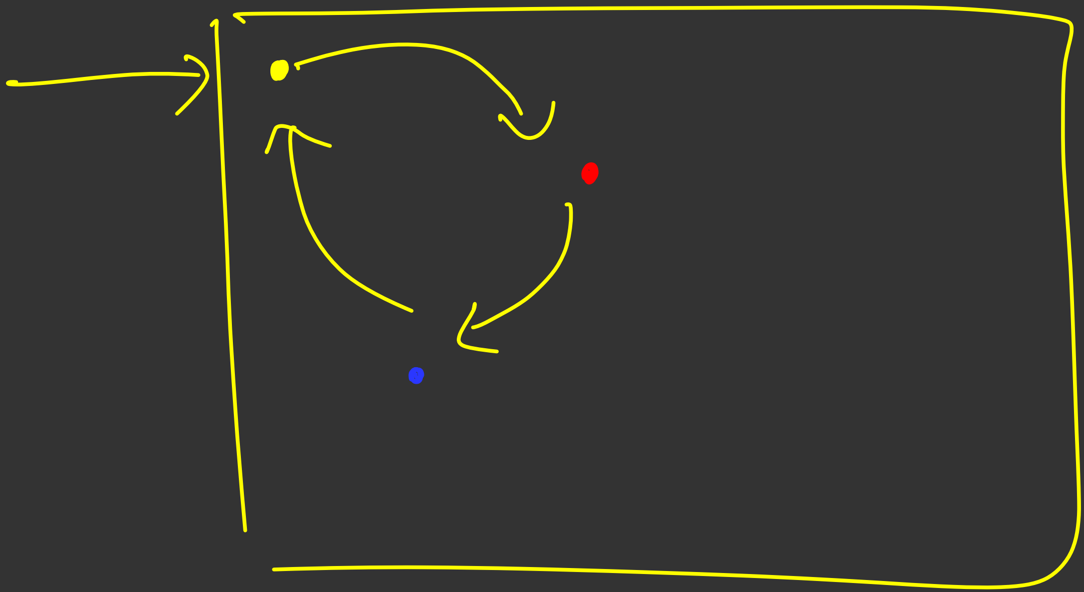
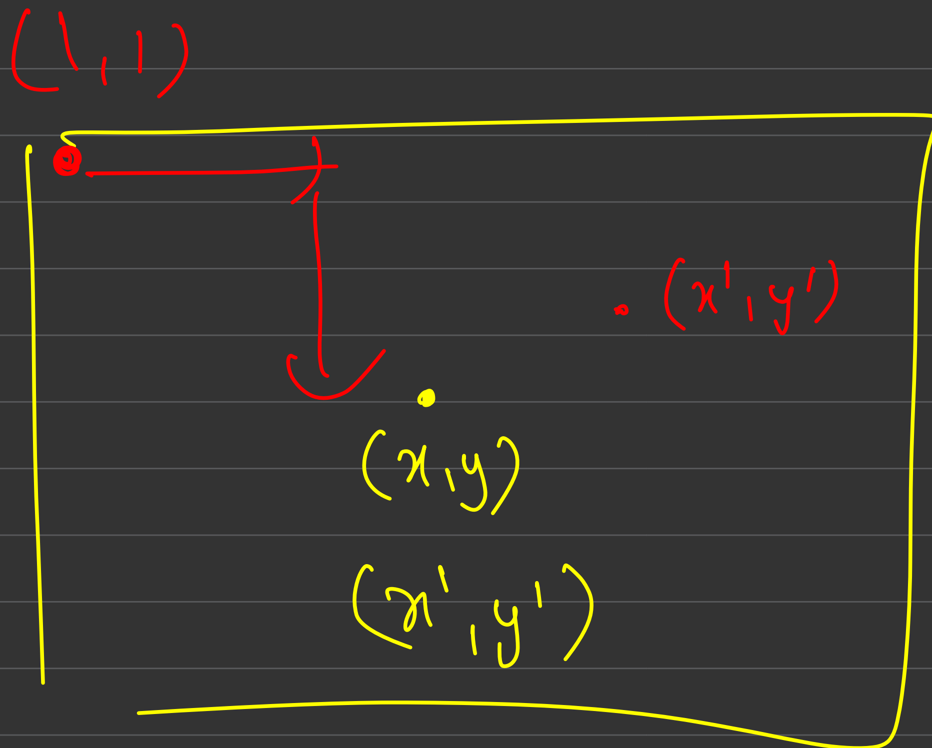
Dynamic Programming
Class 4

- Priyansh Agarwal

Problem 1: Link

- State:
    -
- Transition:
    -
- Base Case:
    -
- Final Subproblem:
    -

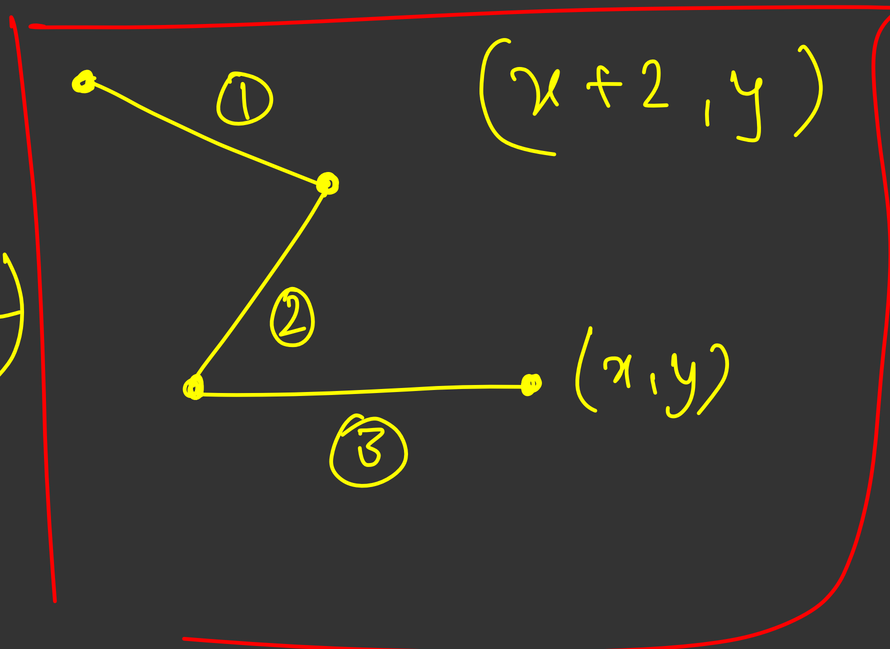




$\{X_i, Y_i \geq 0\}$

$$|x - x'| \leq X_i \;\&\&\; |y - y'| \leq Y_i$$

$(x+2, y+2)$



(x', y')

