A weighted string is a string of lowercase English letters where each letter has a weight. Character weights are 1 to 26 from a to z as shown below:

a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8
i	9
j	10

k	11
l	12
m	13
n	14
o	15
p	16
q	17
r	18

s	19
t	20
u	21
v	22
w	23
x	24
y	25
z	26

- The weight of a string is the sum of the weights of its characters. For example:

apple	1 + 16 + 16 + 12 + 5 = 50
hack	8 + 1 + 3 + 11 = 23
watch	23 + 1 + 20 + 3 + 8 = 53
ccccc	3 + 3 + 3 + 3 + 3 = 15
aaa	1 + 1 + 1 = 3
zzzz	26 + 26 + 26 + 26 = 104

- A uniform string consists of a single character repeated zero or more times. For example, `ccc` and `a` are uniform strings, but `bcb` and `cd` are not.

Given a string, s, let U be the set of weights for all possible uniform contiguous substrings of string s. There will be n queries to answer where each query consists of a single integer. Create a return array where for each query, the value is `Yes` if $query[i] \in U$. Otherwise, append `No`.

Note: The $\in$ symbol denotes that $x[i]$ is an element of set U.

Example

$s =$ 'abbcccdddd'

$queries = [1, 7, 5, 4, 15]$.

Working from left to right, weights that exist are:

```
string  weight
a      1
b      2
bb     4
c      3
cc     6
ccc    9
d      4
dd     8
ddd    12
dddd   16
```

Now for each value in $queries$, see if it exists in the possible string weights. The return array is `['Yes', 'No', 'No', 'Yes', 'No']`.

Function Description

Complete the weightedUniformStrings function in the editor below.

weightedUniformStrings has the following parameter(s):

- string s: a string

- int queries[n]: an array of integers

Returns

- string[n]: an array of strings that answer the queries

Input Format

```
#
# Complete the 'weightedUniformStrings' function below.
#
# The function is expected to return a STRING_ARRAY.
# The function accepts following parameters:
#  1. STRING s
#  2. INTEGER_ARRAY queries
#

def weightedUniformStrings(s, queries):
    weight = 1
    weights = {}

    alphas = string.ascii_lowercase

    # fill in the hash_map of weights of all the ascii lowercase characters
    for alpha in alphas:
        weights[alpha] = weight
        weight += 1


    # calculate the weights of all the uniform substrings and store in hash map
    # initializing hash map for that
    uniform_weights = {}

    # starting with alpha at idx 0 in string s
    # current alpha's weight
    alpha_weight = weights[s[0]]

    # current uniform substring's weight
    uniform_weight = alpha_weight

    # storing the current uniform substring's character in a buffer var
    # current alpha
    buffer = s[0]

    # to store the result
    result = []

    for idx in range(1, len(s)):
        # if this conditions is true
        # the current substring is still uniform
        if s[idx] == buffer:
            # update the map of uniform weights
            uniform_weights[uniform_weight] = buffer
            # update the current uniform string's weight
            uniform_weight += alpha_weight

        else:
            # insert the weight of last uniform substring
            uniform_weights[uniform_weight] = buffer

            # now change the buffer
            buffer = s[idx]
            alpha_weight = weights[s[idx]]
            # new uniform substring's weight is equal to current character's weight
            uniform_weight = alpha_weight

    # the last uniform weight would not have been added to the map
    # so, add it explicitly
    uniform_weights[uniform_weight] = buffer

    # check if the query ⊂ uniform_weights hash map
    for query in queries:
        if query in uniform_weights:
            result.append('Yes')
        else:
            result.append('No')

    return result
```