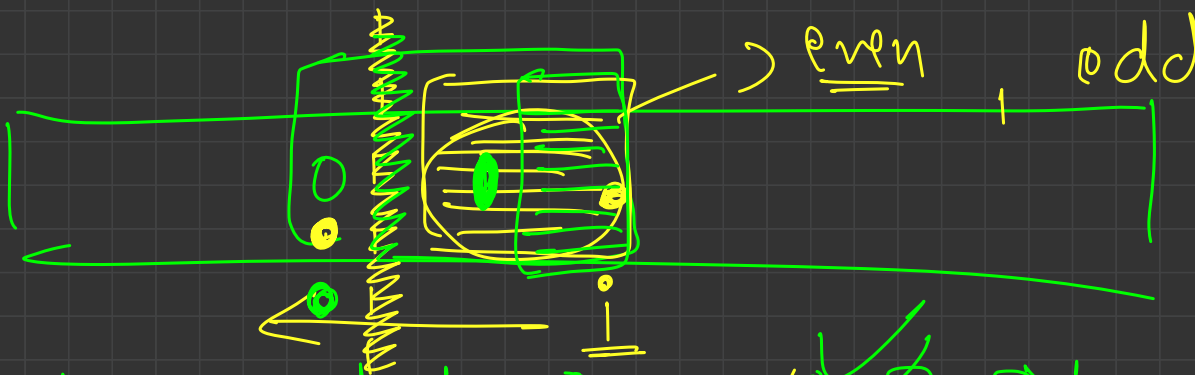
Dynamic Programming 2.3

- Priyansh Agarwal



1 -> has product = 0 ✓

2 -> has product -> +ve ✓

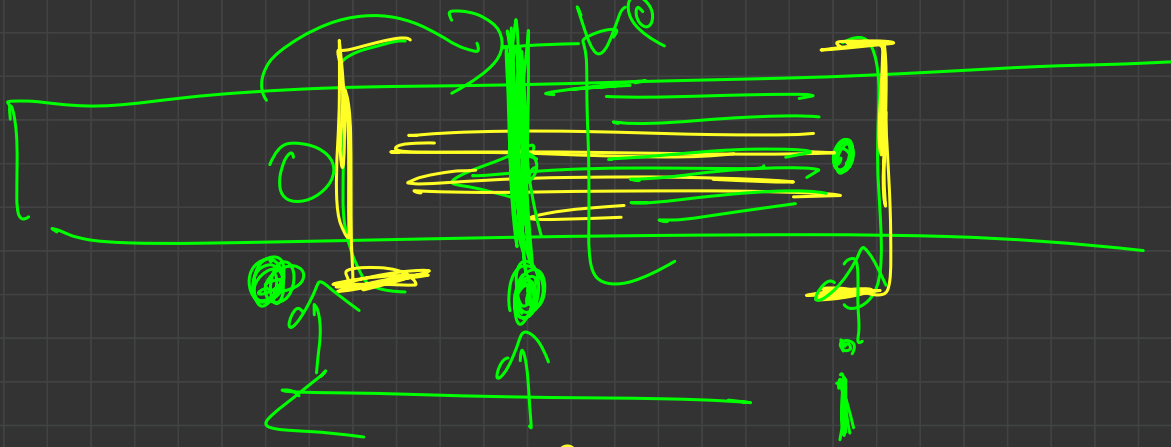
3 -> has product -> -ve ✓

Yes / No

-16

-2

-1 -1 -1 -1



O(n)

O(n)

