values and weight of the ith item are v[i] & w[i] respectively. and W is the weight capacity of the knapsack.

A solution to the knapsack problem can be denoted as a bit string of length n.

Correct & optimal Algo :-

↳ Brute force :- $O(2^n)$ (Answer is string of length n.
↳ 2^n possibilities, choose max. value among them)

(Q2)

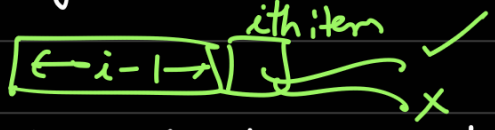
Item	value (v)	weight (w)	v/w
1	5	4	1.25
2	2	3	0.666
3	2	1	2
4	4	3	1.333
5	3	2	1.5

W = 7

Accordy greedy: {5, 4, 3} Total value = 9.

GREEDY = 00111

Suppose you have solution when the first (i-1) items are available for selection.



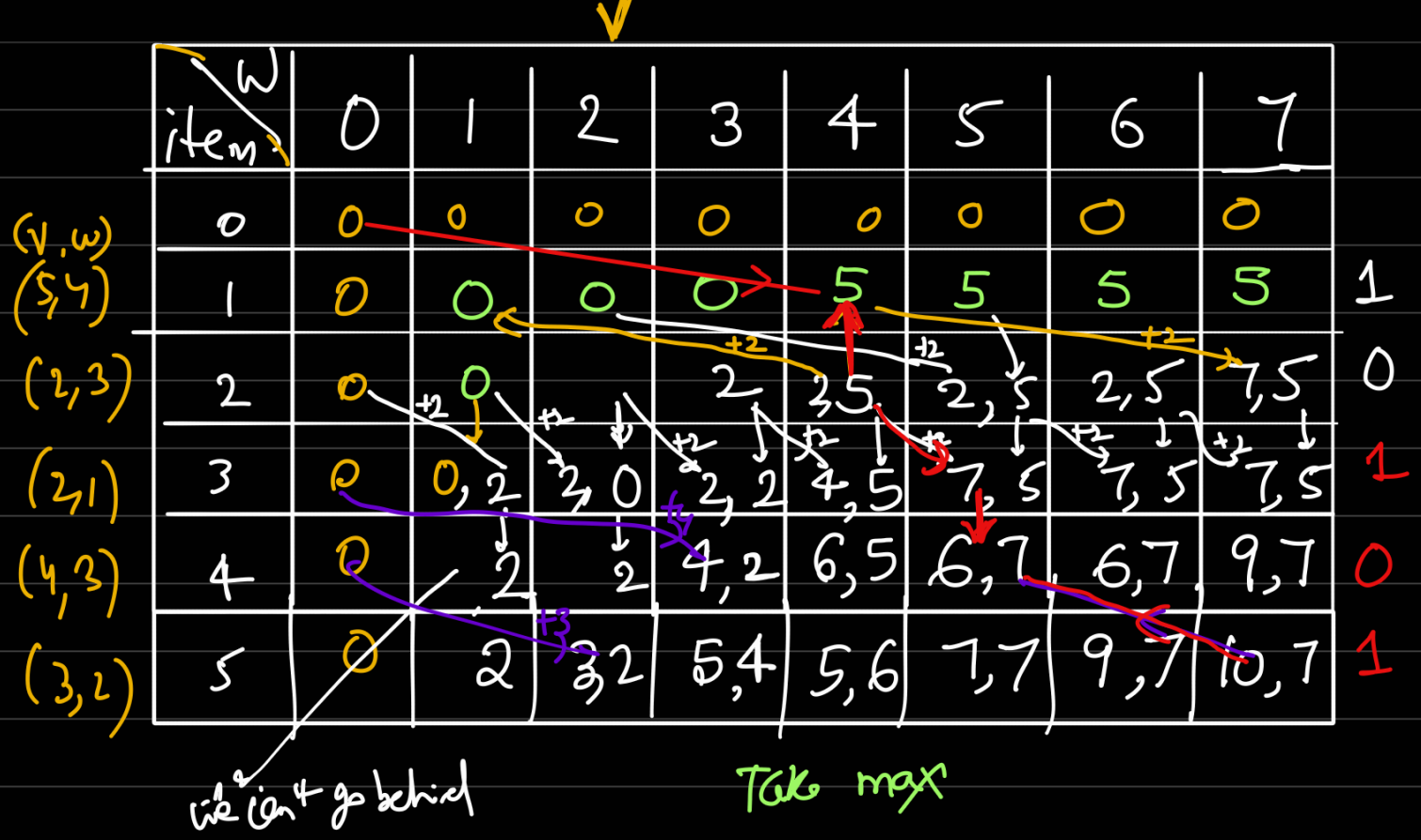
Either the ith item is selected (or) the ith item is not selected.

(Case i): [ith item is not selected]

$V[i][j] = V[i-1][j]$

- $V[i][j]$: the first ith items are available for selection
- j: total weight is capacity of knapsack

(Case ii) ith item is selected



V[i][j] denotes the optimal value of a knapsack of weight capacity j, when the first i items are available for selection. (1, 2, 3, ... i items)

We interested to compute V[5][7]

if i select ith item, go to $V[i-1][j-w[i]]$

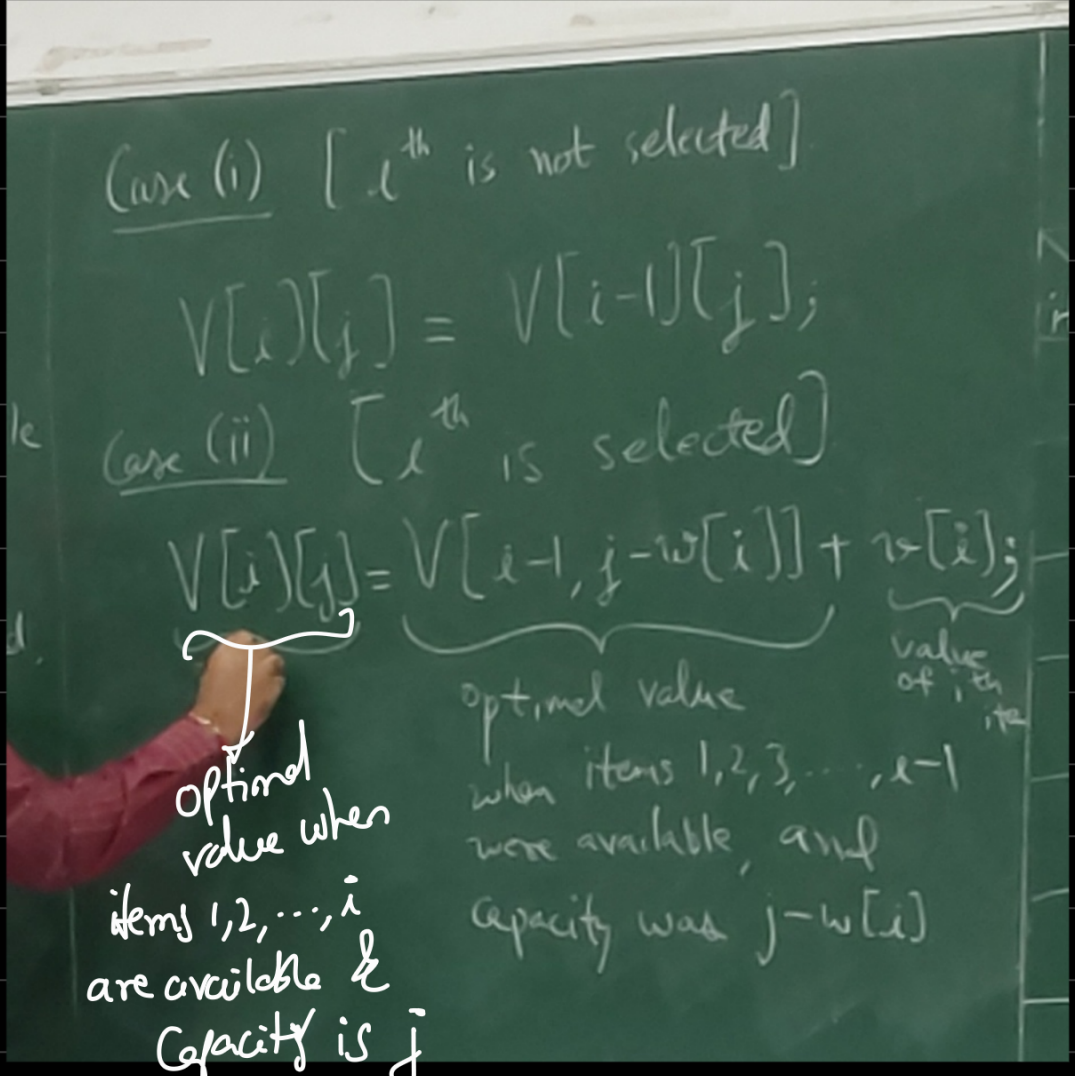
if $(j-w[i]) < 0$; Take is INTMIN

else, Take is $V[i-1][j-w[i]] + V[i]$

↖ → keep 1

↑ → keep 0

V[i][j] = max (Not Take, Take)



$n \rightarrow$ no of items
$w \rightarrow$ given max. capacity of knapsack

$V[0][j] = 0 \quad \forall j = 0, 1, 2, \ldots w$

$V[i][0] = 0 \quad \forall i = 0, 1, 2, \ldots n$

```
for(i=1, i<=n; i++)
{
    for(j=1; j<=w; j++)
    {
        if(j-w[i] >= 0)
            V[i][j] = max(V[i-1][j],
                          V[i] + V[i-1][j-w[i]])
        else
            V[i][j] = V[i-1][j]
    }
}
```

Find o/p :- V[n][W]

(What is 3*n as I/P
row is value, weight, Item)

V[5][7] is our ans.

Trivial Sol
(Subhadeep)

(I/P :- n, W)

TC
O(n*W)

SC
O(n*W)

→ we can optimise by storing ith row & (i-1)th row

→ we have (n+1)(W+1) subproblems to solve each take O(1).

Definition :- An algorithm is called polynomial-time Algo. if its running time (T.C) is atmost a polynomial in the size of the input.