$$X_4,\ Y_4$$

$$X_4 = 2$$

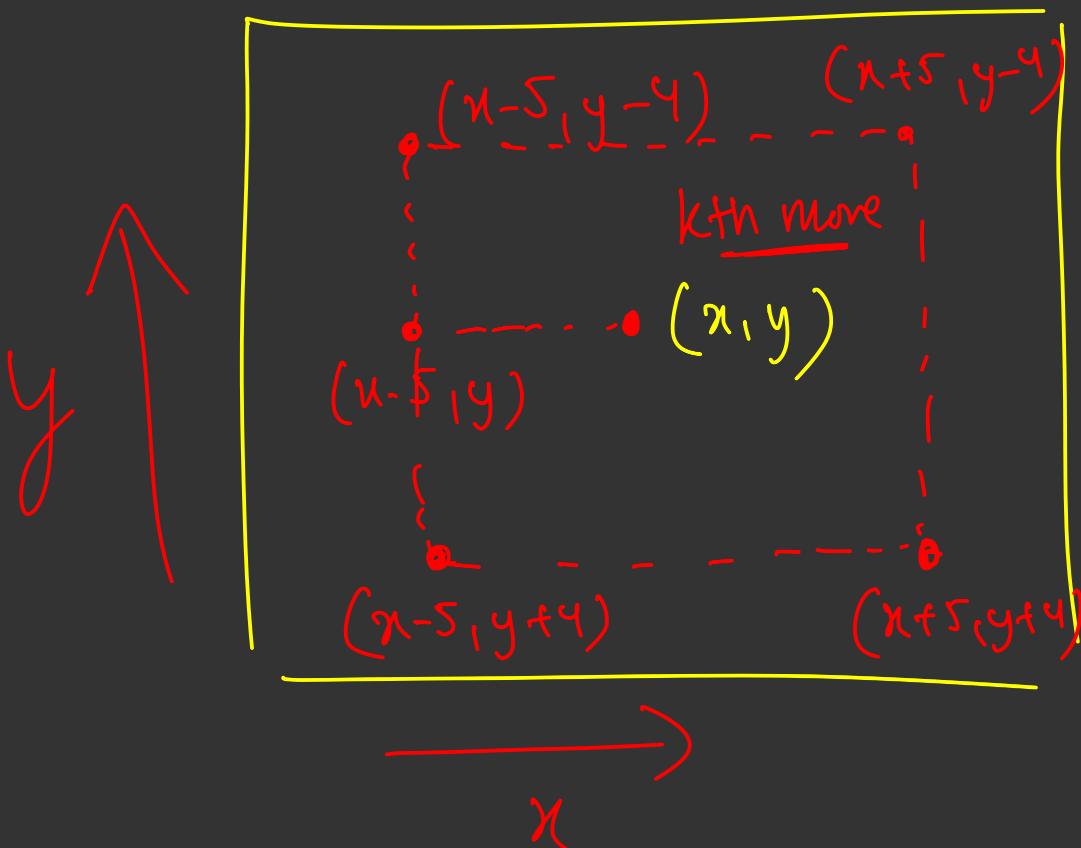
$$Y_4 = 1$$

$$|x - x'| \le 2 \ \&\& \ |y - y'| \le 1$$

$\longrightarrow$ $\underline{\underline{n \le 1000}}$

State : $\longrightarrow$

$dp[k][x][y]$ = # of ways to reach (x, y) in first k moves



$|x - x'| \le 5$

&&

$|y - y'| \le 4$

$X_i = 5$

$Y_i = 4$

$X_k = 2$

$Y_k = 3$

(x, y)

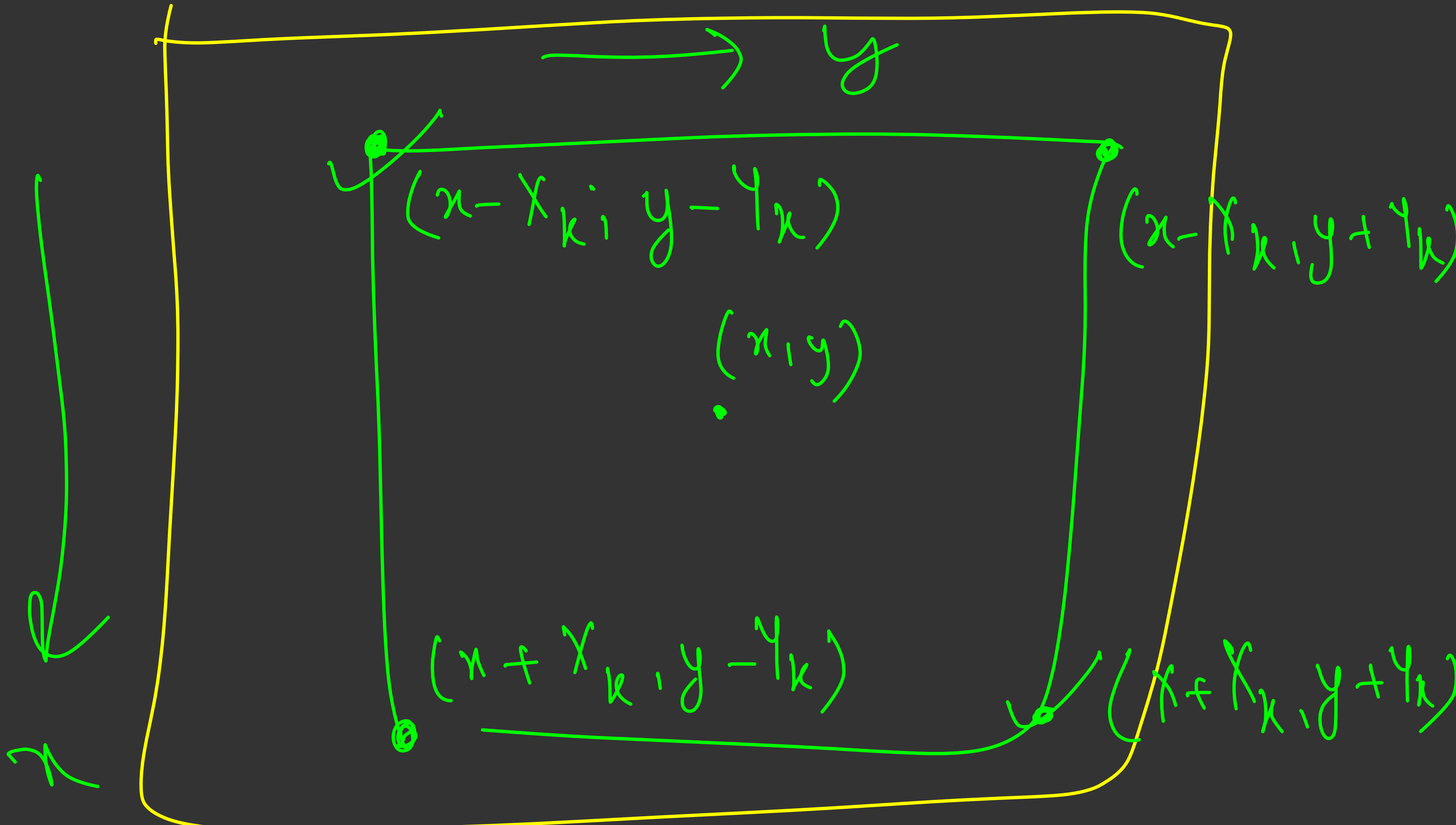
$\left\{ \begin{array}{l} x_k = 3 \\ y_k = 2 \end{array} \right\}$

kth move

$|x_{prev} - x| \leq 3$

ΔL

$|y_{prev} - y| \leq 2$


y
x
(x - X_k, y - Y_k)
(x - X_k, y + Y_k)
(x, y)
(x + X_k, y - Y_k)
(x + X_k, y + Y_k)

$$dp[k][x][y] = \sum_{x', y'} \underline{\underline{dp[k-1][x'][y']}} \Big\} \Big(\underline{\underline{256 \times 256}} \Big)$$

$$|x - x'| \le X_k \quad \&\& \quad |y - y'| \le Y_k$$

$$dp[k][x][y] \longrightarrow \quad \# \quad \underline{\underline{1000 \times 256 \times 256}}$$

$$\left[1000 \cdot (256)^2\right] \cdot \left[(256)^2\right]$$

$$\downarrow \qquad\qquad \downarrow$$

$$\underline{\underline{6 \times 10^7}} \longrightarrow \times 256 \times 256 > \underline{\underline{10^{10}}}$$

