14. Dynamic Programming - 2

27 July 2024 14:01

Q/. n = 3 [1, 2, 3] → infinite

value = 4

1+1+1+1 → 4
1+1+2 → 4
3+1 → 4
2+2 → 4

4

int dp[n+1][val+1];
-1

(n, val)
→ (n-1, val)
→ (n, val - coins[n-1]) $\boxed{<0}$

val = 10 → - - - - → 0
val>0, n=0 → 0

val < 0 → 0
val = 0 → 1
val > 0, n=0 → 0
val > 0, n > 0

Q/. ⟶ ⟵

abcddcba → even
abcdcba → odd

String → I/P
↓
min no. of char to be inserted to convert it to a palindrome.

to convert it to a palindrome.

abca

abcba

acbca

longest palindromic subsequence?

a b c d d c b a

→ abca
→ acba] → LCS

(aba, aca)

3

(orig, rev) → LCS → LPS

n − LPS = smallest

4 − 3 = 1

abcdcba

abcd
dcba] → LCS = 1

n = 4

n = 4 − 1 = 3

→ abacdab
→ badcaba

badab

abacdab a → abacdcaba

2

Q. S = "1234" → 10^5

modulo 10^9+7

7 → 1234567

1 + 2 + 3 + 4 + 12 + 23 + 34 + 123 + 234 + 1234.

= 1670.

1 → 1111

$1 \rightarrow 1111$

$2 \rightarrow 2+2+20+20+200+200 \rightarrow 444$

$3 \rightarrow 3+3+30+3+30+30 \rightarrow 99$

$4 \rightarrow 16$

$$\begin{array}{r} 444 \\ 99 \\ 16 \\ \hline 1670 \end{array}$$

12   0 $n=4$ $n=4$

2   $1+1\times10+1\times100+1\times1000 \rightarrow 1111$

2

n , i , K

$\boxed{i+1}$

4567
34567
1234567
234567

1234567

$(i+1)(K)+(i+1)(K)(10)$
$+(i+1)(K)(100)$
$+(i+1)(K)(1000)$

45
345
2345
12345

4
34
234
1234

$(i+1)(K)(1+10+100+1000) \rightarrow$ GP $(i+1)(5)(111)$

$(i+1)(K)(1111)$ $(i+1)(3)(11111)$ $\frac{a^n-1}{r-1}$

$\frac{a(r^n-1)}{r-1}$

$\frac{1(10^4-1)}{9}$

$= \frac{9999}{9} = 1111$

0 1 2 3 $n=4$ 1111

(1234)

0

$(n-i)$ times)

$$\sum \boxed{(i+1)(k)(1111\_\_\ (n-i)\ \text{times})}$$

$(1)(1)(1111) + (2)(2)(111) + (3)(3)(11) + (4)(4)(1)$

$\rightarrow 1111 + 444 + 99 + 16 \rightarrow 1670.$

KMP Algo :-

Q/. txt = abcdefghi

pat = cde → YES

pat = cdf → False

txt → n

pat → m

pat = cde

abcdcdefghi → true

c d e

i → i+m-1

$O((n-m+1)(m))$

$O(n \cdot m)$

⇒ KMP Algo :-

abcd { a, ab, abc, abcdx

... which is

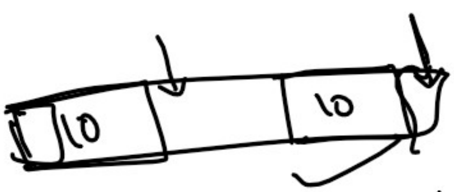
lps → longest proper prefix which is also a suffix.