I/P Size :- x

T.C :- x^2 (quadratic)

2^x (not polynomial)

IITKanpur 2002
AKS primality test

$\underbrace{11111}_{\text{5 bits req}} \Rightarrow 31 \rightarrow 2^5 - 1$

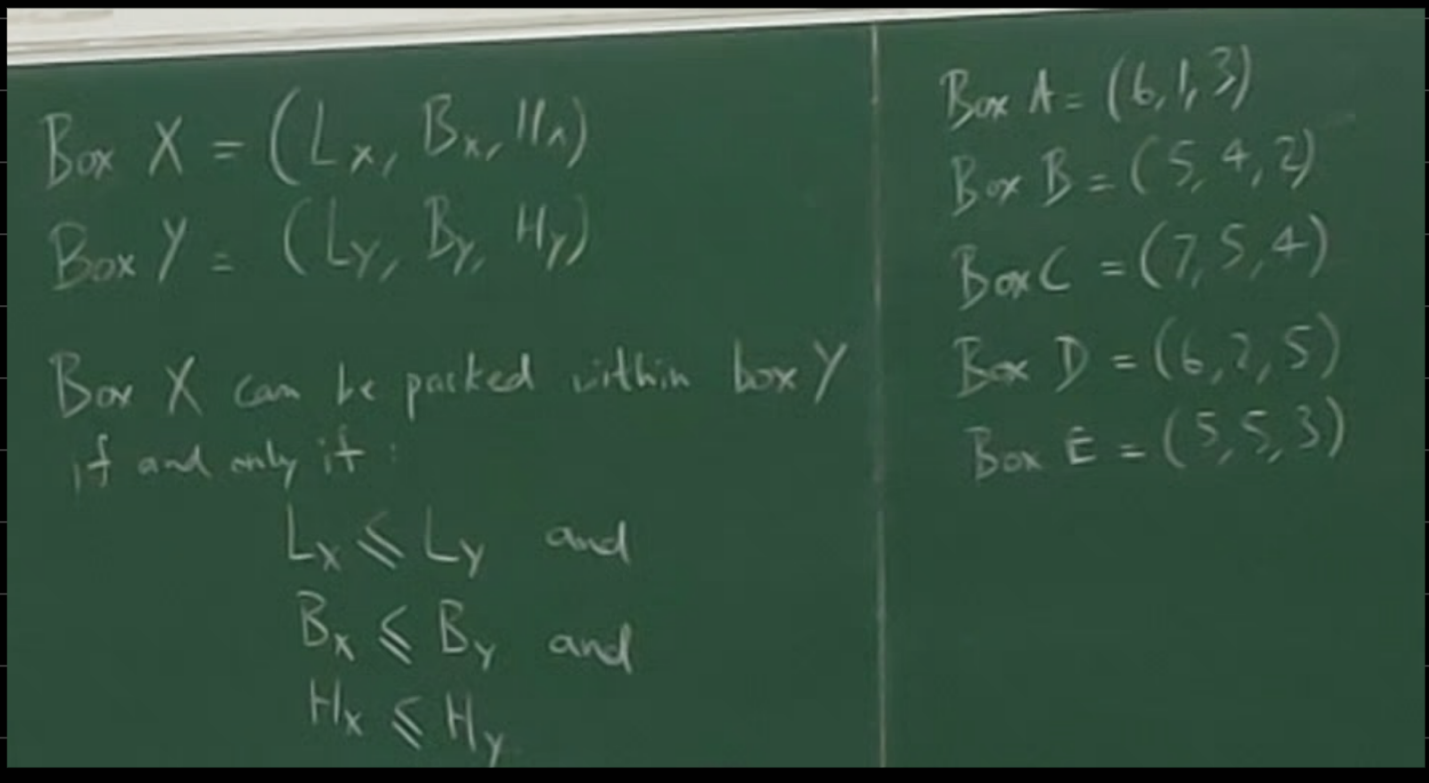
Size of input $\geq 3n\log n + \log_2 W$

L8 25-Aug-25

L9 28Aug25

Box Packing

You have no. of cuboidal boxes which you want to store efficiently. You want to minimize wastage of space in ur store room by packing them within each other. No two of boxes have exact coordinates.



A can be packed within C. $A < C$
A can be " " D (6,1,3) (7,5,4)
B can be packed within E.
E " " " C

Q) What about A & B?

A) Neither one can be packed within other.

$$\overset{A}{(6,1,3)} \not\leq \overset{B}{(5,4,2)}$$

A, B incomparable.

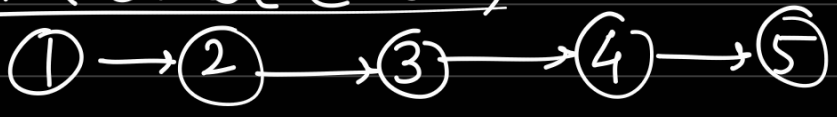
$$\overset{A}{(5,4,2)} \not\leq \overset{B}{(6,1,3)}$$

There are n non-identical boxes: (L_i, B_i, H_i) $\forall i = 1$ to n

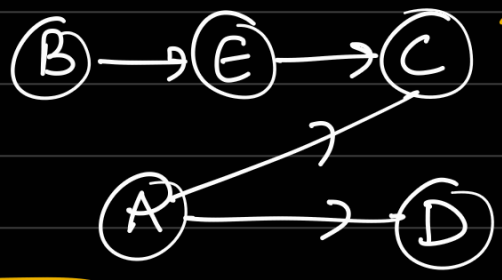
A sequence of boxes s.t. each box can be packed within the next box in that chain (sequence)

u have small, large box. u can fit only 1 small box in large box no matter how much space left

Total Order (chain) :- → just example



Partial Order



→ wrt above eg.

chains in this are

B→E A→C
E→C A→D
B→E→C

(A can go inside the D)

POSET

A **partially ordered set** or **poset** is a set P and a binary relation $\preceq$ such that for all $a, b, c \in P$

1. $a \preceq a$ (reflexivity).
2. $a \preceq b$ and $b \preceq c$ implies $a \preceq c$ (transitivity).
3. $a \preceq b$ and $b \preceq a$ implies $a = b$. (anti-symmetry).

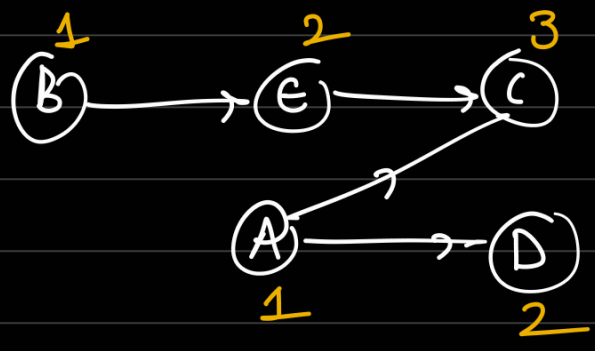
Examples

1. $P = \{1, 2, \ldots, \}$ and $a \leq b$ has the usual meaning.
2. $P = \{1, 2, \ldots, \}$ and $a \preceq b$ if a divides b.
3. $P = \{A_1, A_2, \ldots, A_m\}$ where the A_i are sets and $\preceq = \subseteq$.

D-P

Recall:- In coin denominations M[i] was defined as the min. # coins needed to add up to exactly the amount i

Now, In box packing, M[i] is defined as max. no of boxes in a chain ending at box i (including box i itself)



no. indicates longest chain that ends at box.

* for box i, look at all the earlier boxes 1, 2, 3, ... i-1

* which of these earlier boxes can be packed within box i ?

For eg: i = 23

Suppose boxes 4, 7, 13, 18 can be packed within box (i=23)

M[4] +1
M[7] +1
M[13] +1
M[18] +1

} max = m[23]

↳ longest chain ending at 18, now u put (18) in 23 so add 1

Analysis:

① Correct. ✓ (It's obvious)

② Optimal ⟶

a) How u say this optimal?

1 Box A = (6, 1, 3)
2 Box B = (5 4 2)
3 C = (7 5 4)
4 D = (6 2 5)
5 E = (5 5 3)

Approach 1 :-

Lets think, for 3, we check previous boxes 1, 2.

See box C (i.e. 3): $M[3] = 1 + \max(M[1], M[2])$

$= 1 + 1$

$= 2$

But in reality, $M[3] = 3$.

∴ our approach is wrong.

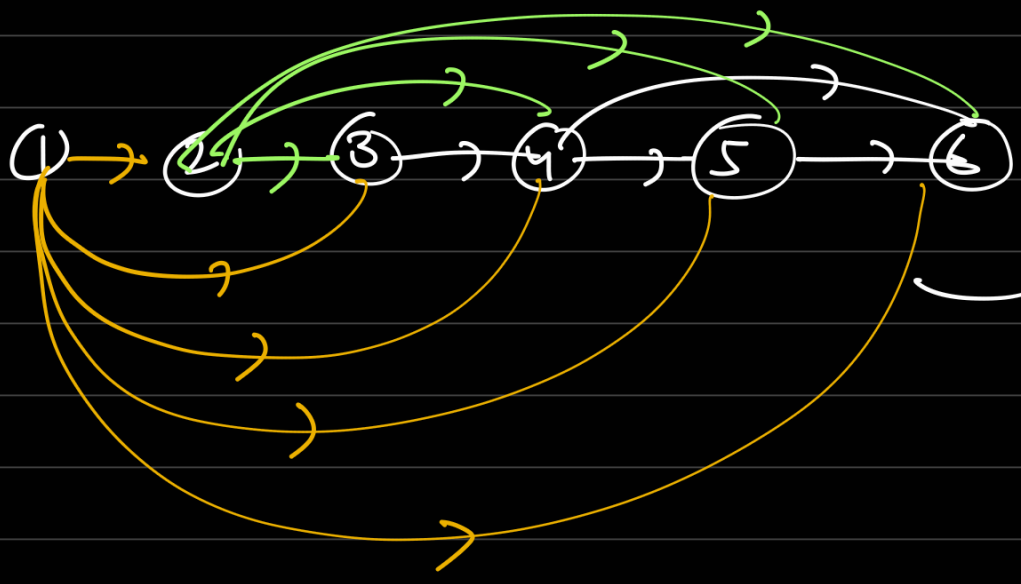
There is no notion of ordering boxes. bec. some of elements is incomparable.

Approach 2 : sort coordinates by x, if same they y then z

1 Box B (5, 4, 2)
2 Box E (5, 5, 3)
3 Box A (6, 1, 3)
4 Box D (6, 2, 5)
5 Box C (7, 5, 4)

Lets take from ① to ⓝ, each edge:

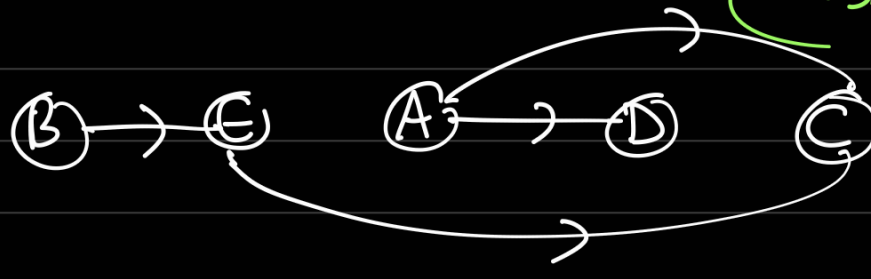
ⓐ → ⓑ it means $a \leq b$.



→ Totally ordered set.