even → +ve

O(n) → 0

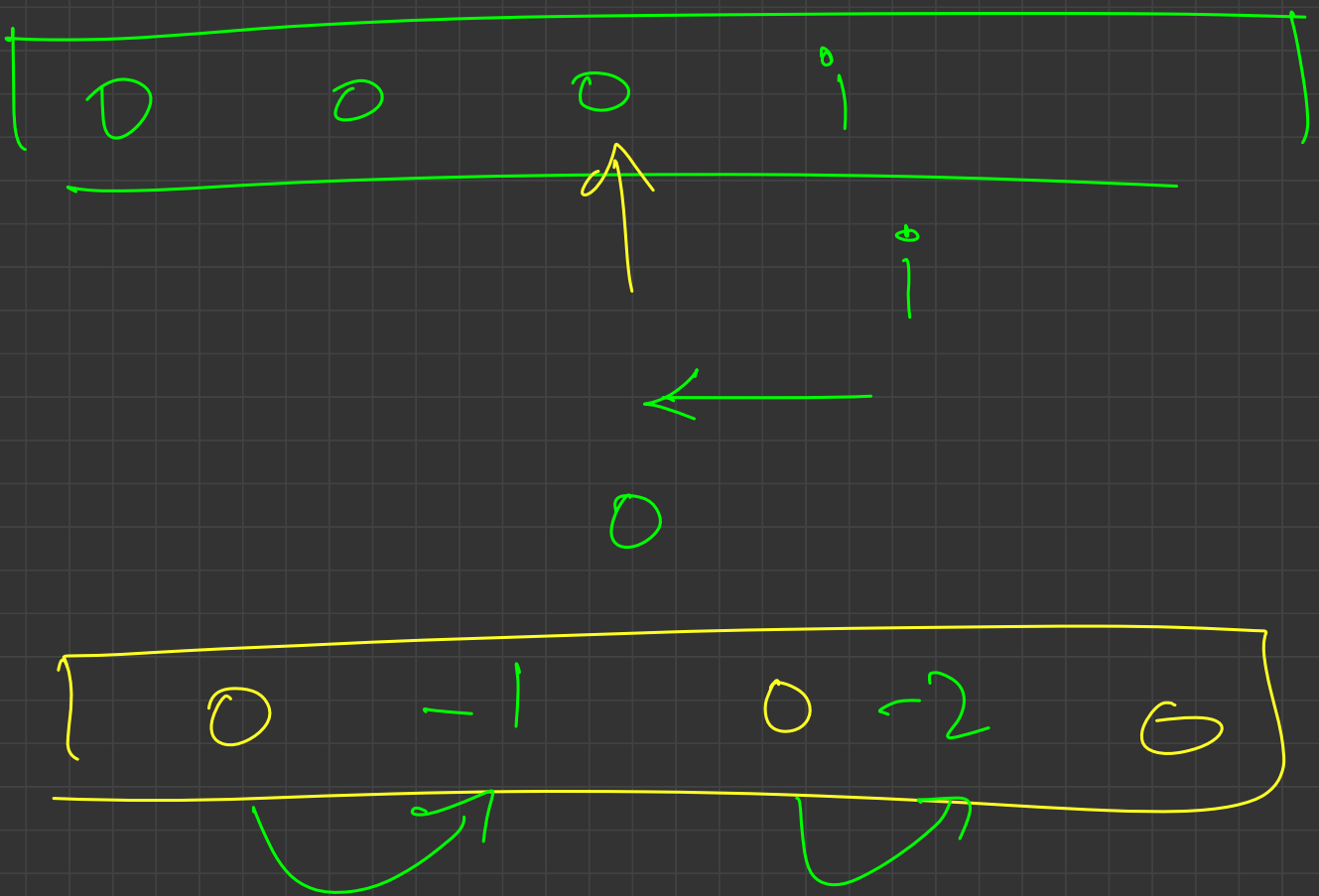
O(n) → even

O(n) → even

even → -ve → odd

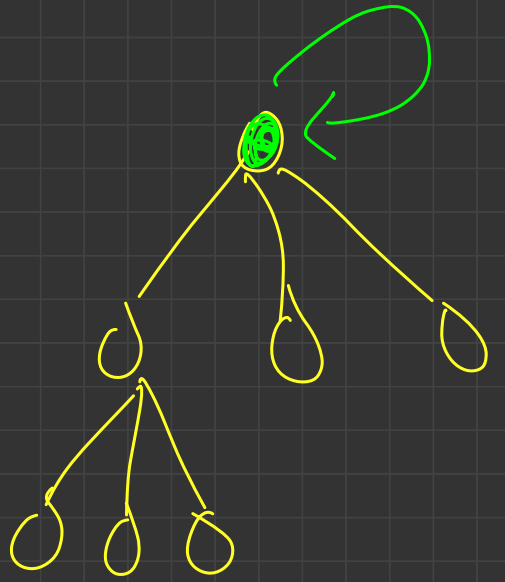
O(n)

O(n)


0 0 0 i
i
0
0 -1 0 -2 0

① Representing Non-Integer DP parameters

② Cycling DP states



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

dp[i][j][k] → (i, j, k) are all +ve integers

dp["Priyanth"][0][2]

$dp[1], dp[2] \quad --- \quad dp[n]$

$dp["Priyanshi"] \quad dp["Jain"]$

Memoization :