State : $dp[k][x][y]$

$=$ # of ways to get to (x,y) in exactly k moves

Transition :

$$dp[k][x][y] = \sum_{x',y'} dp[k-1][x'][y'] \quad (256 \times 256)$$

$|x-x'| \le X_k$ && $|y-y'| \le Y_k$

$2 \times k \times x \times y$

$(2 \times 256 \times 256)$

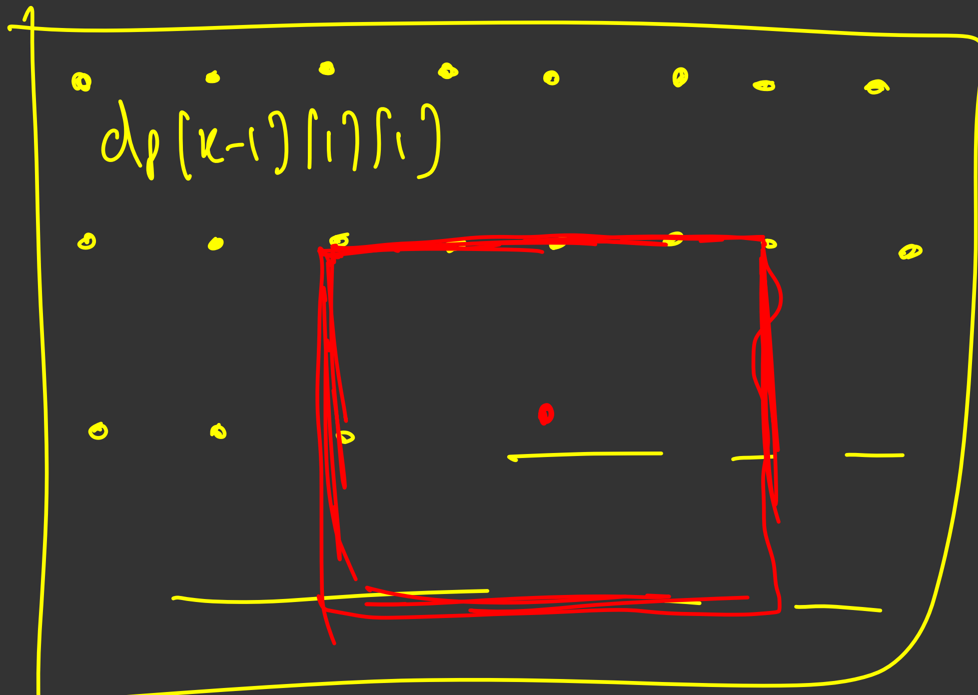
$k = 0$

Base Case : $dp[0][1][1] = 1$

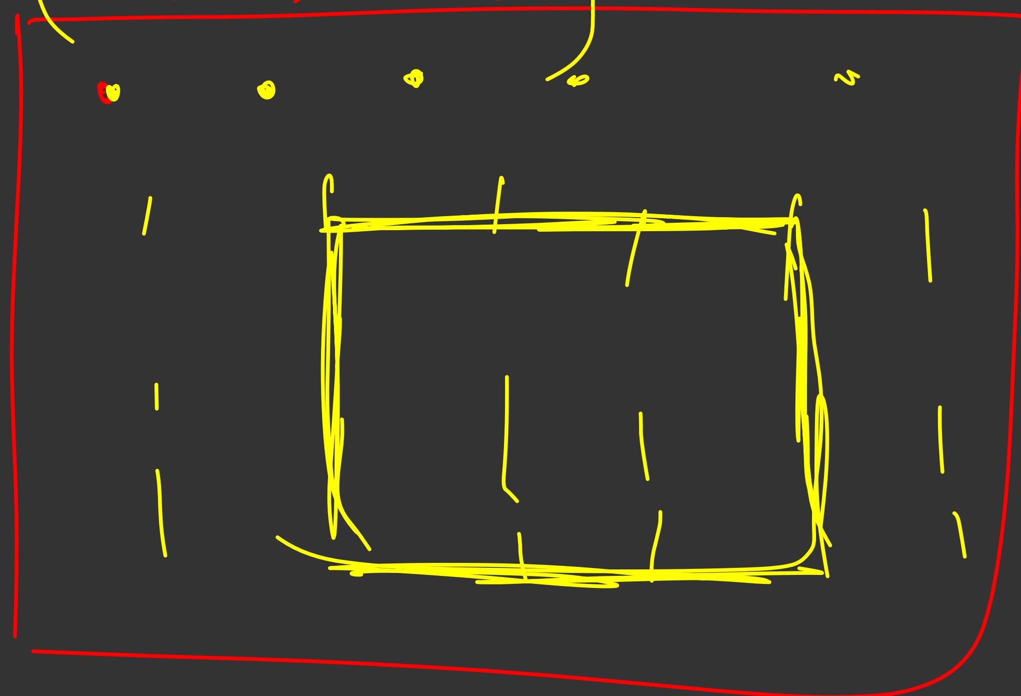
$dp[0][x][y] = 0$

$dp[n][1][1]$

$$dp[k-1][x][y] \longleftrightarrow dp[k][x][y]$$



$\{dp[2][x][y]\} \to$ # ways



$dp[3][x][y]$

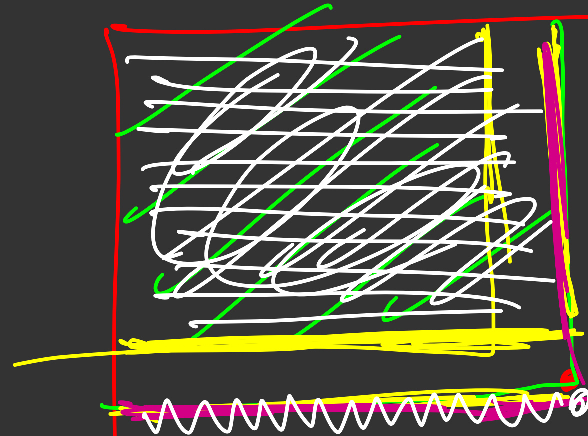
$dp[2][x][y]$

$\{ Sum[2][x][y]$

$= \sum dp[2][x'][y']$

$x' \leq x$

st $y' \leq y$

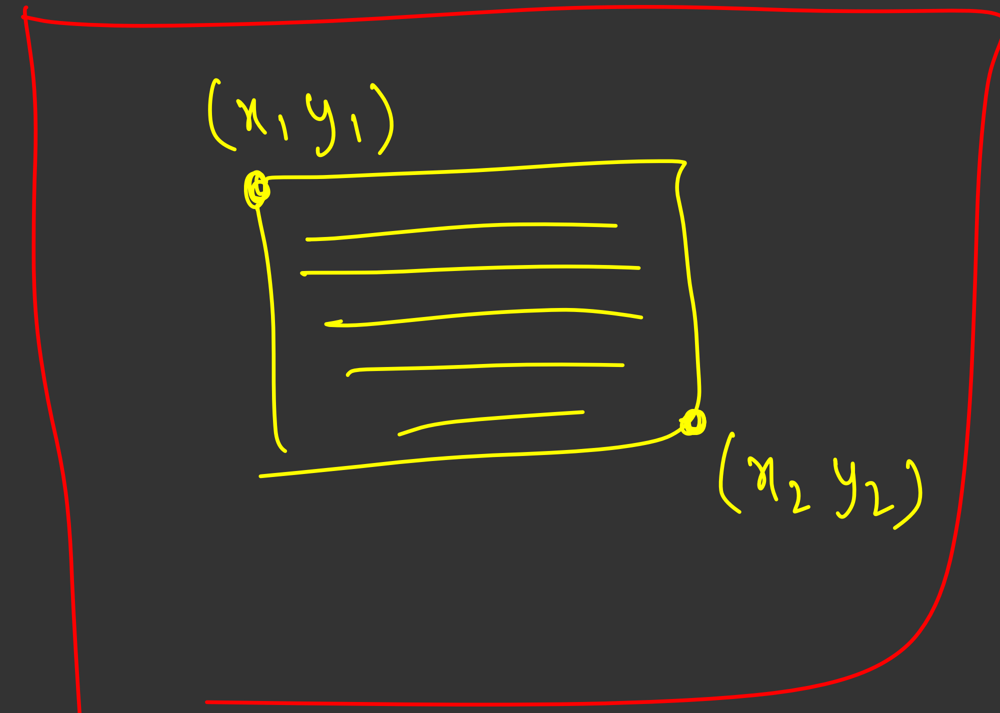


$= Sum2[x-1][y]$

$Sum2[x][y]$

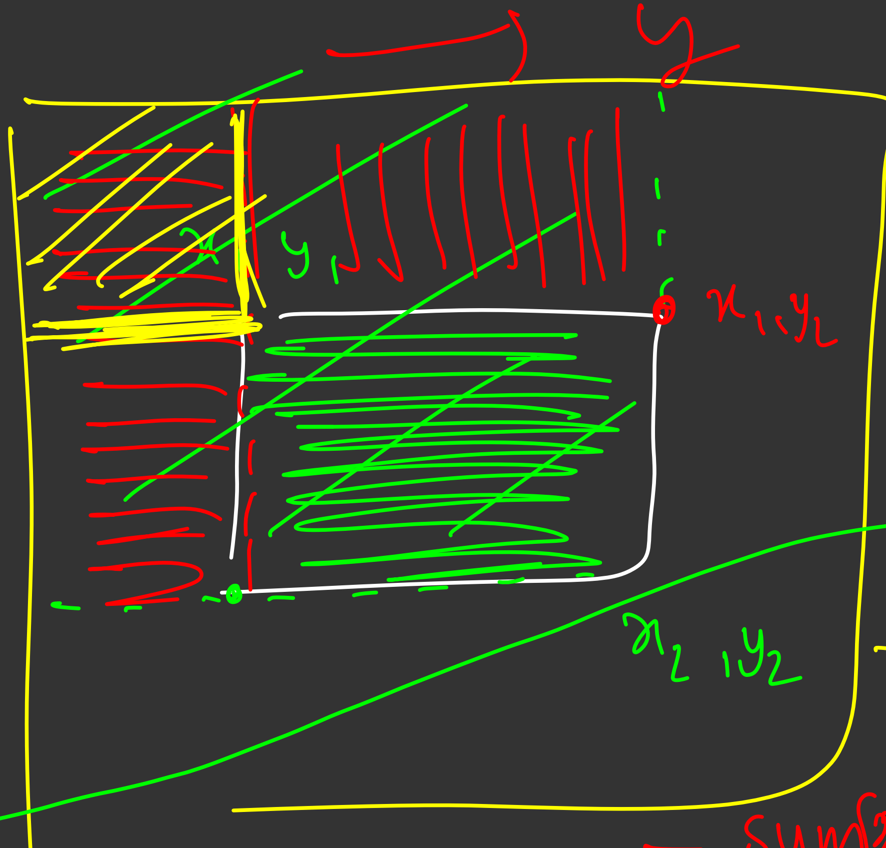
$= Sum2[x-1][y]$

$+ pref[x][y]$



dp[2][x][y] , sum[2][x][y]

$dp[k][n][y] \rightarrow O(1)$



$$Sum[2][x_2][y_2] - sum[2][x_2][y_1] - sum[2][x_1][y_2] + Sum[2][x_1][y_1]$$

$dp[0][1][1] = 1$ $\qquad$ $dp[0][x][y] = 0$

find sum

for (k = 1, k ≤ n, k++)

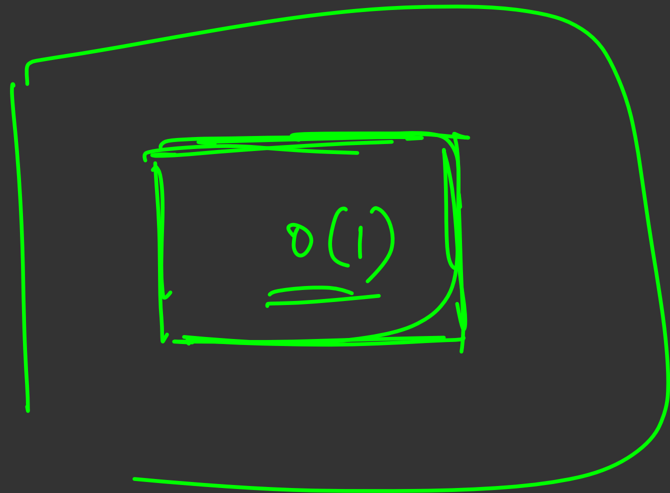
{

for (x = 1 , x ≤ 255 , x++)

for (y = 1 , y ≤ 255 , y++)

$dp[k][x][y] = O(1)$

compute sum matrix $\qquad$ $dp[n][1][1]$


O(1)

Representing non-integer parameters