Ⓐ No. of arrows from ① to ⓝ if all arrows drawn is $\left({}^{n}C_{2} = \frac{n(n-1)}{2}\right)$

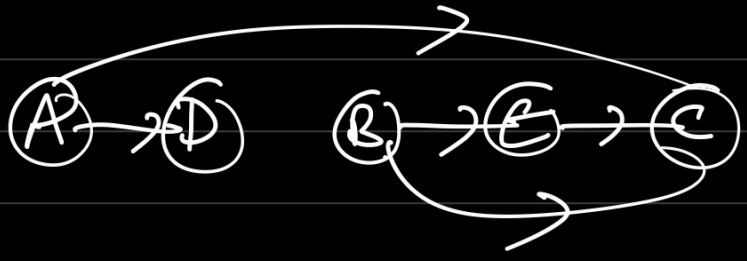
Arrows are left to right (All boxes we pack appears left)

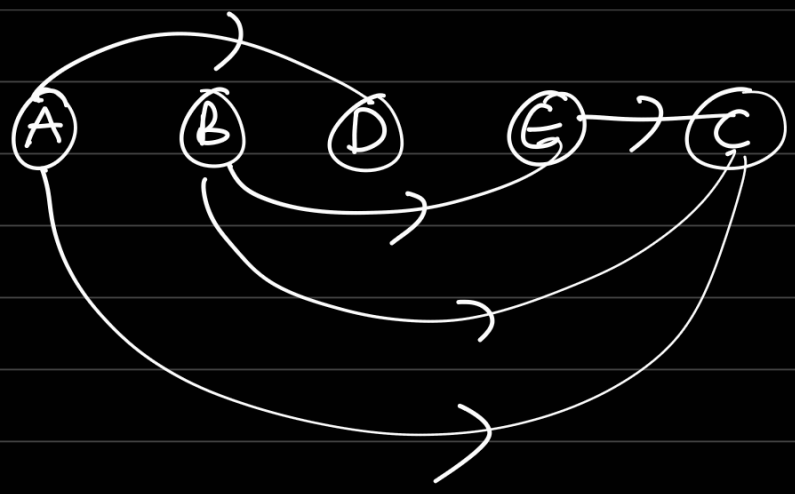


→ Poset.

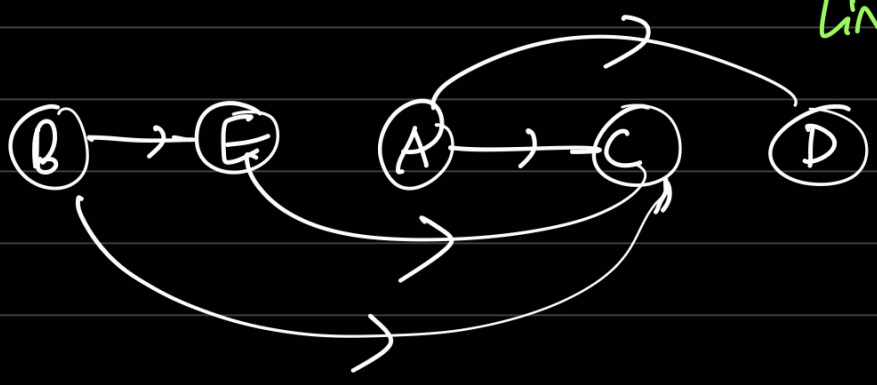
we are making from T.O. set, removing some edges to make it as Poset.

This is proof for optimality.





Linear extension



Q) How many optimal s/n - will be there?

Suppose n boxes. They are not totally ordered. They are partial ordered

How many linear extension possible there can be?

↳ left to right arrows.

Ans) W.C → n!

Given partial ordered set, $|S| = n$. # linear extension = ?

W.C = n! if there no arrows then we can arrange in n numbers

Avg n!

For T.O.S; we have exactly 1 linear extension.

---

we can have many optimal K length. but this does always give optimal length (K).

D-P Algorithm: (Crct & optimal)

Let say u have poset.

$M[i] = 1$ for all $i \in \{1, 2, 3, \ldots n\}$

find no. of comparable with "i" among $\{1, 2, 3, \ldots i-1\}$ | let say K are present $l_1, l_2, \ldots l_K$

$$M[i] = 1 + \max(M[l_1], M[l_2], \ldots M[l_K])$$

This is exponential time.

1. Arrange the boxes from L to R. st. every arrow goes in forward (left to right) direction.
   → There are many ways to do it.
   one easy way : we have seen in class (Osama told) person in day
2. Look at all possible paths.
3. Select the longest path & o/p that as the answer

pathlength is 3 for a → b → c
↳ see #nodes.

At most $O(n^2)$ arrows in every poset on n elements.

1, 3 are polynomial time

(2) step is not polynomial time bec.

#poss chains in W.C = 2^n

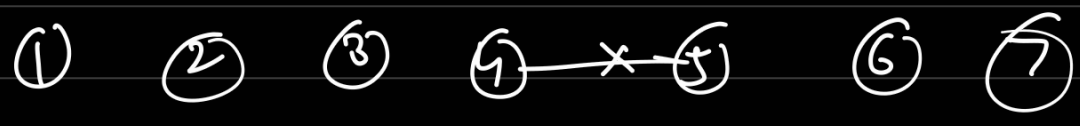
Let say u have T.O.S · ($^nC_2$ arrows)

// remove 2/3 arrows from T.O.S, it make Poset.

- no. of chains in T.O.S is exactly 2^n bec.

#Subset of T.O.S is 2^n.

every subset of T.O.S give us one chain. $2^1 - 1 = 20$

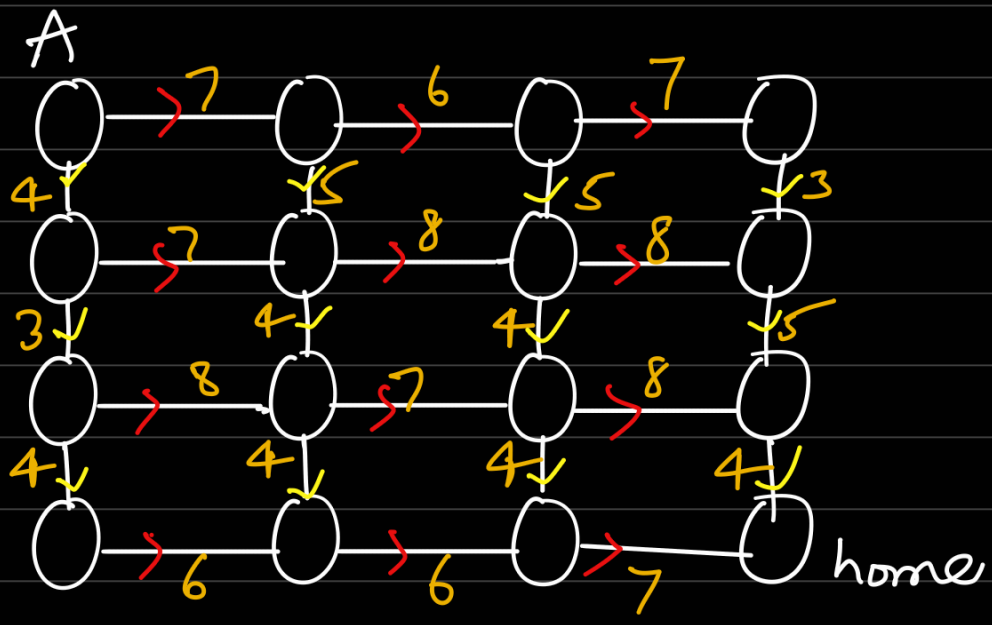


- see we work on Poset not Toset:

So just in Toset remove 1 arrow, that becomes Poset.

L10 1st Sept 25

Getting home (4×4 grid graph)

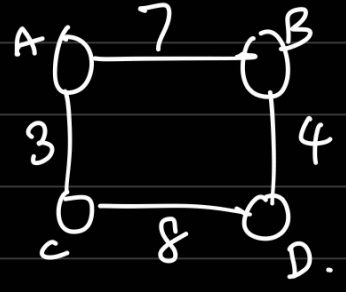


edges → roads
edge no. → time taken

* A shortest path (or a quickest route) always takes edges (or roads) in the left to right (or) the above to below directions, since only these edges (or roads) will take you closer to home.

→ Its assumption :- (i.e. graph has only → & ↓ edges)

Take any grid, below is true.

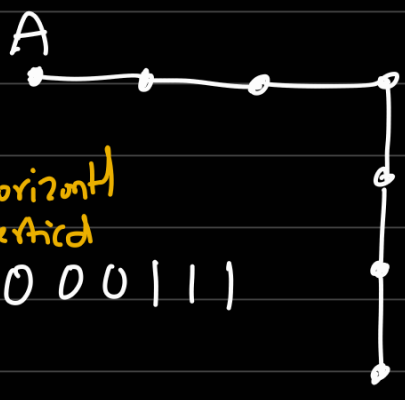


C - D → 8

C - A - B - D → 3 + 7 + 4 = 14

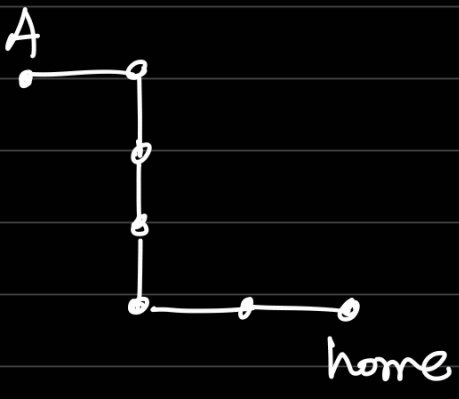
$$\left(\frac{6!}{3!3!}\right)\left({}^{12}C_3 \cdot {}^{9}C_3\right)$$

How many paths are there from A to home?