→ states

- already calculated
- to be calculated

Calculate & then store

Array

$\int O(1)$

to access it very fast

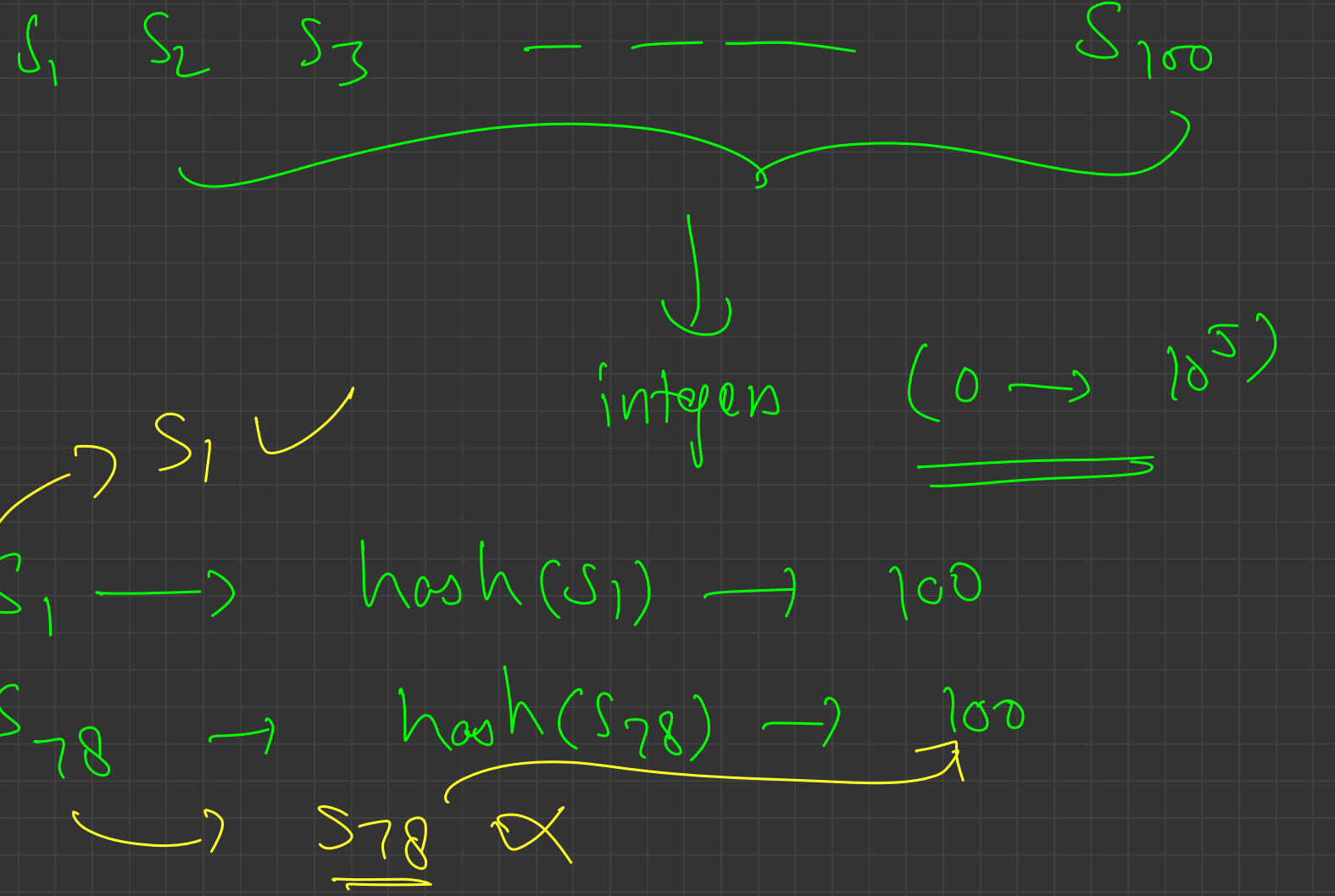
$O(1)$

dp[1], dp[2] — — — dp[n]

Access time → O(1)

Update | Store time → O(1) ≤ 10^7

dp["priyansh"] → hash("priyansh") into an integer ≤ 10^5


s_1 s_2 s_3 — — — s_{100}
integer
(0 → 100)
s_1
s_1 → hash(s_1) → 100
s_{78} → hash(s_{78}) → 100
s_{78}

hash a string and convert to integer and then use normal dp

Access → O(1)

Update → O(1)

No ✗

map < key, value >

Access time
insert / update } $\geq \log n$

integer, string, vector, map, set

, double value

Hash map

Hash map

Tree map

Hash map

keys are in a random order

Update
Insert
Delete
Search

Avg time → O(1)

Worst time → O(n)

Normal / Tree Map

keys are sorted

Avg → O(log n)

Worst → O(log n)

O(1)

Average

Access / Insert / Delete / Update time

O(log n)

dp["Tie"] → int dp(N)

&

map<string, int> dp

def (string)

vector

map

set

double

map

dp[integer][string] dp[2]["TLE"]

① dp[{2, "TLE"}]

map<pair<int, string>, int> dp

② dp[2]["TLE"]

vector<map<string, int>> dp

dp[1][2]["priyanshi"]

