A DNA sequence can be represented as a string consisting of the letters A, C, G and T, which correspond to the types of successive nucleotides in the sequence. Each nucleotide has an *impact factor*, which is an integer. Nucleotides of types A, C, G and T have impact factors of 1, 2, 3 and 4, respectively. You are going to answer several queries of the form: What is the minimal impact factor of nucleotides contained in a particular part of the given DNA sequence?

The DNA sequence is given as a non-empty string S = S[0]S[1]...S[N-1] consisting of N characters. There are M queries, which are given in non-empty arrays P and Q, each consisting of M integers. The K-th query (0 ≤ K < M) requires you to find the minimal impact factor of nucleotides contained in the DNA sequence between positions P[K] and Q[K] (inclusive).

For example, consider string S = CAGCCTA and arrays P, Q such that:

P[0] = 2 Q[0] = 4 P[1] = 5 Q[1] = 5 P[2] = 0 Q[2] = 6

The answers to these M = 3 queries are as follows:

* The part of the DNA between positions 2 and 4 contains nucleotides G and C (twice), whose impact factors are 3 and 2 respectively, so the answer is 2.
* The part between positions 5 and 5 contains a single nucleotide T, whose impact factor is 4, so the answer is 4.
* The part between positions 0 and 6 (the whole string) contains all nucleotides, in particular nucleotide A whose impact factor is 1, so the answer is 1.

Write a function:

def solution(S, P, Q)

that, given a non-empty string S consisting of N characters and two non-empty arrays P and Q consisting of M integers, returns an array consisting of M integers specifying the consecutive answers to all queries.

Result array should be returned as an array of integers.

For example, given the string S = CAGCCTA and arrays P, Q such that:

P[0] = 2 Q[0] = 4 P[1] = 5 Q[1] = 5 P[2] = 0 Q[2] = 6

the function should return the values [2, 4, 1], as explained above.

Write an **efficient** algorithm for the following assumptions:

* N is an integer within the range [1..100,000];
* M is an integer within the range [1..50,000];
* each element of arrays P and Q is an integer within the range [0..N - 1];
* P[K] ≤ Q[K], where 0 ≤ K < M;
* string S consists only of upper-case English letters A, C, G, T.

def getmin\_impact(string2):

if "A" in string2:

return 1

elif "C" in string2:

return 2

elif "G" in string2:

return 3

elif "T" in string2:

return 4

else:

raise string2

def solution(S, P, Q):

arr1 = []

for i,j in zip(P,Q):

arr1.append(getmin\_impact(S[i:j+1]))

return arr1

or

def solution(S, P, Q):

arr2 = []

for i,j in zip(P,Q):

if "A" in S[i:j+1]:

arr2.append(1)

elif "C" in S[i:j+1]:

arr2.append(2)

elif "G" in S[i:j+1]:

arr2.append(3)

elif "T" in S[i:j+1]:

arr2.append(4)

else:

return S[i:j+1]

return arr2

S ='CAGCCTA'

P = [2, 5, 0]

Q = [4, 5, 6]

car1 = solution(S,P,Q)

print(car1)