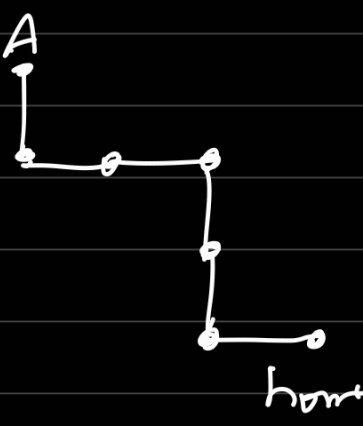


0 → horizontal
1 → vertical

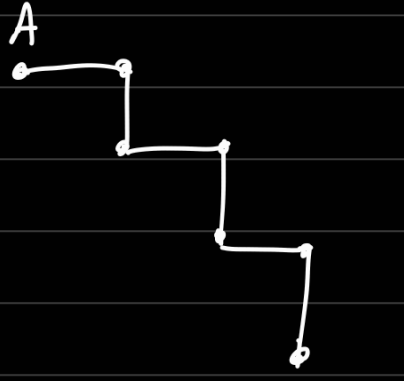
000111



011100

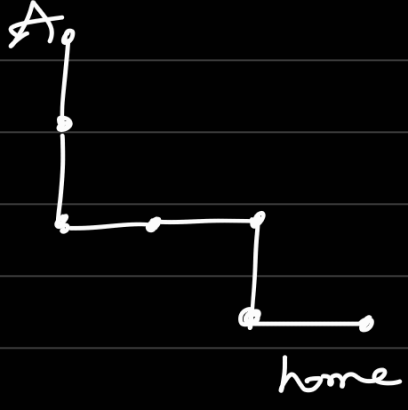


100110



010101

111000



110010

* Every path from A to home can be represented by a balanced binary string of length 6.
  → (reduction)

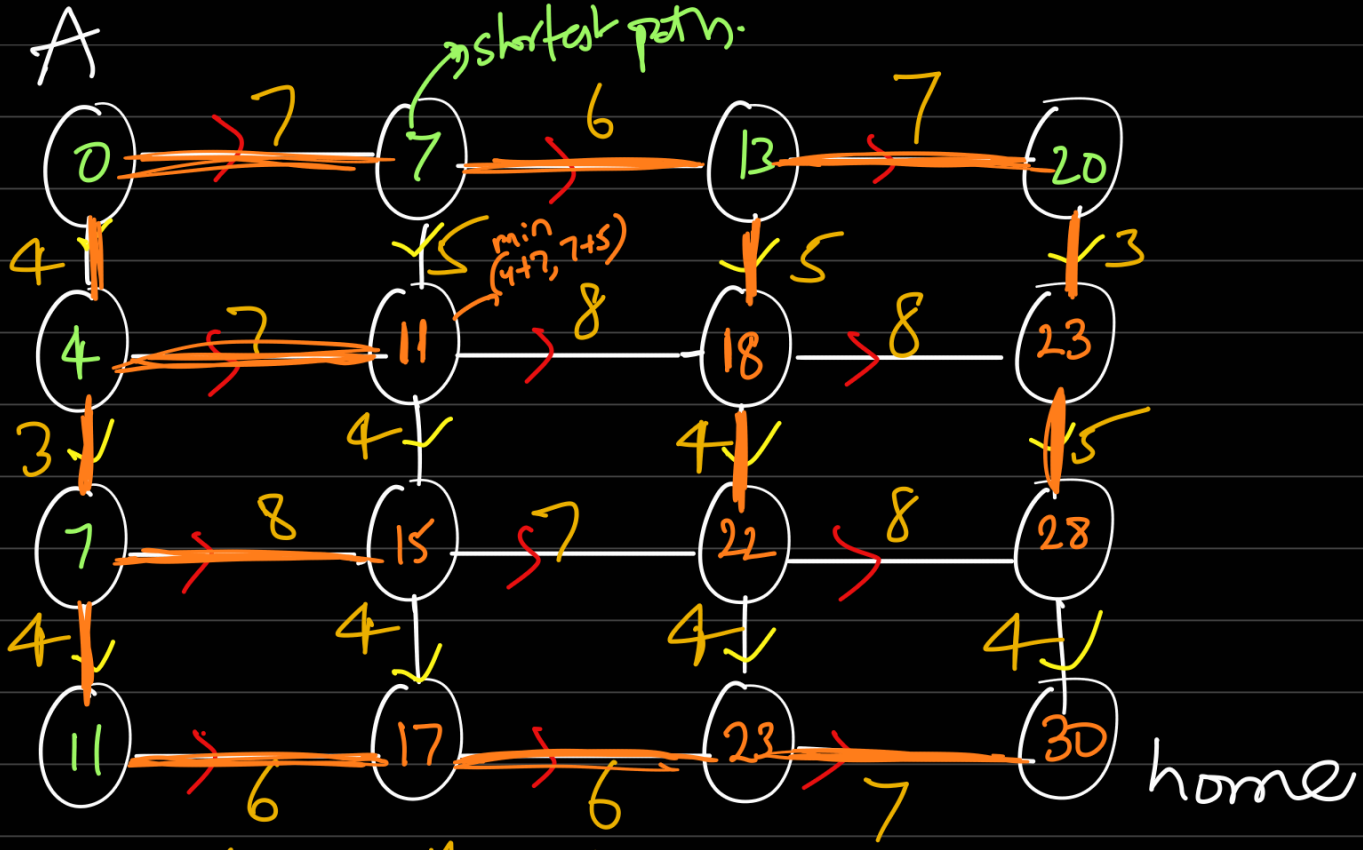
Q) How many balanced binary strings of length 6 are there as above?

A) $\frac{6 \times 5 \times 4}{3!} = \binom{6}{3} = 20.$

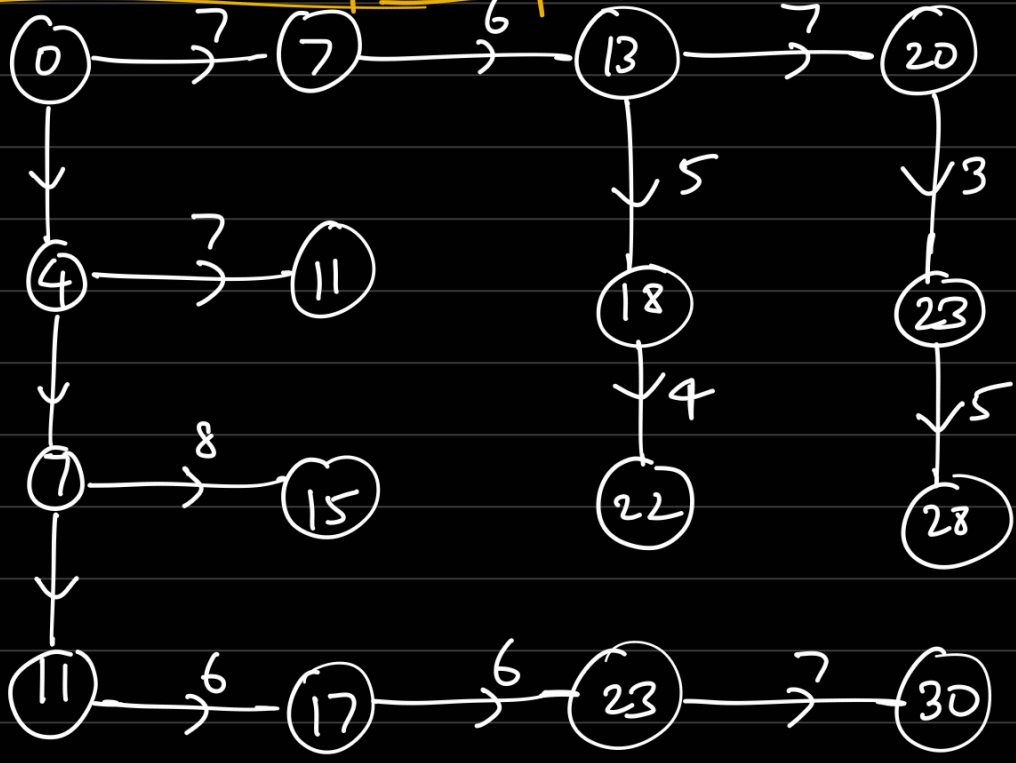
Suppose $(n+1) \times (n+1)$ grid graph,

No. of paths from A to home $= \binom{2n}{n} \approx \exp(n)$

[Stirling's approx.]



Just draw shortest path graph:



In this shortest path we have 15 edges

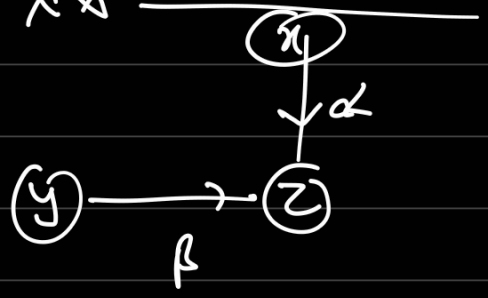
If we change weights. still we have 15 edges.

why?

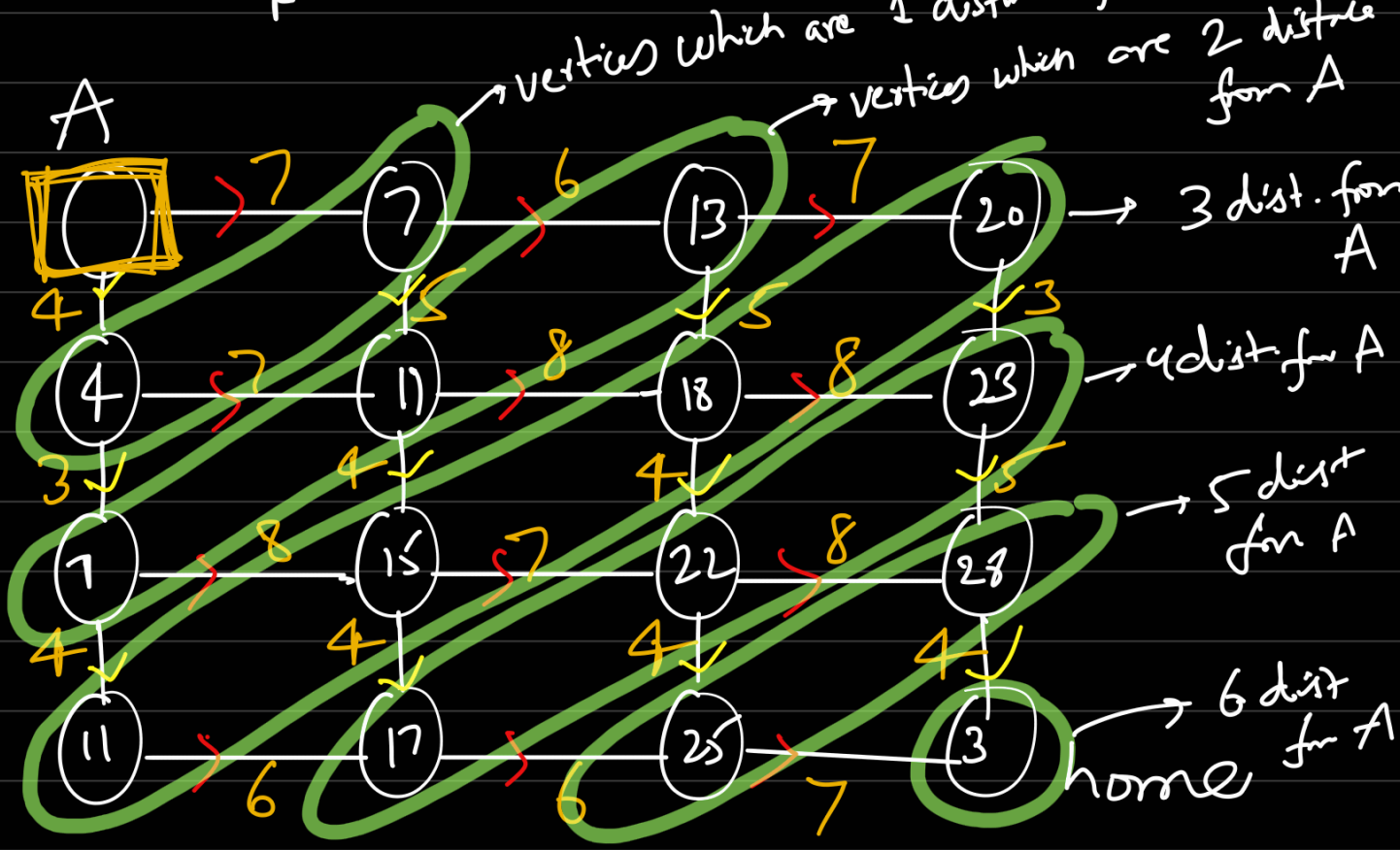
sum of indgree = (|E|-1)

In SP-tree,
16 vertices. 15 vertices has 1 incoming vertex
intial vertex has 0.