10^2 10^2 10^4

$\log(10^8)$

①

Map < pair < pair < int, int >, string >, int > dp

②

vector < vector < map < string, int > >> dp

→ $\log(10^4)$

map < pair < int , string > , int > dp

↓ total combinations

20 iterations → 10^6

total keys in map → 10^6 → $\log(10^6) \rightarrow 20$

dp [int] [string]

10^3 integer 10^3 string

vectr < map < string , int > dp

10^3

integer

dp[2]["TLE"]

1 : 10

10 operating

dp[2] → map < string, int >

O(1)

"TLE"

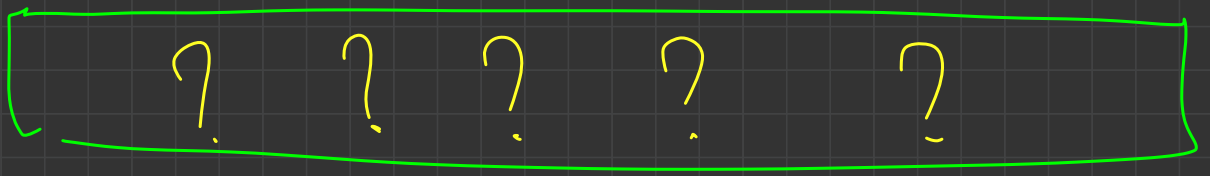
10^3 string

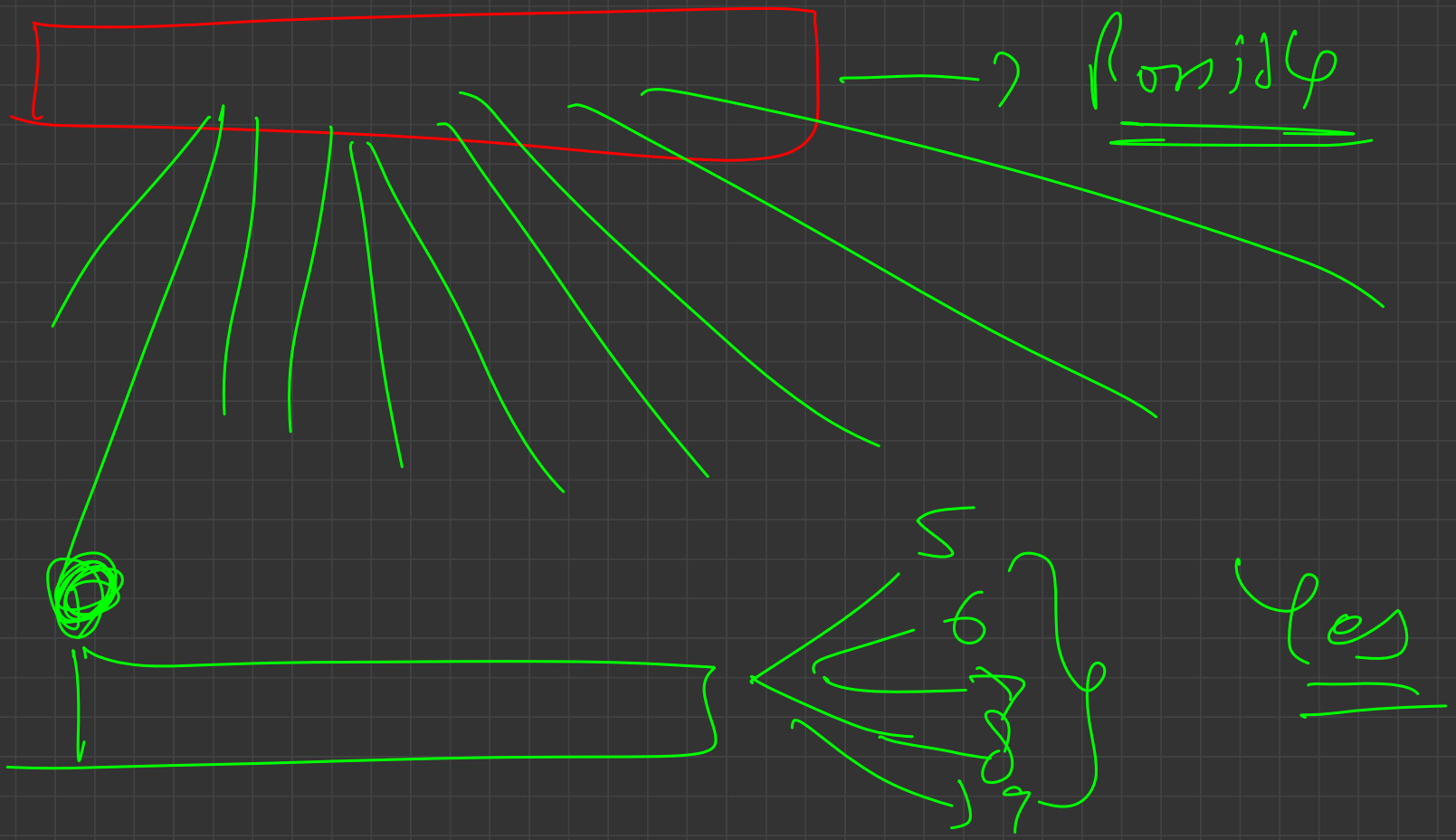
$O(\log(10^3)) \rightarrow 10$

