

Word Break Wrap Problem

Date: _____
Page No. _____

nums = { 3, 2, 2, 5 } K = 6 output = 10

no. of char in one word

max limit of char in line

line 1	n ₀	n ₀	n ₀	s	n ₁	n ₁	Cost for 1st line = 0
line 2	n ₂	n ₂	space	s	s	s	→ 4 ² + 3 ² = 9
line 3	n ₃	n ₃	n ₃	n ₃	n ₃	s	→ 1 ² = 1

Total = 10

nums = { 3, 2, 2 } K = 4

n ₀	n ₀	n ₀
n ₁	n ₁	
n ₂	n ₂	

no. of char in one word

max no. of char in line

nums = { 3, 2, 2, 5 } K = 6

extra space

Costs extra spaces by whole square

line 1	n ₀	n ₀	n ₀	s	n ₁	n ₁	Cost = 0 ²
line 2	n ₂	n ₂	space	s	s	s	Cost = (4) ² = 16
line 3	n ₃	n ₃	n ₃	n ₃	n ₃	s	Total cost = 16

Cost zero as given that for last line Cost = 0

But this is NOT the minimum cost

line 1	n ₀	n ₀	s	n ₁	n ₁	Cost = 1 ²
line 2	n ₂	n ₂	n ₂	n ₂	n ₂	Cost = 3 ²
line 3	n ₃	n ₃	n ₃	n ₃	n ₃	Total cost = 10 Minimum

one condition is that word should be in the same order as in the array sequence

line 1	n_0	n_0	n_0				Cost = 3^2
line 2	n_1	n_1		n_2	n_2		Cost = 1^2
line 3	n_3	n_3	n_3	n_3	n_3		Total = <u>10</u> <u>Minimum</u>

So it is of choice type of ~~prog~~ problem

Look Hence

'Car' in 'by' night

car can fill 'car' in the 1st line

line 1	c	a	r			
line 2						
line 3						

for $n_0 \rightarrow 9$ have
2 choice either fill
it in line 1 or line 2

But we do not know
which will give us the
minimum Hence

We need to take the Both case in ^{finish one then 2 then} parallel ^{find min} and store it in ~~DP~~ the info of 'extra space' in the Matrix (or any other DS) Hence D.P.

I Don't know But how to code the DP problem

How to store ~~cost~~

so look we need a Matrix \Rightarrow of size
 $dp[nums.size()][k]$

constraint given

$1 \leq nums.size \leq 500$

Hence

$dp[501][2001]$

$max(nums[i]) \leq k \leq 2000$

to handle all
Test cases

Q2 How to write code
Base condition

+ some edge condition (optional)

{ logic function call

↳ on the basis of pick or
not pick

}

Q what to return

here we will return the cost after making a particular
choice for the given word

Q what will be the Base condition or edge
Condition

Base condition comes when we are at
the last word of name hence •

($i == \text{name.size}$)

and we know for last word where ever it
comes

~~scribbled out text~~

Base case if $(i == \text{num.size})$

return 0;

and to use the dp we will have

if $(dp[i][\text{rem}] \neq -1)$ // means for this we have
return $dp[i][\text{rem}]$; cal. earlier
lost

So, there are 2 possibility

Pick ① When the word will be in same line
next

Not Pick ② When the next word will be taken in next line

But in the ① case (Pick) there is also
a thing if remaining space is $< \text{word length}$
then we will ~~do~~ do only ① function
call ~~with~~ which is of ~~cost~~ Not Pick
(word is in next line)

if $(\text{num.size} > \text{rem})$
only one possibility Hence

$$\text{ans} = (\text{rem} + 1) * (\text{rem} + 1) + \text{rec}(i+1, K - \text{ans}[i] - 1, -)$$

Cal. relating cost

example

a | | | | |

$\text{num.size} = 5$ word = "night"
 rem is 4

Satisfy the above if condition
Hence before making call
for Another word (next word) $(i+1)$
we will cal. cost for this line

we have reserved the $(\text{ans}[i] + 1)$ space
word char of
"night"

in the next line Hence

we are passing $K - \text{ans}[i] - 1$ as

rem ~~to the~~ space for the $(i+1)$ th
word.

if the above if condn. unsatisfy means
we have both possibility for that word(right)
① pick ② not pick

Hence

else

int c1 = ~~sum~~ (sum+1) * (next) + rec(i+1, ^{cost} ~~sum~~ - arr[i] - 1, arr, k);

int c2 = rec(i+1, sum - arr[i] - 1, arr, k);

↓
Because
same line

then we will ~~also~~ choose the
minimum cost

ans = min(c1, c2);

⚡

dp[i][sum] = ans; // so that we reuse it
if it comes again for
evaluation

return ans;

⚡

right)

i+1,

k);

Because next line

$\text{arr}[i] - 1 \dots$; // Condition of Not pick that word on that particular line

// Condition of pick that word on that particular line Hence there is no cost as ~~cost~~

a	b	c	d
c	d		

num[i] = 2 word = "cd"
~~rem~~ = 3

One more thing here

rem can become -1 and we have use $\text{dp}[i][\text{rem}]$

line n

k	g		
a	b	c	d

num[0] = 4 word = "abcd"
 k = 4

in our function
 And in some compiler
 it can give segmented
 fault Hence
 you can use
 $\text{if}(\text{rem} > -1)$

$\text{int c} = \text{get}(\text{it}, \text{arr}[i] - 1 \dots)$

-1

$\text{int c} = \text{get}(\text{it}, \text{rem} - \text{arr}[i] - 1 \dots)$

-1

Hence for next word rem will be -1