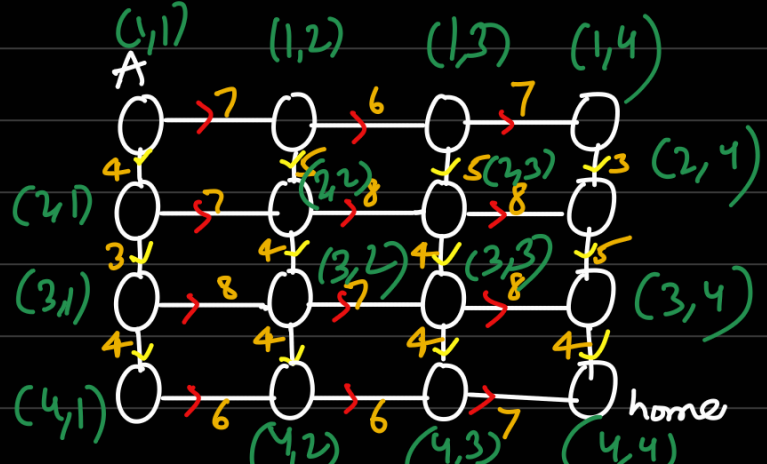
∴ we have 15 edges.



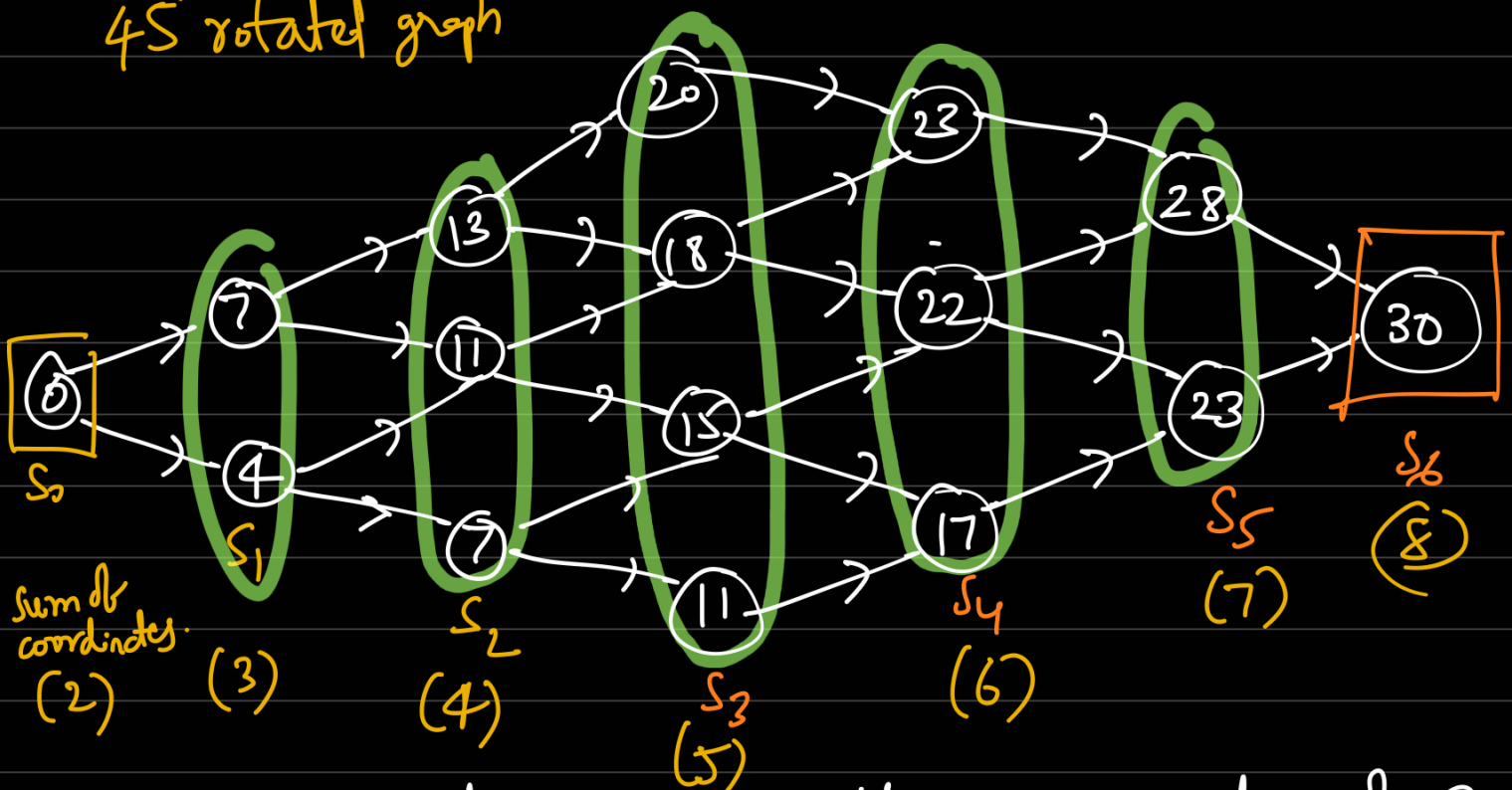
$z = \min\{x+\alpha, y+\beta\}$



S_i → set of vertices reachable from 0 at distance i



45° rotated graph



Every vertex of S_2 comes after every vertex of S_1.

Now make compress them, it become 1D.

When u compress, array each set, how u keep them in 1D order?

(i,j) is in set (S_{i+j-2}).

→ doubt ↳ refer other notes

(2 points (in same set) are reachable to each other if $x_1 \neq x_2$ & $y_1 \neq y_2$
(x_1, y_1) (x_2, y_2)

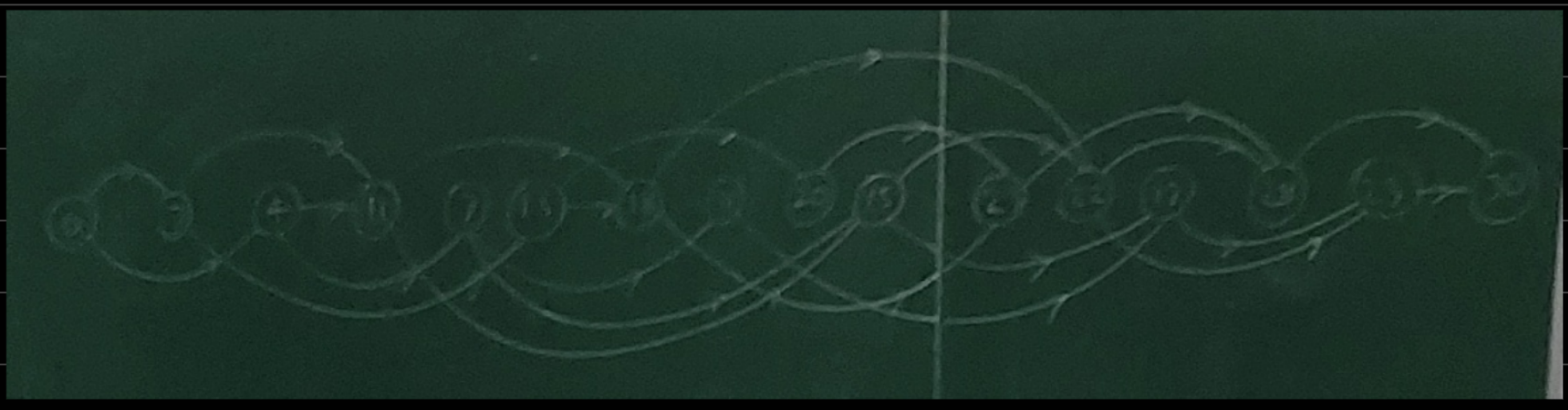
(Within single set, we can't have be any arrows b/w them)

all have same sum $P_1(x_1, y_1)$ $P_2(x_2, y_2)$

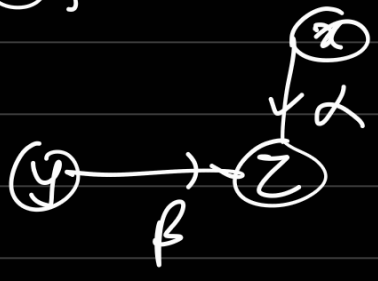
$x_1 + y_1 = x_2 + y_2$

we can keep points in any order.

Directed Acyclic graph (DAG).



Alg :- ① make a graph as 1D (Linear extension)
② for vertex, look its incoming vertices.



$z = \min\{x+\alpha, y+\beta\}$

if u keep $\max\{x+\alpha, y+\beta\}$ then by this algo., u can get longest path.

③ If all edge weights are negative then take their absolute value, compute a longest path and output shortest path = (-1) * longest path

short	longest
$10 < 30$	
$-10 > -30$	

longest path = $(a+b+c+d)$

↓

original graph : $-a-b-c-d$

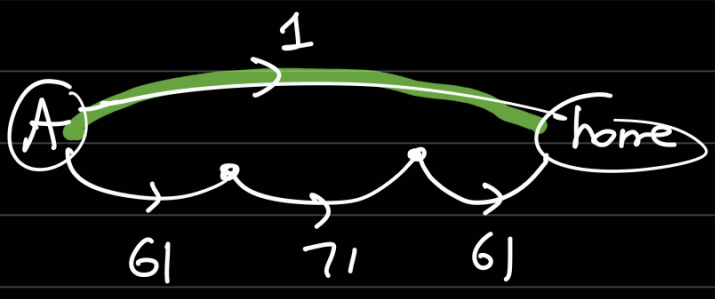
$= -(a+b+c+d)$

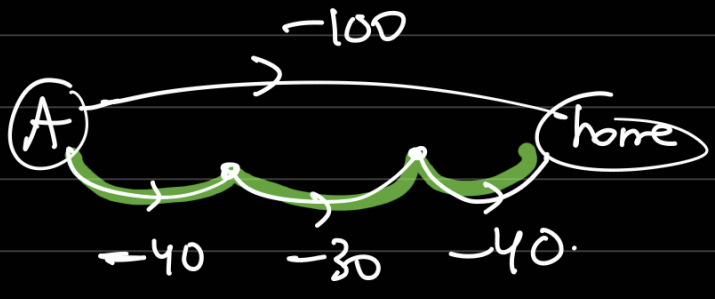
$= -[\text{longest path}]$

Q) If u have above graph as some negative weights will above algo. work?