Problem: Link

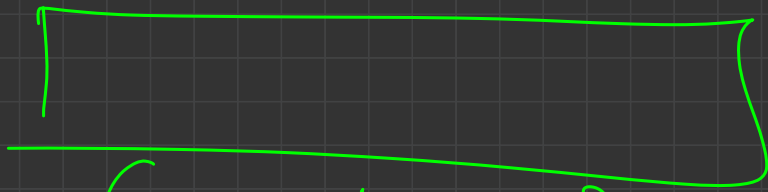
- State:
  -
- Transition:
  -
- Base Case:
  -
- Final Subproblem:
  -

string of length n




Ramidh
Yes

N →



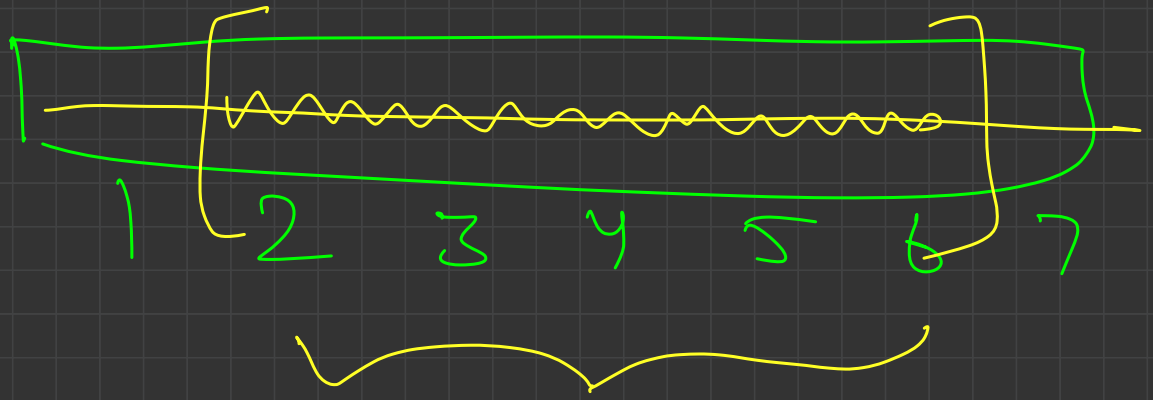
(5, 6, 7, 8, ---- n)

$O(n^2)$ strings . $\Theta(n)$

$O(2^n \cdot n^2 \cdot n)$ → $n \to 15$

generation substring palindrome check

Palindrome of length → 7



① if a string doesn't have a palindrome of leng 5 then it won't have a palindrome of length 7, 9, 11, 13 ....

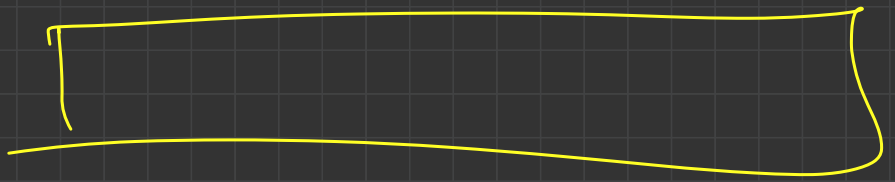
② Similarly for length 6

for a string :