- How will you store the dp states if instead of integer parameters you had a string or a vector or a map or any complex data type?
- Use a map instead of an array.
- Tradeoff map<pair<int, string>> DP or vector<map<string>> DP

1 2

dp[i][j] where i and j both are ≥ 0 and integer

dp[string x][integer j]

dp[i][j]

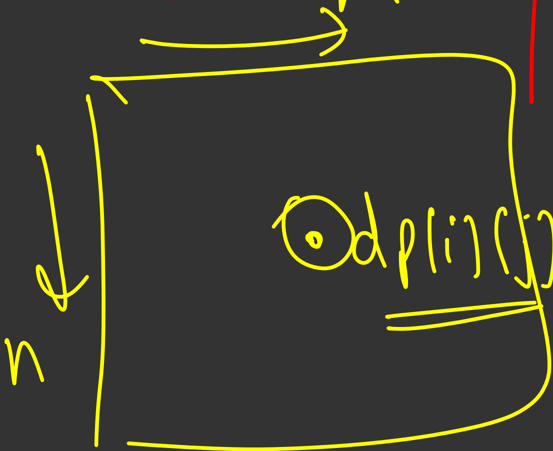
long long ~~int~~ dp[n][m];

string

dp["priyansh"][2]

dp[i][j] → log

~~dp[i][j]~~ → O(1)



i > 0

j > 0

dp[i]

value value

dp[0] dp[1] - - - dp[n-1]

→ key → key

dp("Priyansh")

key - value

→ key, value → string, int

if (dp(x) is not calculated)

calculate it

Array

→ if (dp(x) == default_value)

↳ −1 , −10, −100

Map

→ if (dp.find(x) != dp.end())

$dp[\text{integer}][\text{string}]$

≥ 0

$(\leq 10^5)$ (10^2)

$\text{map}(\text{state} \rightarrow \text{value})$

$\{ \text{map}<\text{pair}<\text{int}, \text{string}>> \ dp \}$

(x, y)

$\downarrow \quad \downarrow$

int string

$\text{map}(\text{pair}<\text{int}, \text{string}> \rightarrow \text{value})$

keys $\rightarrow 10^7$ $\log(1e7) \rightarrow 22$

dp(int) → map(string, value)

map < pair <int, string> > dp1 (x, y)

vector < map <string, value> > dp2

dp1.find({x, y}) == dp1.end() ← log(1e7)

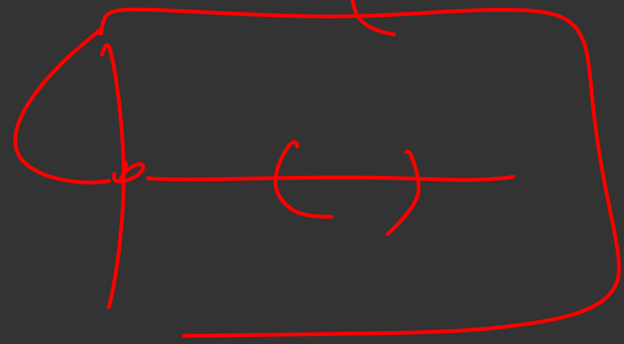
dp2(x).find(y) == dp2(x).end() → log(1e2)