A) Yes. for above graph it work → (Why?)

observe: Take any vertex, (V) all paths from intial to V have same path length





L11 8-Sept

Addition

digit : [0 – 9]

possible operations when adding 2 single digit numbers = 10×10 = 100.

$x + y$
↓ ↓
10 10

0+0
0+1
2+3
} 3 possibility

*) adding 2 numbers of n digits each gives a number of n digits (or) a number of (n+1) digits.

* Computation for each digit (each column) ≤ 2   1+2+3 (1) 2 3

* Total computation ≤ 2n = O(n)

Q) T.C of adding 2 numbers a and b is

No. of single digit additions to add up 2 numbers a & b is $O(\max\{\log_{10} a, \log_{10} b\})$

```
  2999999
 +      1
 --------
  3000000
```

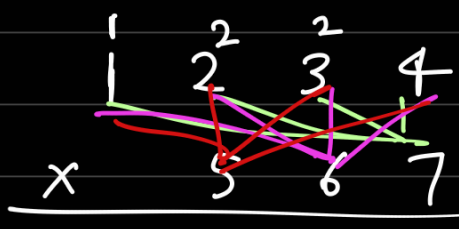
Multiplication

digit : [0 – 9]

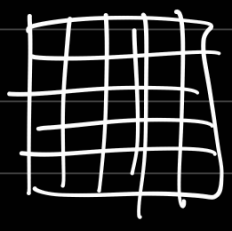
possible operations when multiplying 2 single digit numbers = 10 × 10 = 100.

→ Store all 100 in table.

School method:



$$
\begin{array}{r}
1234 \\
\times\ \ 567 \\
\hline
8638 \\
7404\times \\
6170\times\times \\
\hline
699678
\end{array}
$$



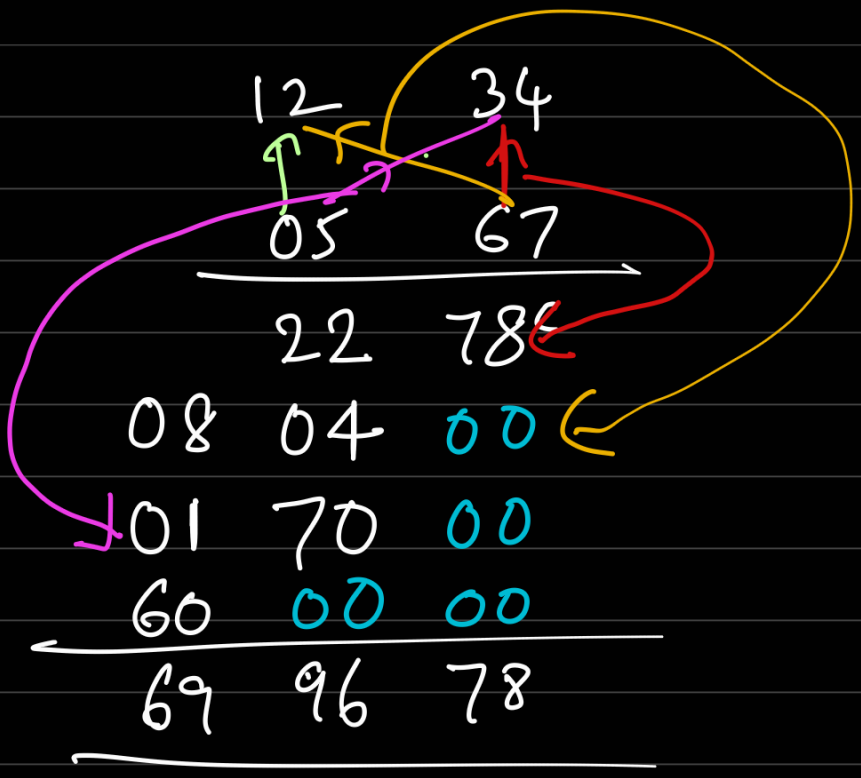
Total multiplications performed = 4 + 4 + 4 = 12

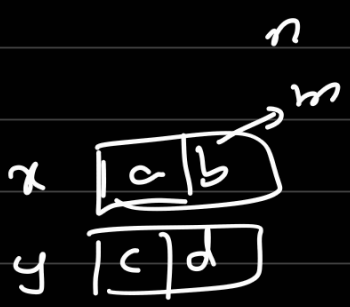
= (4 × 3) → 3 digits in 567

4 digits in 1234

No. of single digit multiplications to multiply up 2 numbers a & b is $O\left[(\log_{10} a)^{*}(\log_{10} b)\right]$

Multiplying 2 n-digit numbers in conventional way is n^2 multiplications.





Let T(n) be the T.C of multiplying 2 n-digit nos

$$27651398 \times 18324469$$

2765	1398
1832	4469

×

// 1398 × 4469

// 2765 × 4469 [0000]

// 1832 × 1398 [0000]

// 2765 × 1832 [0000 0000]

$$T(n) = 4T(n/2) + O(n)$$ (no. multiplications)

$$= 4(4T(n/4))$$

$$= 4(4(4T(n/8)))$$

$$= 4^3 T(n/2^3)$$

$$= (2^2)^K T(n/2^K)$$

When does $(n/2^k)$ equals to 1?

Ans) $n = 2^k$

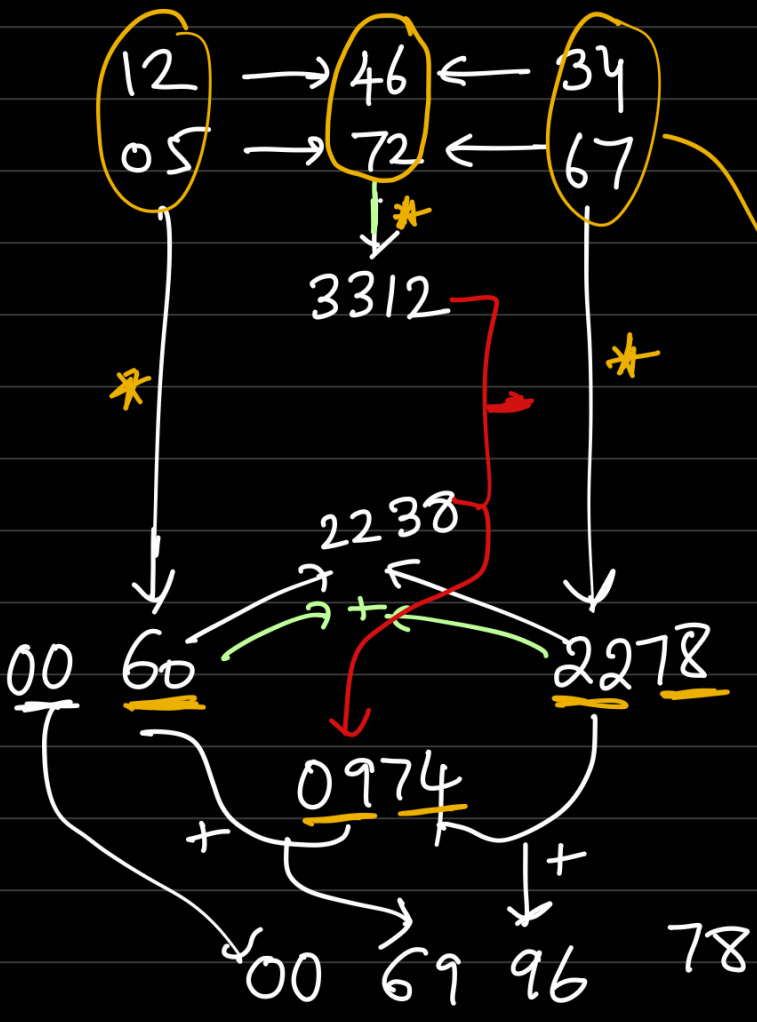
$$\boxed{k = \log_2 n}$$

$$T(n) = (2^2)^k T(n/2^k)$$

$$= 2^{2\log_2 n} T(1)$$

$$= n^2 \cdot T(1)$$

$$= n^2$$



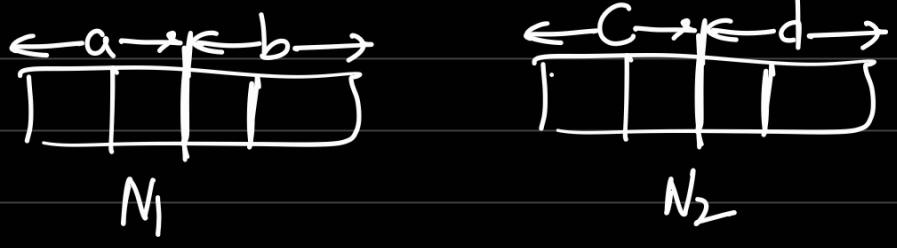
Let addition coming for free

This color, multiplication is costly.

+, − both same cost

Why does it work?

4 digit no.



$N_1 = 100a + b$ $\qquad$ $N_2 = 100c + d$

$N_1 N_2 = (100a+b)(100c+d)$

$= 10^4 ac + 100\,ad + 100\,bc + bd$

just shifting

we are not counting 10^4 as multiplication

4 multiplications.

$10^4(ac) + 10^2(ad+bc) + bd$

Compute (1 multipl)

Compute (1 multipl)

Given ac, bd.