5, 6, 7, 8, 9, 10

---: n

5, 6, 7



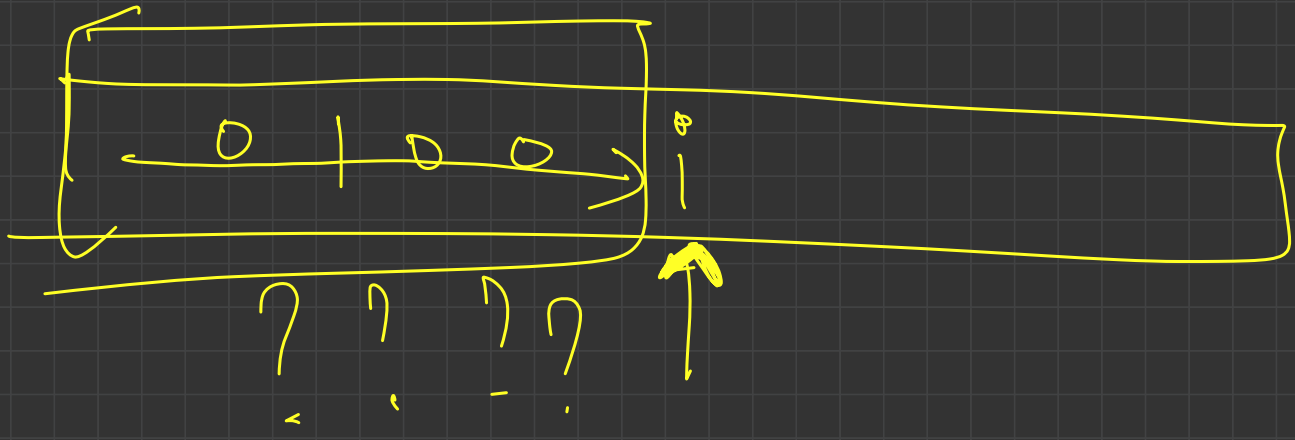
$2^n \cdot n^3$

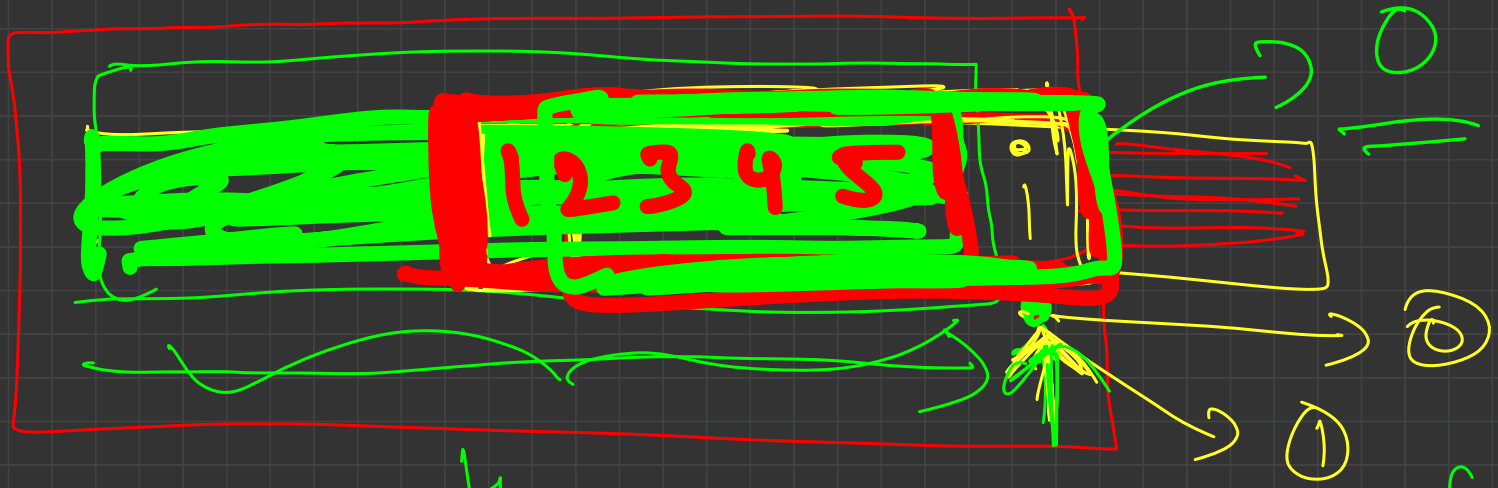
$2^n \cdot n^2$

5 | 6

$O(n)$ $O(n)$

$O(n) \cdot O(n)$


0 1 0 0
i

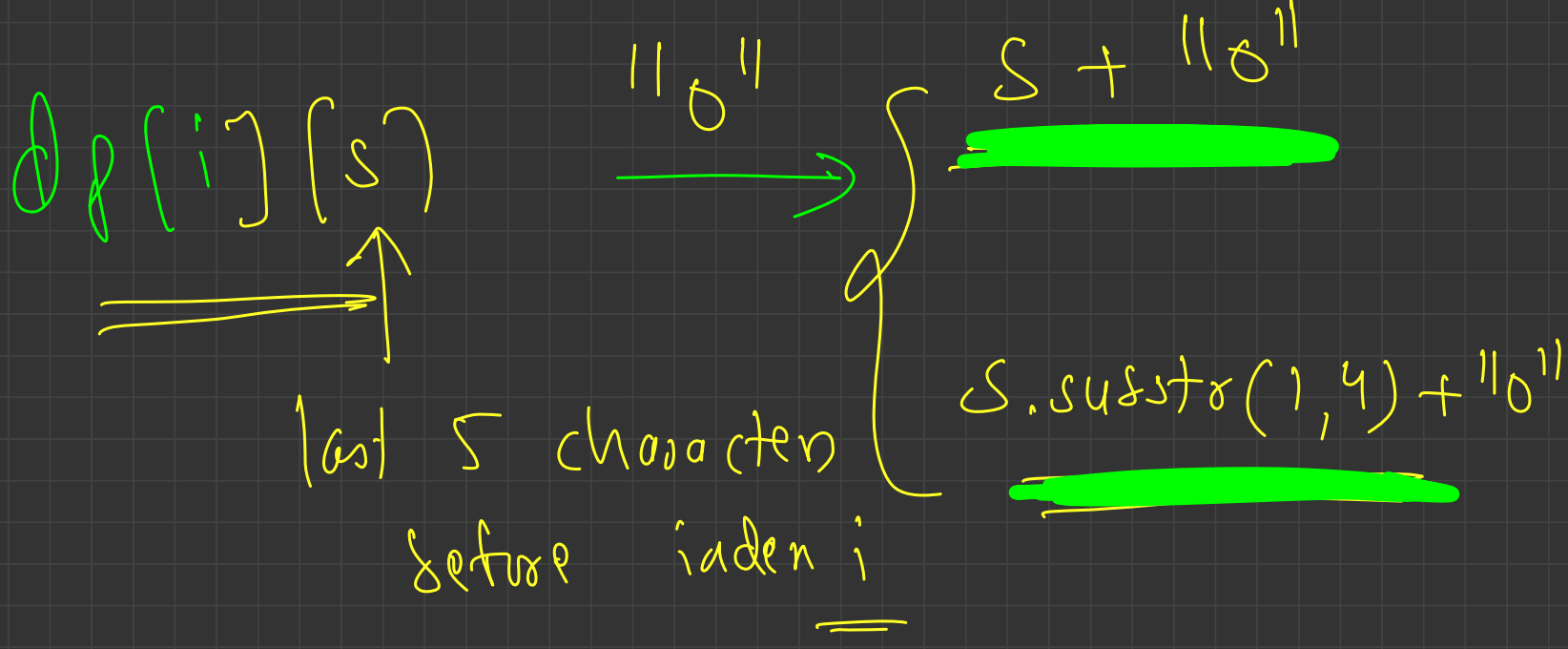


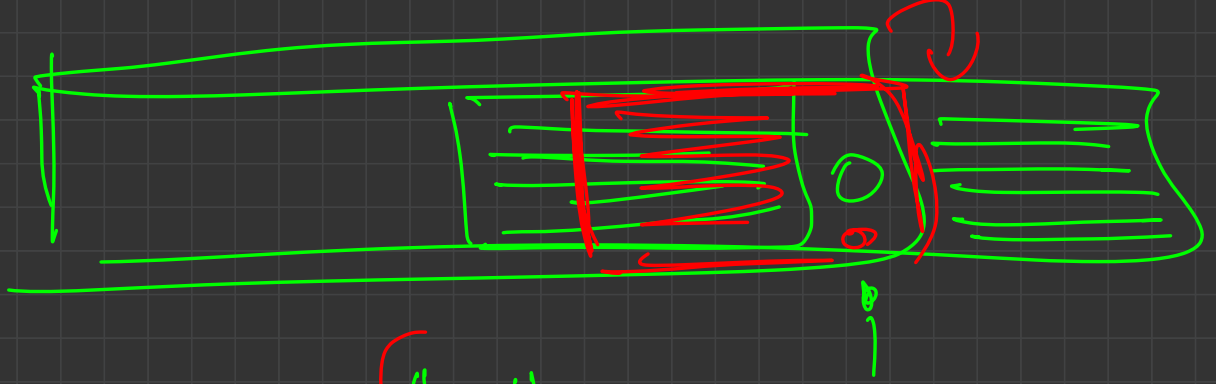
there are no substring of length 5 or greater which are palindromes

dp[index][" last 5 characters"]

—) I have filled up the string till [index − 1] such that the last 5 characters are ↓, is there a way to replace question marks from (index) to (n-1) such that no

