String   Word Wrap

In this question, we will be having an array which will store the length of the word and a value k which tells us that how much letters can be stored in a single line.

We have to find the cost which is ^sum of the^ square of the ~~extr~~ no. of extra space each line contains except the last line.

Ex

arr = [3, 2, 2, 5]

K = 6

In this example, we have array of length 4 and k=6

So, line 1 can store first two words as (3+2) equal to 5 and one space b/w them.

~~for~~ now the 1st line is full it cannot contains any other letters or word.

We will move to second line, we will 1st store 2 in that then we will left with 4 space in which we cannot store the next word of character length 5

So we will left with 4 space we will square it and store in some resualtant vaiable.

We will move to 3rd line to store last word but as the question only suggest that we not need to count the extra space of the last line.

1st line   o o o _ o o   extraspace = 0

2nd line   o o _ _ _ _   extra space = $4^2 = 16$

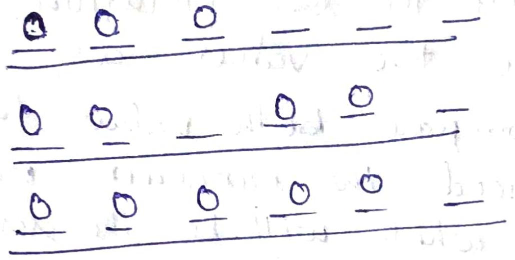
3rd line   o o o o o _   extra space = 1   = 0

$$\begin{array}{r} \\ \hline 16 \end{array}$$

result = 16

but the question ask that we need to find the minimum cost for that we will use recursive we will see by adding the word in the same line as will as adding the word in the another line then we will compare it with each other then the minimum of among two will be the output or result.

In the same ~~as~~ example as previous.



1st line — extraspace = 3 = 3^2 = 9

2nd line — extraspace = 1 = 1^2 = 1

3rd line — extraspace = 1 = x

$$\frac{}{10}$$

This is the minimum ~~ab~~ among two

we will acheieve this using recussion

```
SolveWordWrap (nums[],k){
  a= helper ( 1, num[0],nums; k);
  returna;
}   //recurssive.
helper (int curr, int space, nums[],k){      //base case
     if( curr==nums.length){
         return0; }

  int newspace = space +1 + nums[curr];   // same line.
  if (newspace <=k){
      a= helper (curr+1, newspace, nums, k);
  }
  // new Line
  b= (k-space)^2 + helper (curr+1, nums[curr], nums, k);

 return min(a,b);

}
```

In this recurssive will check every possible outcomes of the ~~co~~ then gives the minimum among them.

making one helper ~~co~~ method which will work as recurssive, it has two approach keeping the word in the same line if the k allows and also keeping the word in the new line and so on going ~~it~~ till the last iteration at last it will have two values as a & b the we will compare both values (a, b) among two we need the minimum value which will be the result for the cost. of the code.