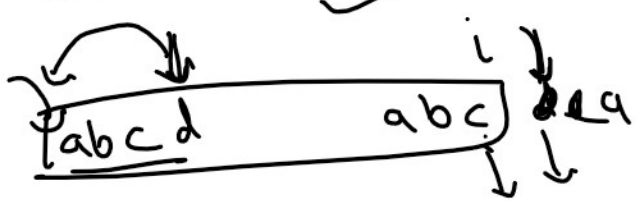
abcdeabc → lps = 3

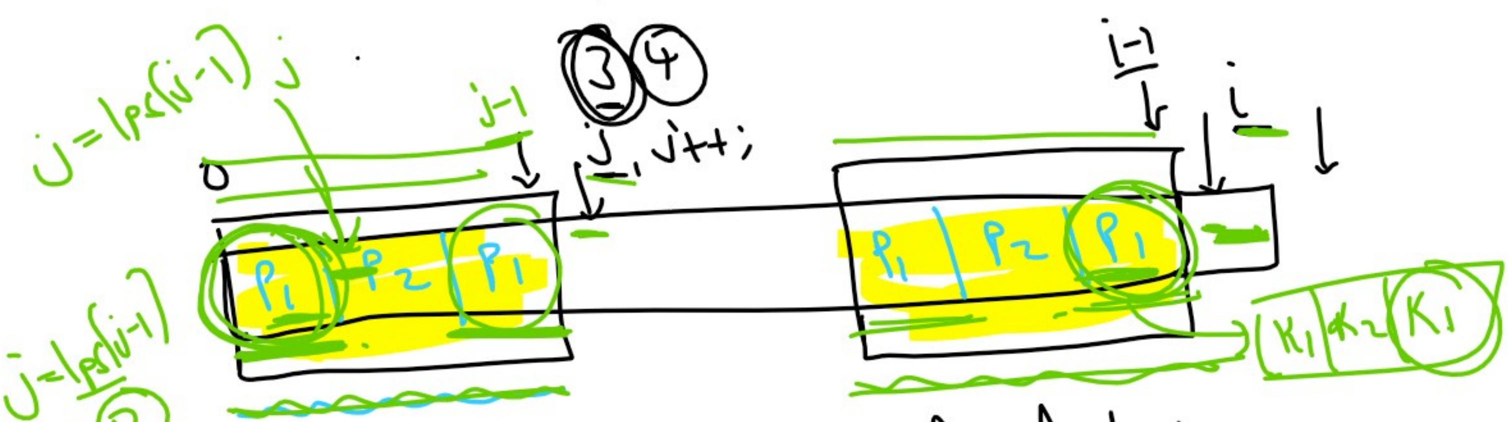
abcabdpabc

0 0 0 1 2 0 0 1 2 3

a → 0
ab → 0
abc → 0
abca → 1
abcab → 2
abcabd →
abcabdp → 0
abcabdpa
abcabdpab
abcabdpabc







lps[0] to lps[i-1] is calculated.

for (i=0; i<n;

{ if(s[i] == s[j])

{ lps[i] = j+1; j++; i++; }

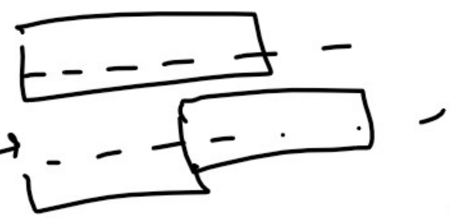
else {

lps(i ...

}else{
    if(j==0)
    lps(i)=0; i++;
    else
    j=lps(j-1);
}
}

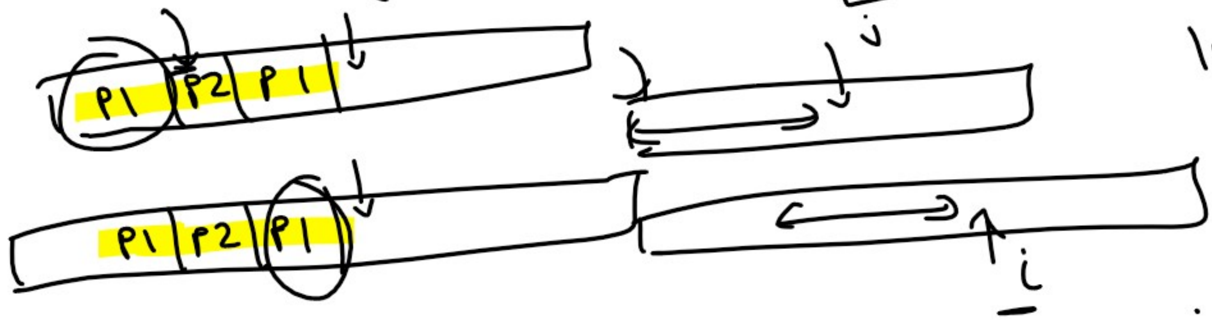
⇒ lps → pat
txt ↗

pat →
txt →



j=lps(j-1)

lps(j-1)



0 to (i-1) lps values are calculated.

j=lps(j-1)   (|P1|+|P2|+|P1|)

