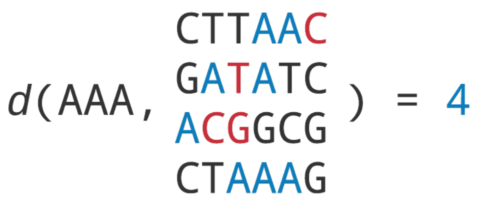
2H Implement DistanceBetweenPatternAndStrings

Distance Between Pattern and Strings Problem
Compute DistanceBetweenPatternAndStrings.

Input: A DNA string *Pattern* and a collection of DNA strings *Dna*.
Output: Distance D(*Pattern*, *Dna*) between *Pattern* and *Dna*.



Formatting

Input: A DNA string *Pattern*, followed by a space-separated collection of DNA strings *Dna*.
Output: An integer representing the output of DistanceBetweenPatternAndStrings(*Pattern*, *Dna*).

Constraints

- The length of *Pattern* will be between 1 and 10^1.
- The number of strings in *Dna* will be between 1 and 10^2.
- The length of each string in *Dna* will be between 1 and 10^2.
- *Pattern* and each string in *Dna* will be DNA strings.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
AAA
TTACCTTAAC GATATCTGTC ACGGCGTTCG CCCTAAAGAG CGTCAGAGGT
```

Output:

```
5
```

Case 2

Description: This dataset checks multiple potential mistakes. First, it checks that you are actually using all three sequences of *Dna* (and not just a single sequence). The Hamming Distance between *Pattern* and each individual sequence in *Dna* is 1, so if your code returns a total score of 1, we fail it for this reason. Next, it checks if you are only using the first *k*-mer in each sequence of Dna. For example, if you do this, you would output d(TAA,TTT)+d(TAA,CCT)+d(TAA,GGT) which is 8, instead of the correct answer of 3. Finally, it checks if you are only using the last *k*-mer in each sequence of *Dna*. For example, if you do this, you would output d(TAA,TTT)+d(TAA,CAC)+d(TAA,GAG) which is 6, instead of the correct answer of 3.

Input:

```
TAA
TTTATTT CCTACAC GGTAGAG
```

Output:

```
3
```

Case 3

Description: This dataset checks if your code is using maximum or sum instead of minimum. First, it checks if your code is using maximum instead of minimum. In this case, the output would be d(AAA,ACT)+d(AAA,AAC)+d(AAA,AAG), which is 4, instead of the correct answer of 0. Next, it checks if your code is using sum instead of minimum. In this case, the output would be d(AAA,AAA)+d(AAA,AAC)+d(AAA,ACT)+d(AAA,AAA)+d(AAA,AAC)+d(AAA,AAA)+d(AAA,AAG), which is 5, instead of the correct answer of 0.

Input:

```
AAA
AAACT AAAC AAAG
```

Output:

```
0
```

Case 4

Description: This dataset checks if your code has an off-by-one error at the end of each sequence of *Dna*. Notice that each sequence has a perfect match of `AAA` at the very end, so if your code returns a nonzero answer to this test dataset, it must have missed the last k-mer of each.

Input:

```
AAA
TTTTAAA CCCCAAA GGGGAAA
```

Output:

```
0
```

Case 5

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.