Palindrome of length 5 or greater exist





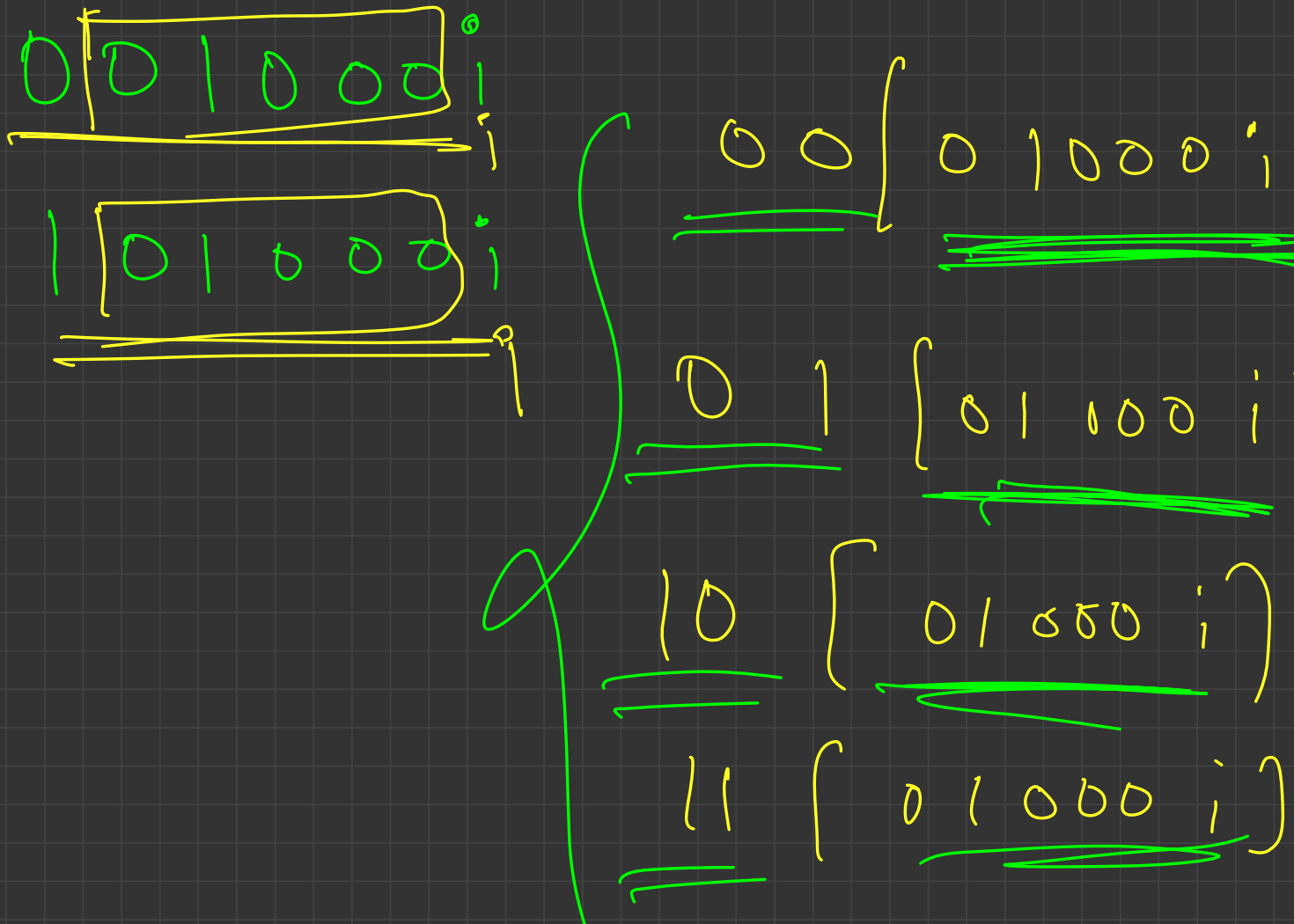
dp[i][s] { "0" → check for palindromes } dp[i+1][s.substr(1,4) + "0"]

dp[i][s]

true

dp[i+1][s.substr(1,4)+"0"] dp[i+1][s.substr(1,4)+"1"]

```
bool possible ( int index, last, s)
    if (index == n)
            return true
    if (dp[index][last] != -1)
            return true
    put 0 ---> check for palindrome
        possible( index+1, last.substr(1,4)
                          + s[index], s )
```


0 0 1 0 0 0 i
1 0 1 0 0 0 i
00 0 1000 i
01 100 i
10 1000 i
11 0 1000 i

Cycling DP states

- What happens when your current state is dependent on itself?
  - dp[i] depending on dp[i] itself

Problem 1

- Given a positive integer N <= 1e6, at every step the following 3 things can happen to N with equal probability.
  - N = N / 2
  - N = N - 1
  - N remains unchanged
- Find expected number of steps it will take to convert for N to become 0

Problem 1

```
state:
dp[n] = expected number of steps to convert n to 0


transition:

dp[n] = 1/3 (1 + dp[n - 1] + 1 + dp[n / 2] + 1 + dp[n])

dp[n] = 1/3 (3 + dp[n - 1] + dp[n / 2] + dp[n])

dp[n] = 1 + dp[n - 1] / 3 + dp[n / 2] / 3 + dp[n] / 3

2/3 * dp[n] = 1 + dp[n - 1] / 3 + dp[n / 2] / 3

dp[n] = 3 / 2 + (dp[n - 1] + dp[n / 2]) / 2

base case:
dp[0] = 0

final subproblem = dp[n]
```

Problem 2: Link