O(1)

$dp(\text{state}) \to \text{key} \qquad (x, y)$

state1, state2 $\qquad ((x, y))$

$dp(\text{state1}) \to [\ dp(\text{state2}) \to \text{value})$



i, j

$dp[x][y] \rightarrow$ string

$\downarrow \quad \downarrow$

$10^5 \quad 2 \cdot 10^2$

map $\rightarrow$ fast

$2.5 \cdot n \rightarrow$ TLE

$1.5 \cdot n \rightarrow$ AC

$2 \cdot 10^7 \cdot 22 = \boxed{4.4 \times 10^8}$

$\# \; 2 \cdot 10^7 \cdot \log(1e7)$

> 2000

$2 \cdot 10^7 \cdot \log(1e2)$

$2 \cdot 10^7 \cdot 7 \; \boxed{1.4 \times 10^8}$

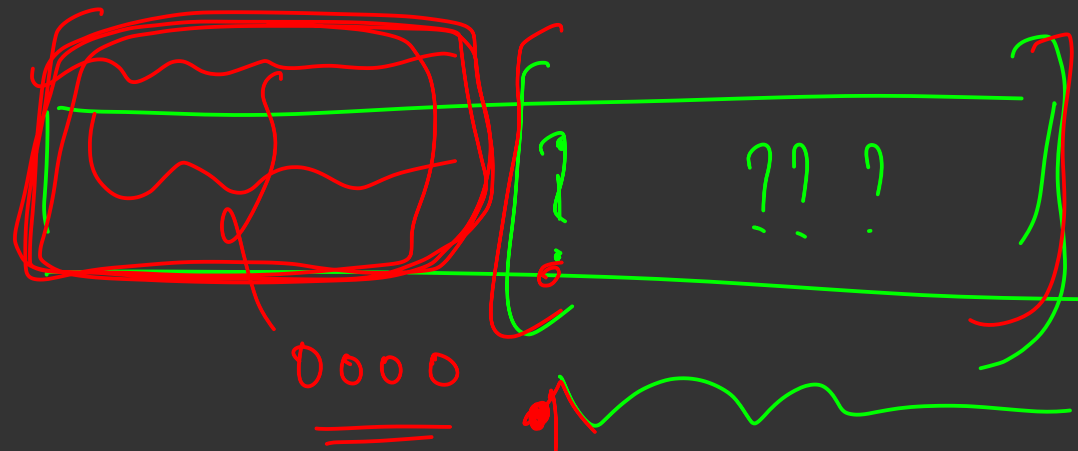
Problem 2: Link

- State:
    -
- Transition:
    -
- Base Case:
    -
- Final Subproblem:
    -

0 1 0 ? ? ? 0 0 1 1 0 1 1 1

$\underline{\underline{k}}$ $\qquad$ 2^k

$dp[i] \rightarrow$ whether we can replace question marks from (i to $n-1$) in a way that there are no palindromes of length 5 or more

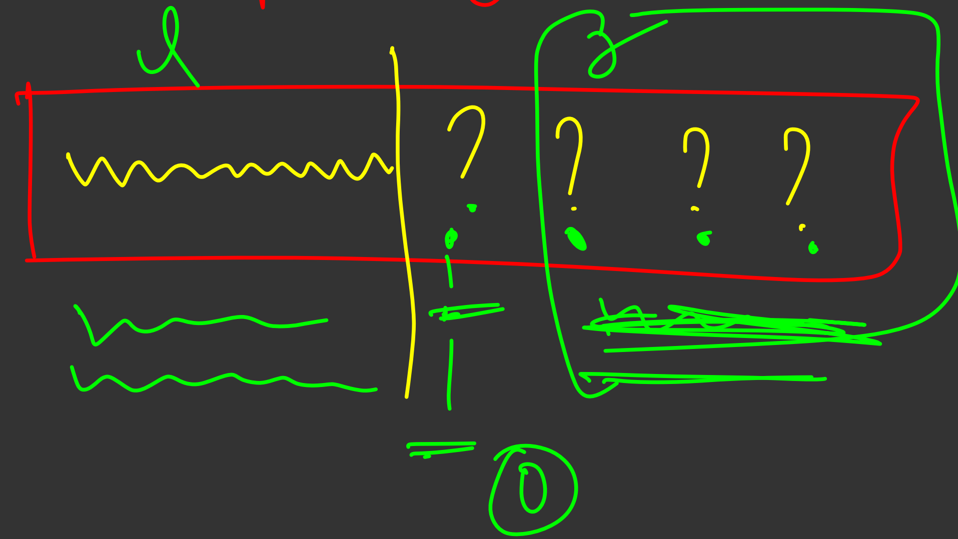


in the entire string

(dp[i][prev string]) — whether or not we can replace all ? from (i to n-1)
(s.t. no palin of len 5 or greater in entire)

string, s: i characters from (0, i-1) are stored in prev string.

prev string



0 1

i

No palindrome of len 5 or greater

5 i-4 i ✗

6 i-5 i ✗

7 i-6 ✓

5
6
7
⋮

dp[i][prev string]

= [dp[i+1][prev_string + "0"] &&

[prev_string + "0" has no palin_ of 5 or greater]]

|| [dp[i+1][prev_string + "1"] &&

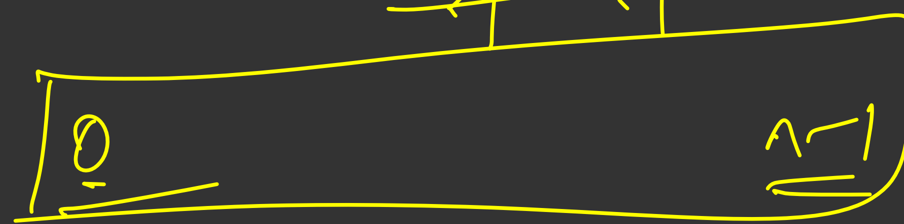
prev_string + "1" has no palin_ of 5 or greater]

0 —— $n-1$

dp[n][any string] = True

Base Case

final subproblem → dp[0][""]



$dp[i][prev\_string] \rightarrow (N \cdot 2^N)$

$5e4$ $\quad 2^{50000} \rightarrow (50000)$

50000

$N \cdot N \cdot 2^N \rightarrow [50000]$

$O(15 \cdot 15 \cdot 2^{15}) = 225 \cdot 1000 \cdot 32$

if string doesn't contain a palindrome of length 5, it will never contain a palin— of leng 5 + 2k

11
6 ——

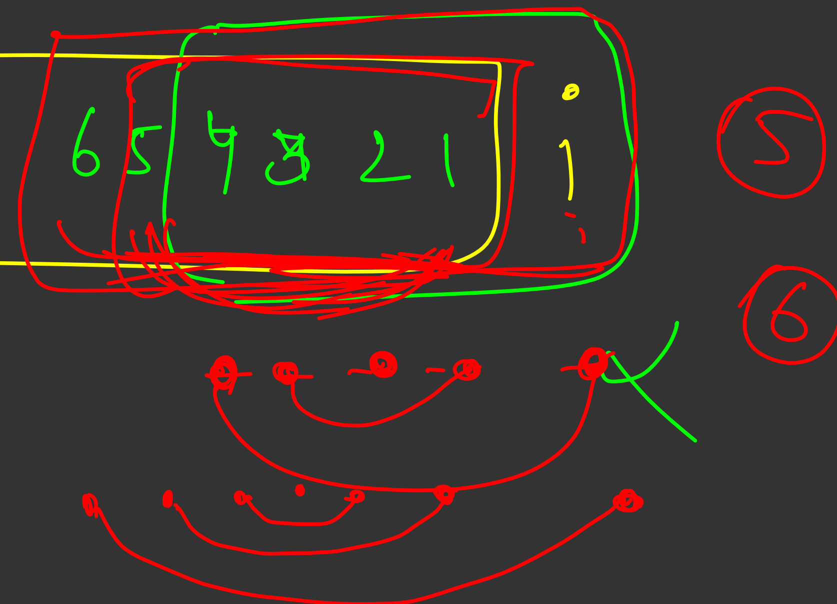
6 + 2k

dp[i][prev_string]

pal_ of length 5 or 6

(6 — i) → palin

(5 — i) → palin



dp[i][last 5 characters]