No need compute 2 multipl — we will use 1 multipl

Compute, ad+bc using only one extra multiplication operation

$(a+b)(c+d) = ac + \boxed{bc+ad} + bd$

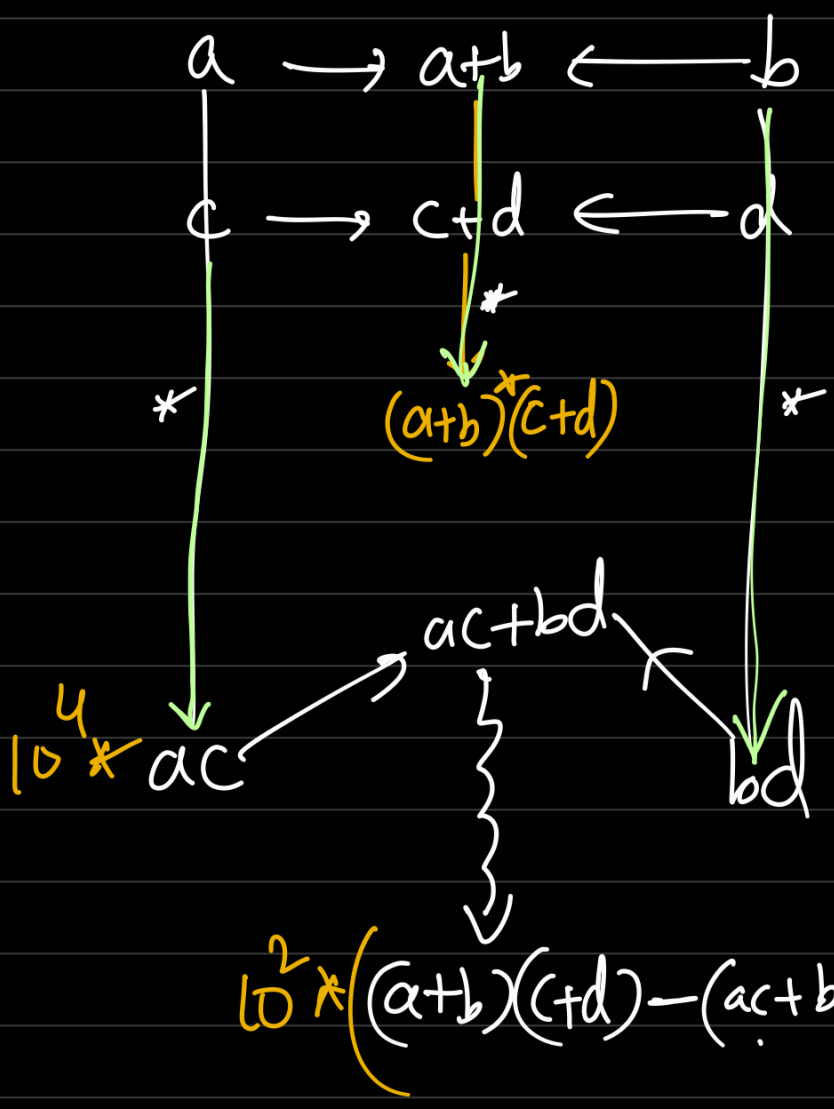
This product using 1 multiplication

$(a+b)(c+d) - (ac+bd) = (bc+ad)$

Now I brought down from 4 to 3 multiplication

$$10^4 ac + 10^2\left[(a+b)(c+d) - ac - bd\right] + bd$$

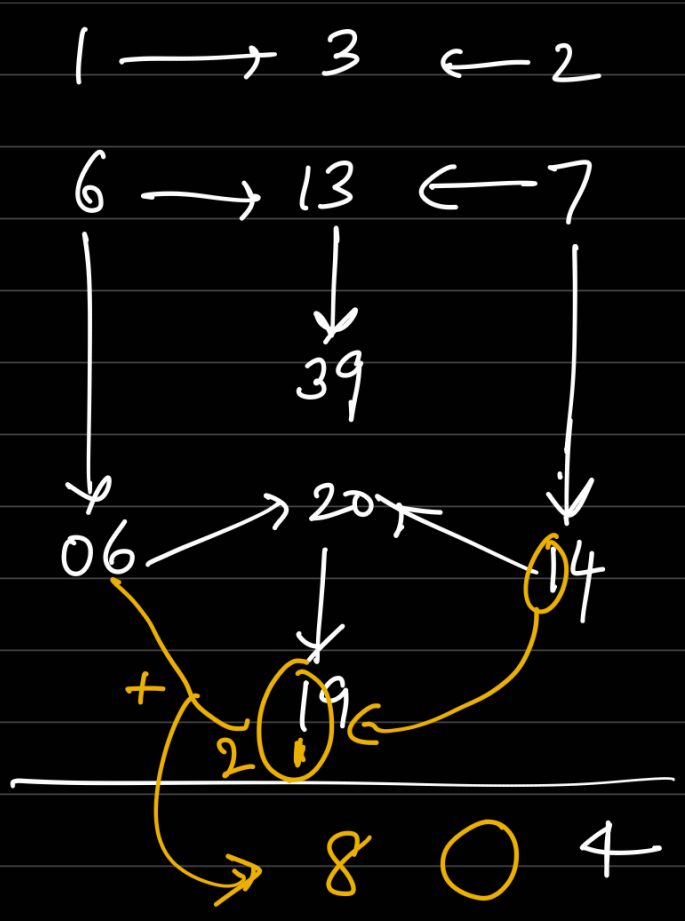
↑ increase # additions to compensate # multiplications



$$\text{Final Ans} = 10^4 ac + 10^2\left((a+b)(c+d) - (ac+bd)\right) + bd$$

(eg) $\begin{array}{r} 12 \\ \times\ 67 \\ \hline 804 \end{array}$ → (osama asked)

with single digits, u need to deal with double digit



Karatsuba's Algo

w/o Karatsuba, if u do conventional way
then its $T(n) = 4T(n/2)$

Let $T(n)$ be the T.C of multiplying 2 n-digit nos

$$T(n) = 3T(n/2)$$

$$T(n) = 3[3 \cdot T(n/2)]$$

$$= 3^k T(n/2^k)$$

$$= 3^{\log_2 n} T(1)$$

$$\boxed{T(n) = n^{(\log_2^3)}} = n^{1.585}$$

$\log_2^3 \leq 1.585$

Conventional method $= n \times n$

Karatsuba's Algo [1960] $= n \times n^{0.585}$

Fast Fourier Transform [1968] $= n \times \left[\log_2 n \times \log_2(\log_2 n)\right]$

Galactic Algo [2019] $= n \times \log_2 n$

$C \times n \times \log_2 n$

Constant is so much larger than our computer holds

In practice, people use fast fourier transform Algo.

Till Now,

D.P
- Coin Denomination (general)
- Box packing
- longest common subsequence
- shortest paths on a D.A.G.
- Fibonacci series
- Knapsack (0/1).

D&C
- Peak finding
- closest pair

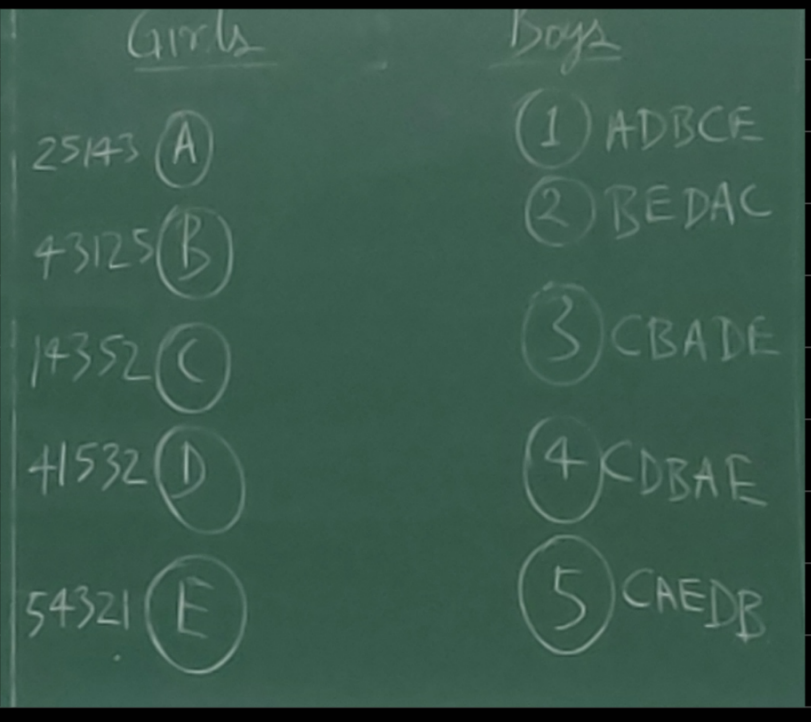
- Merge Sort
- Skyline.
- Karatsuba's Algo.

Greedy Algo.

- Coin denomination (Greedy)
- Interval Scheduling
- ?

L12 11th Sept

Stable Marriage Problem



Unstable pair

A (b, g) pair is said to be unstable if the following are true:

(i) b & g are not married to each other

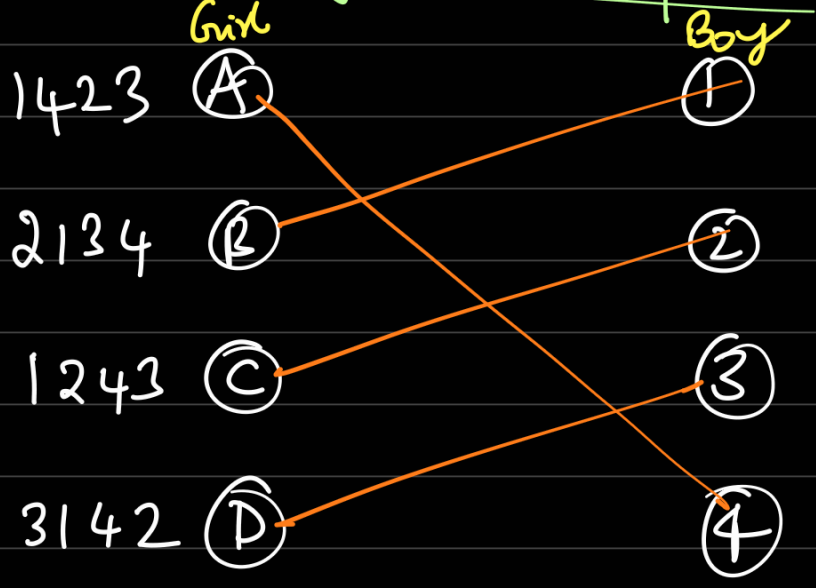
(AND)

(ii) b prefers g over his current wife

(AND)

(iii) g prefers b over her current husband

Example of unstable pair



Orange indicates, let say by fate, they married.

Observe (B, 1) is unstable pair. &

Now they break off

Now (A, 1) is married.