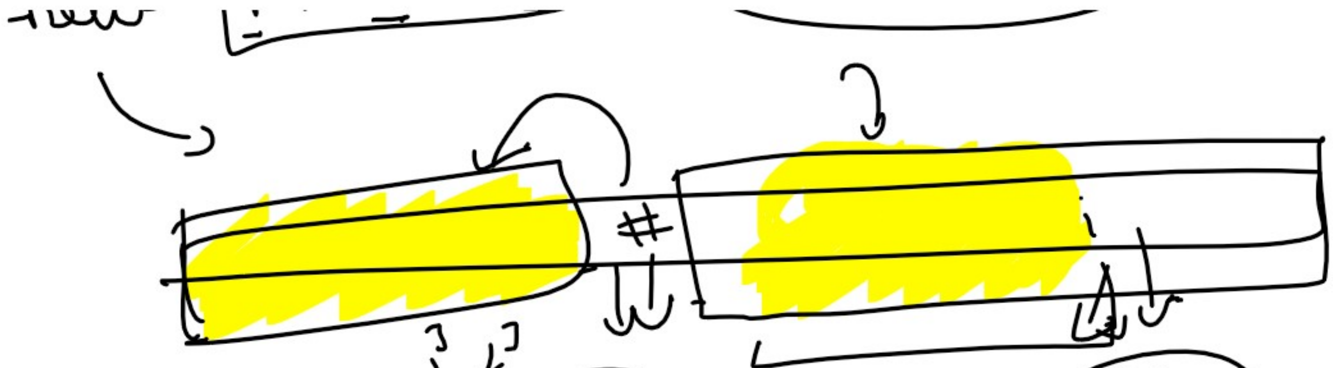
if(s(i)!=s(j)){
    if(j==0)
    lps(i)=0; i++;
    else
    j=lps(j-1);
}

(m)

→ p, t → lps

new = p + '#' + t

lps(i) = m



aaz # aada → 2

n + n + m

O(n + m)

n = 11, {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}

s 3

(i+1, s+1) (i+2, s+1) (i+3, s+1)

DP

~~dp(n~~ dp(n) → ∞

dp(0) = 0;

for(i=0; i<n; i++)

{ if(dp(i) != ∞){

dp(i)

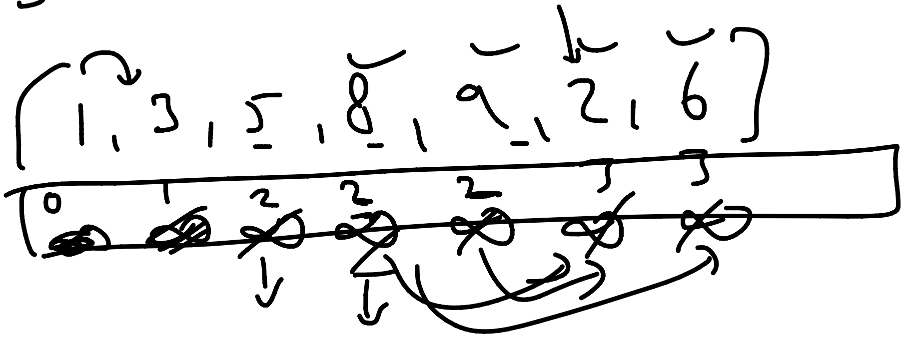
5

i+1

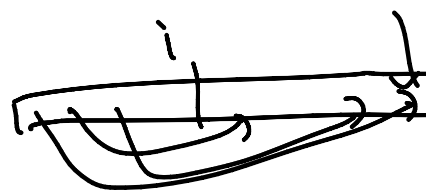
{ if(dp[i] != ∞) {

i+1
↓
dp[i]+1

} }



jump   maxReach   rem



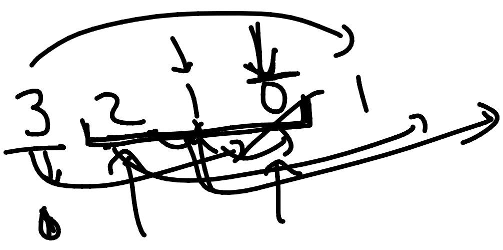
?   1

jump = 1+1

rem = arr[0] = 10

maxReach = 0 + arr[0]
= arr[0];

rem = maxReach − i



i = 1   rem = ~~3~~ ~~2~~ ~~1~~ 0

maxR = 3

jump = 1