= { dp[i+1][last 4 character + "8"]

&& (last 5 characters + "0" does not have a pal of 5 or 6)]

|| ["1"] ,)

dp[i][last 5 character] ← 0 1

[0 [0 1 1 0 0]]

5 or 6

$N \cdot 2^5 \cdot O(1)$

$k \le 10$

$5000 \cdot 10 \cdot 32$

DP with Bitmasking

- Bitmasks
- Basic operations on Bitmasks
- Limitations on “N” (You will need 2^N integers to represent all the subsets)

(0 1 0 0 0 0 1 1 0 0 1 1 1 0)

$[1, 2, 3] \longrightarrow$

$\{\}$

$\{1\}$

$\{2\}$

$\{3\}$

$\{1, 2\}$

$\{2, 3\}$

$\{1, 3\}$

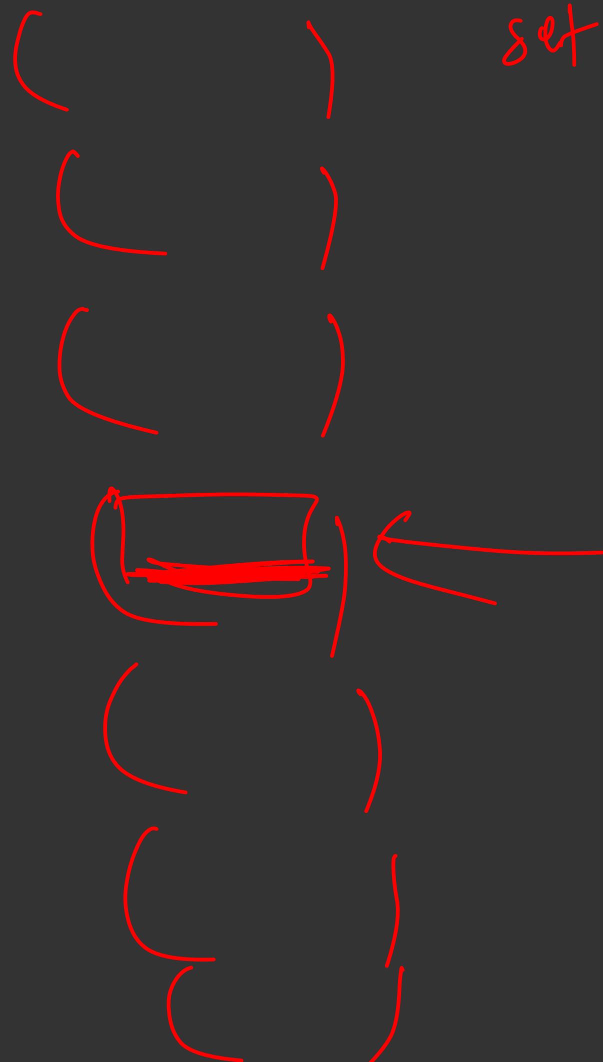
$\{1, 2, 3\}$

N

$2^n \cdot n$

$\{1, 3\} \leftarrow \{2\}$

set



(subset.insert(x))

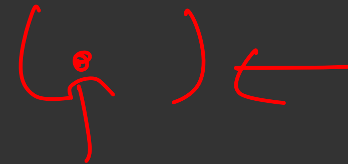
logn

logn

logn

n

	set	vector
insert	logn	1
remove	logn	n
query	logn	n
	O(n)	O(n)



Bitmask $\longrightarrow$ $\{5, 7, 6\} \longrightarrow \underline{\underline{n}}$

(001001)

$0 \to$ ✗

$1 \to$ ✓

$\{\} \to 000 \to 0$

$\{5\} \to 001 \to 1$

$\{7\} \to 010 \to 2$

$\{6\} \to 100 \to 4$

$\{5, 7\} \to 011 \to 3 \longleftarrow \underline{\underline{3}}$

$\{7, 6\} \to 110 \to 6$

$\to \{5, 6\} \to 101 \to 5 \longleftarrow$

$\{5, 7, 6\} \to 111 \to 7$

$\underline{\underline{2^n}}$

	Set	vector	Bitmask
Space	$O(n)$	$O(n)$	$O(1)$
Insert	$O(\log n)$	$O(1)$ pb	$O(1)$
Remove	$O(\log n)$	$O(n)$	$O(1)$
Query	$O(\log n)$	$O(n)$	$O(1)$

single subset

OR

XOR

&

$1 \; (1)(2^2)$

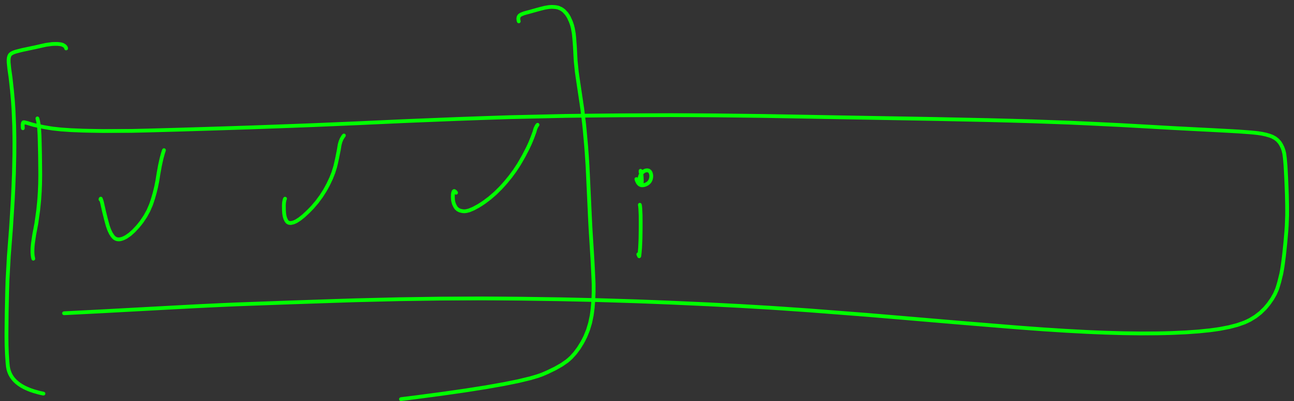
$\{5, 6, 7\}$

$0 \; 0 \; 1 \rightarrow 1$

ch

$(\text{bitmask} \;\&\; (1 \ll i)) \; \cancel{== 0} \rightarrow O(1)$

dp[i][prev_string]



dp[i][subset picked so far]

Problem 1:

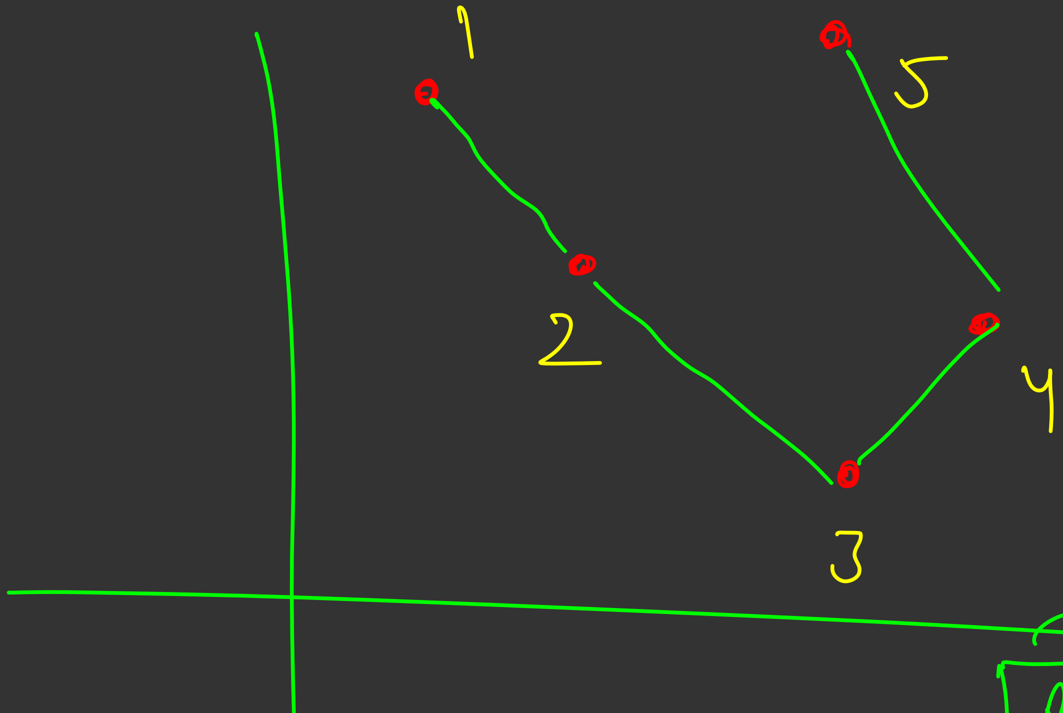
Given a list of points on a 2D plane, rearrange these points in any way such that in the final permutation of points, the sum of distances of the adjacent elements is minimized.

Constraints: [N <= 15], [-1e9 <= Xi, Yi <=1e9]

Points : [{0, 0}, {5, 6}, {1, 2}]

Best permutation -> [{0, 0}, {1, 2}, {5, 6}]]

Ans = Dist(P1, P3) + Dist(P3, P2)