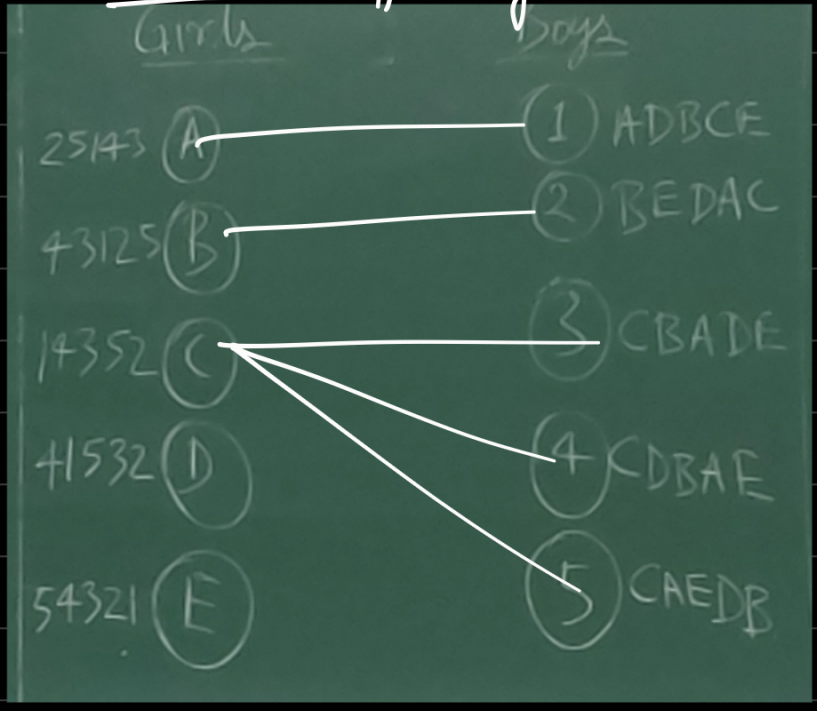
finally:

	Girl	Boy	
1423	A	1	BDAC
2134	B	2	BACD
1243	C	3	CBAD
3142	D	4	DBAC

Gale Shapley Algo → Algo. is greedy. each boy is greedy
each girl is greedy

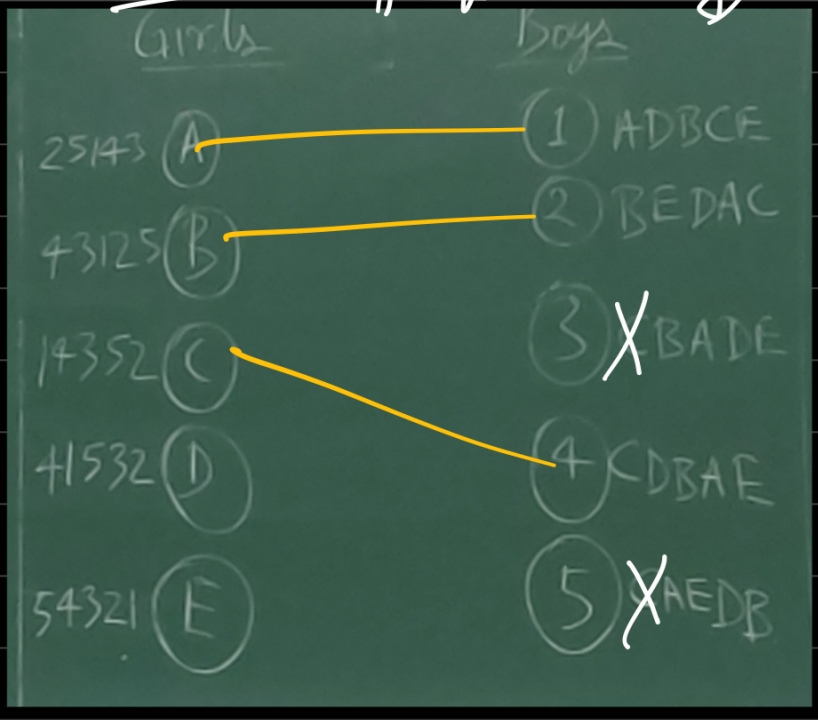
* Intially everyone is un-engaged.
* Every morning, each unengaged boy propses to the topmost girl who has not yet rejected him

1st round of Boys



Every evening, each unengaged girl who received a proposal gets engaged to the topmost boy on her list, if its higher than her currently engaged boy

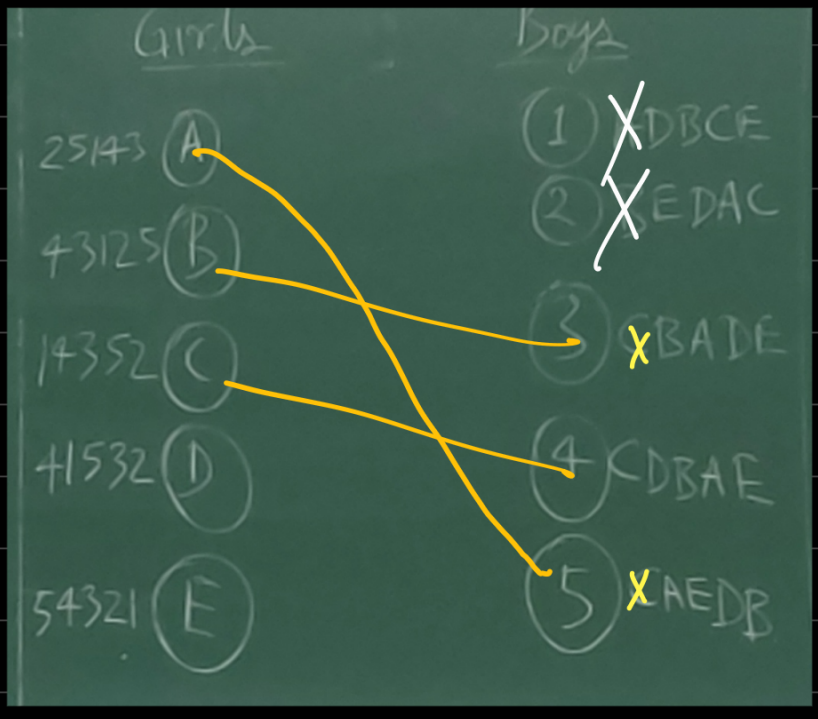
1st round of girl evening)



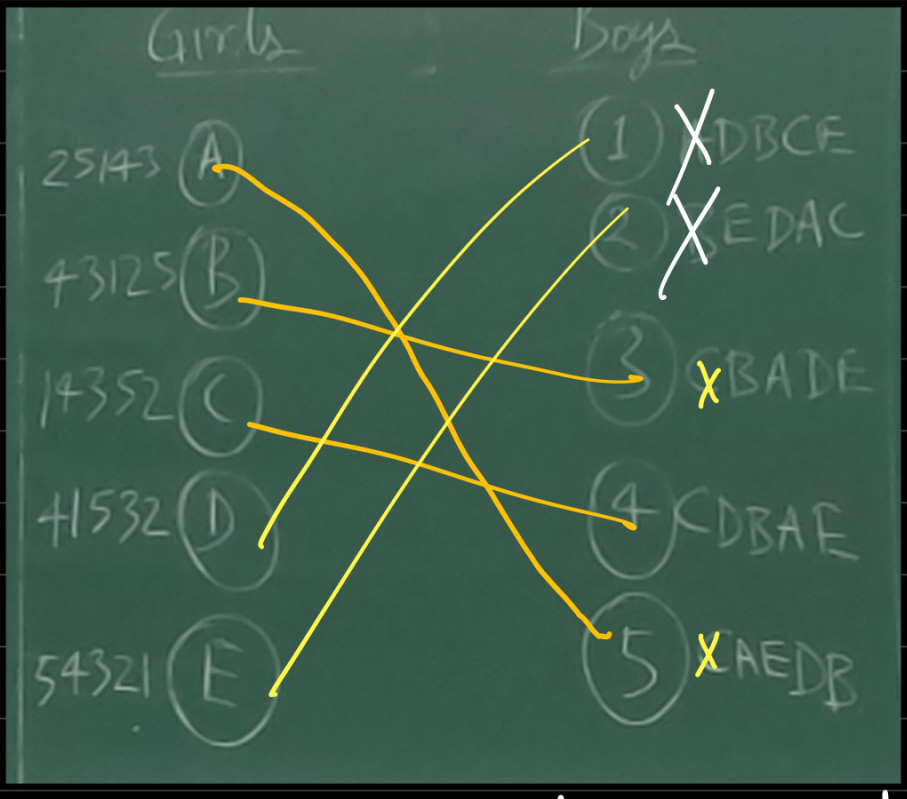
Thus boys gets engaged so next morning they can't go propose another girl.

C is having 3, 4, 5
topmost pref. of C is 4

2nd Morning. 3, 5 proposes to girl.
Then engaged girl will breakoff if she gets better choice



3rd may 1, 2 go to girls.



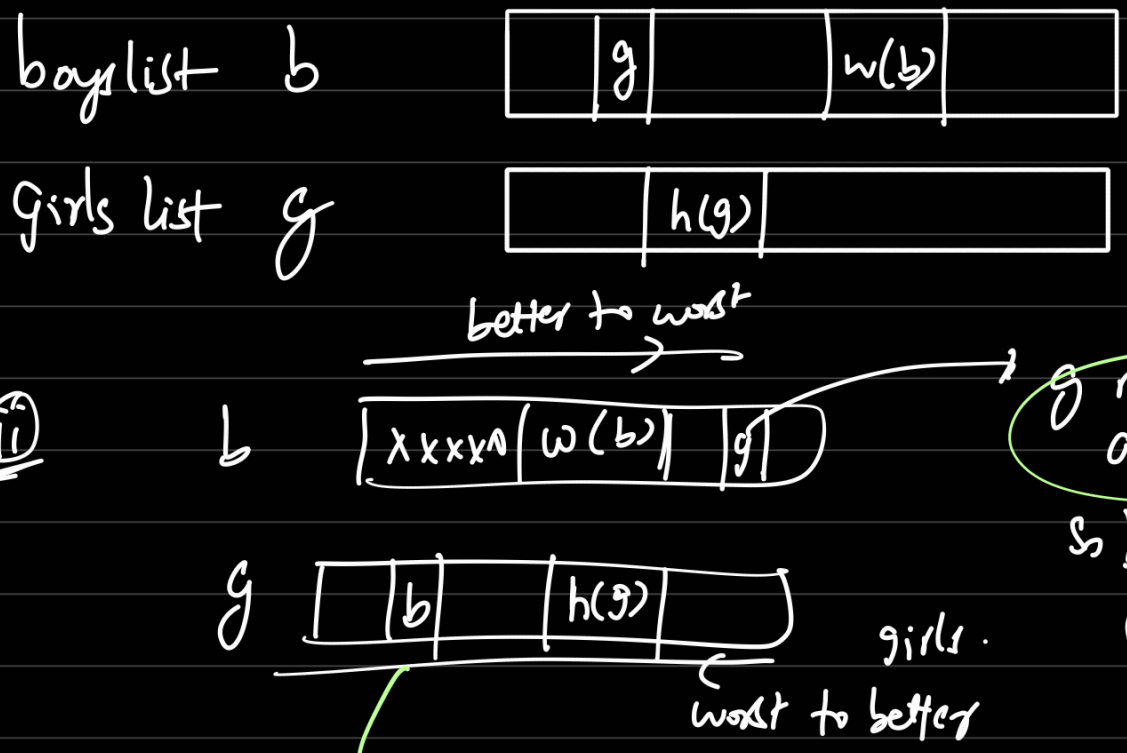
Now all boys gets engaged. So they donot go next day

∴ Algorithm terminates.

④ Alg. terminates when no boys go out to propose in the morning

Stable matching A-5, B-3, C-4, D-1, E-2

If (ii) is true then (iii) is false.
If (iii) is true then (ii) is false.
} why?
↳ Sir told in class.



(Medical intern) (many hospitals)

Dev.	Apple
Web Dev	Microsoft
Tester	Amazon
Prompt Engin	Meta
AI tools	Google
Data Scient	X
Systems engineer	Netflix
S/W Architect	IBM

T.C = $O(n^2)$

It's linear in size of input

I/P

girl $n \times n$.

boy $n \times n$

n girls.

n boys.

each girl has n preferences in it