Median String Problem

Input: An integer *k*, followed by a collection of strings *Dna*

Output: A *k*-mer *Pattern* that minimizes *d*(*Pattern*, *Dna*) among all *k*-mers *Pattern*. (If there are multiple such strings *Pattern*, then you may return any one.)

SAMPLE DATASET:

Input:

```
3
AAATTGACGCAT
GACGACCACGTT
CGTCAGCGCCTG
GCTGAGCACCGG
AGTACGGGACAG
```

Output:

```
ACG or GAC (not both)
```

The sample dataset is not actually run on your code. Notice that there are technically two solutions to the problem ("ACG" and "GAC" are equally optimal), but the problem specifically states to only return a single output (you can arbitrarily pick any optimal solution).

TEST DATASET 1:

Input:

```
3
ACGT
ACGT
ACGT
```

Output:

```
ACG or CGT (not both)
```

This dataset checks that your output is the correct length. Notice that there are technically two solutions to the problem ("ACG" and "CGT" are equally optimal), but the problem specifically states to only return a single output (you can arbitrarily pick any optimal solution). Also, since k = 3 in this dataset, we check that your output is exactly of length k (should not be any longer or shorter than this).

TEST DATASET 2:

Input:

```
3
ATA
ACA
AGA
AAT
AAC
```

Output:

```
AAA
```

This dataset checks if your code considers k-mers that do not occur in Dna. Notice that the best 3-mer is "AAA", which does not actually occur in any of the sequences in Dna. It is perfectly fine that our optimal median string does not actually occur in any of the sequences in Dna (similar to the Frequent Words With Mismatches Problem from the Replication chapter).

TEST DATASET 3:

Input:

```
3
AAG
AAT
```

Output:

```
AAG or AAT (not both)
```

This dataset checks that your output only contains a single k-mer. Notice that there are technically two solutions to the problem ("AAG" and "AAT" are equally optimal), but the problem specifically states to only return a single output (you can arbitrarily pick any optimal solution).