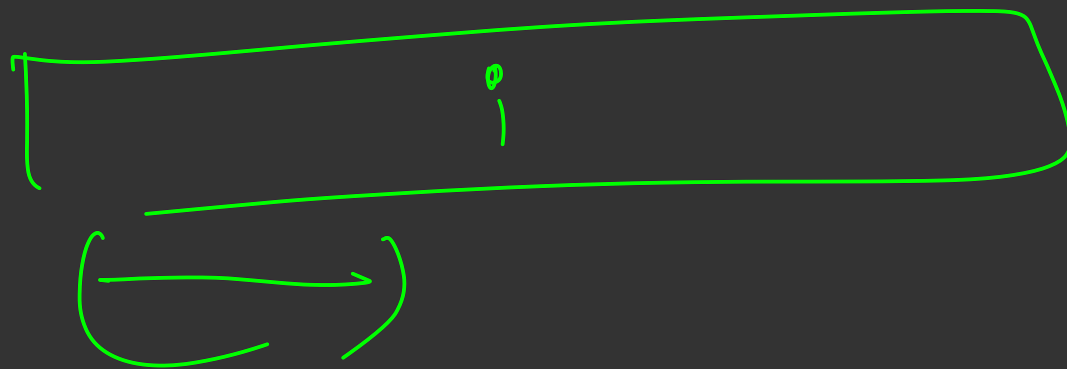


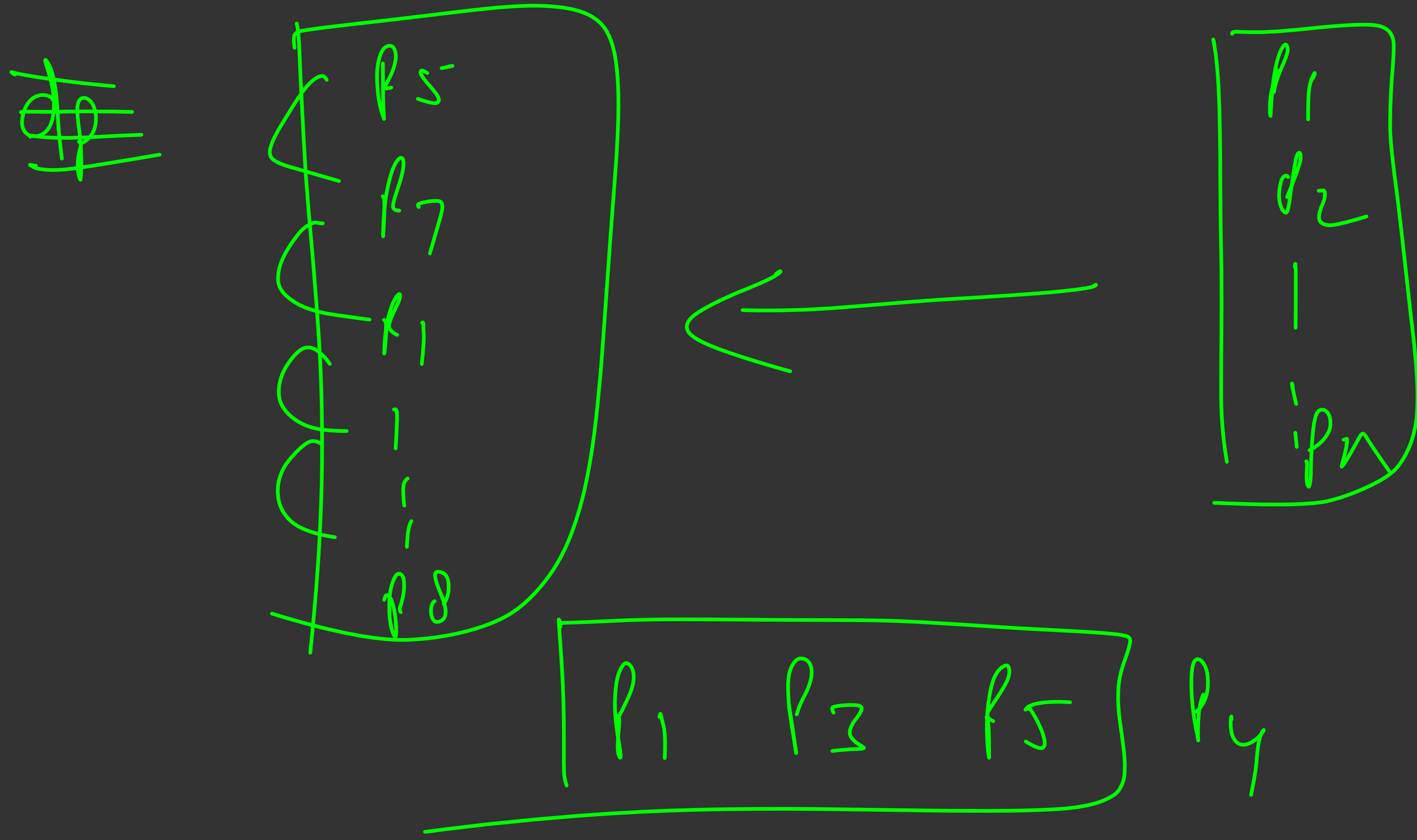
{sum of $d(i,j)$ is minimized}

$\boxed{p_1 \; p_2 \; — \; p_n}$

$(n!)(n)$

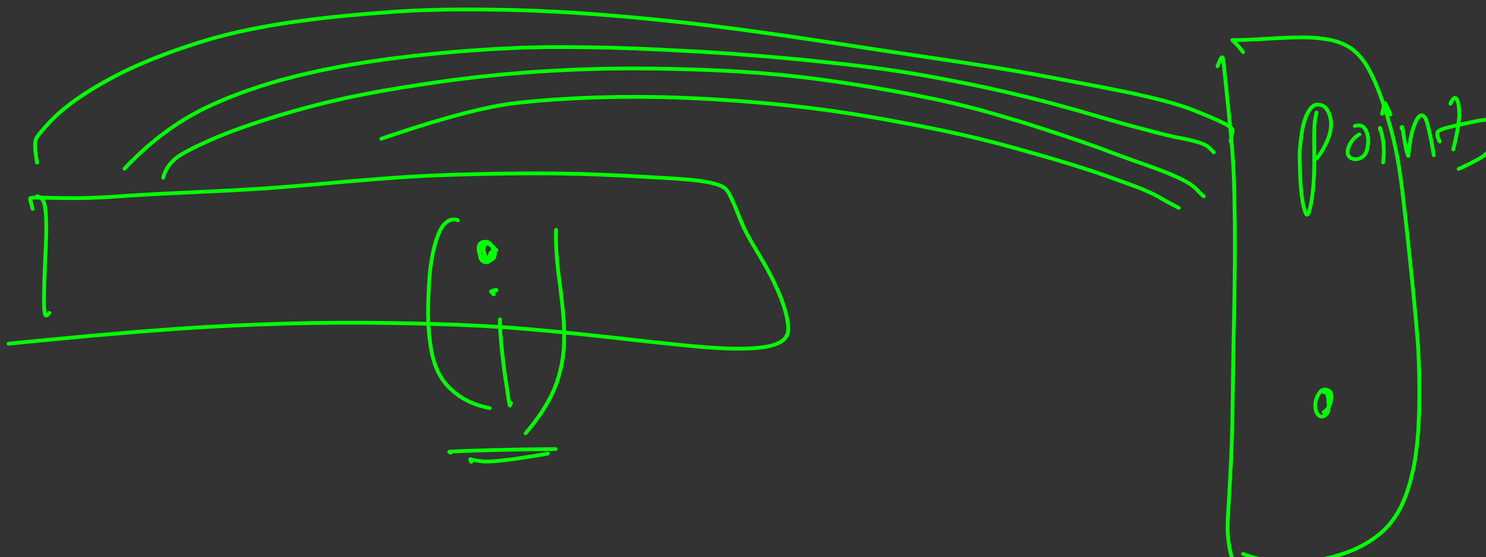
$(15!)(15)$




p5
p7
p1
p8
p1
p2
pn
p1 p3 p5
p4

[illegible] ith element

some i-1 points already ith point



dp[i][mask]

(1 2 3)(4)

(2 3 1)(4)

(3 2 1)(4)

3

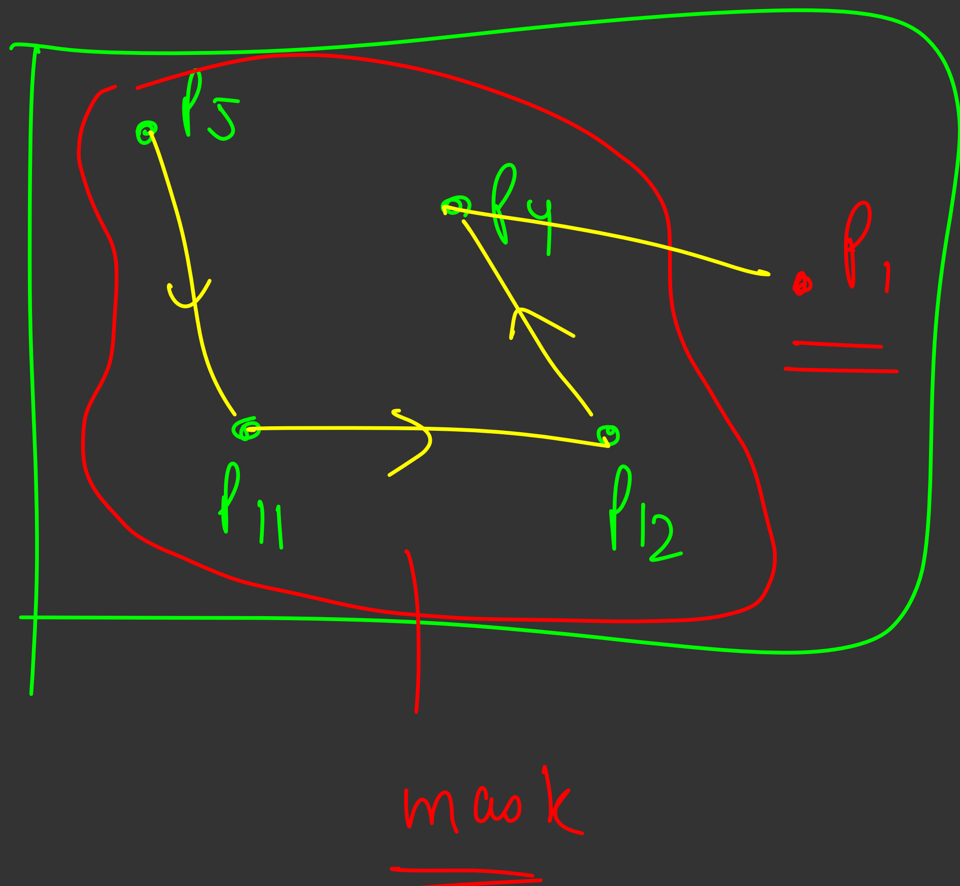
2

1

i

0 0 0 1 0 0 0 0 0 1

$\le 2^n$



p5
p4
p1
p11
p12
mask

$dp[i][mask][last]$ — $[i \quad n-1]$

mask

= min sum of distances while picking up points on locations $[i$ —— $n-1]$ s.t mask contains already picked up points and last is the index of the last picked point.

$j=0$
$j=1$
$\vdots$
$j=n-1$

point

$$dp[i][mask][last] = \min_{\substack{\text{all } j \text{ st} \\ (mask \,\&\, (1<<j)) == 0}} \left\{ dp[i+1][mask \mid (1<<j)][j] + \text{dist}(point[j], point[last]) \right\}$$

mask has atleast 1 element

~~Final~~ Base Case

dp[n][complete mask][any last] = 0

Final subproblem

dp[0][0][-1]

Problem ~~2~~: ~~Link~~

- State
  - dp[i][mask][last] = min sum from [i to n-1], mask = picked up points, last = last picked point
- Transition
  - dp[i][mask][last] = min { dp[i+1][mask | (1<<j)][j] + d(j, last) }
    j, st (mask & (1<<j)) = 0
    (i>0)
- Base Case
  - dp[n][2^n - 1][x] = 0
- Final Subproblem
  - dp[0][0][0] ← min { dp[1][00001][0], dp[1][00010][1] }
    dp[1][00100][i]

State of

dp[i][mask][last] $\rightarrow$ O(n)

$n \quad 2^n \quad n$

Time Space

Both

$n^3 \cdot 2^n$

{ i == set_bits in mask }

$\boxed{\text{dp[mask][last]}}$ $\rightarrow$ $O(n^2 \cdot 2^n)$

n—15

Problem 2: Link

- State
    -
- Transition
    -
- Base Case
    -
- Final Subproblem
    -

Problem 2: Link

- State
  -
- Transition
  -
- Base Case
  -
- Final Subproblem
  -