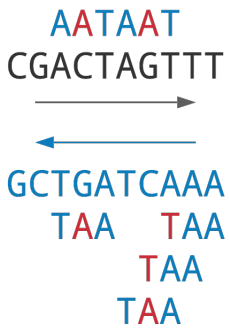
1J Find Frequent Words with Mismatches and Reverse Complements

Frequent Words with Mismatches and Reverse Complements Problem
Find the most frequent k-mers (with mismatches and reverse complements) in a DNA string.

Input: A DNA string *Text* as well as integers k and d.
Output: All k-mers *Pattern* maximizing the sum $\text{COUNT}_d(Text, Pattern)+\text{COUNT}_d(Text, \overline{Pattern})$ over all possible k-mers.



Formatting

Input: A DNA string *Text* as well as integers k and d.
Output: A space-separated list of strings representing all k-mers *Pattern* maximizing the sum $\text{COUNT}_d(Text, Pattern)+\text{COUNT}_d(Text, \overline{Pattern})$ over all possible k-mers.

Constraints

- The length of *Text* will be between 1 and 10^3.
- The integer k will be between 1 and 10^1.
- The integer d will be between 1 and 10^1.
- *Text* will be a DNA string.

Test Cases

Case 1

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

Input:

```
ACGTTGCATGTCGCATGATGCATGAGAGCT
4 1
```

Output:

```
ACAT ATGT
```

Case 2

Description: *Text* contains partial and completes matches for the most frequent word.

Input:

```
AAAAAAAAAA
2 1
```

Output:

```
AT TA
```

Case 3

Description: This dataset makes sure that your code is not accidentally swapping k and d.

Input:

```
AGTCAGTC
4 2
```

Output:

```
AATT GGCC
```

Case 4

Description: This dataset makes sure you are finding k-mers in both *Text* and the reverse complement of *Text*.

Input:

```
AATTAATTGGTAGGTAGGTA
4 0
```

Output:

```
AATT
```

Case 5

Description: This dataset first checks that *k*-mers with exactly *d* mismatches are being found. Then, it checks that *k*-mers with less than *d* mismatches are being allowed (i.e. you are not only allowing *k*-mers with exactly *d* mismatches). Next, it checks that you are not returning too few *k*-mers. Last, it checks that you are not returning too many *k*-mers.

Input:

```
ATA
3 1
```

Output:

```
AAA AAT ACA AGA ATA ATC ATG ATT CAT CTA GAT GTA TAA TAC TAG TAT TCT TGT TTA
TTT
```

Case 6

Description: This dataset checks that your code is looking at *both Text* and its reverse complement (i.e. not just looking at *Text*, and not just looking at the reverse complement of *Text*, but looking at both).

Input:

```
AAT
3 0
```

Output:

```
AAT ATT
```

Case 7

Description: This dataset checks that your code correctly delimiting your output (i.e. using spaces) and verifies that your *k*-mers are actually of length *k*.

Input:

```
TAGCG
2 1
```

Output:

```
CA CC GG TG
```

Case 8

Description: A larger dataset of the same size as that provided by the randomized autograder.