Greedy Motif Search

Input: Integers *k* and *t*, followed by a collection of strings *Dna*.
Output: A collection of strings *BestMotifs* resulting from applying *GreedyMotifSearch*(*Dna*,*k*,*t*). If at any step you find more than one *Profile*-most probable *k*-mer in a given string, use the one occurring first.

Pseudocode

```
GreedyMotifSearch(k,t,Dna)
   bestMotifs ← empty list (score of 0)
   for i from 0 to |Dna[0]| - k
      motifs ← list with only Dna[0](i,k)
      for j from 1 to |Dna| - 1
         Add ProfileMostProbableKmer(Dna[j],k,Profile(motifs))
         to motifs
      if score(motifs) > score(bestMotifs)
         bestMotifs = motifs
   return bestMotifs
```

SAMPLE DATASET:

Input:

3 5
GGCGTTCAGGCA
AAGAATCAGTCA
CAAGGAGTTCGC
CACGTCAATCAC
CAATAATATTCG

Output:

CAG
CAG
CAA
CAA
CAA

The sample dataset is not actually run on your code.

TEST DATASET 1:

Input:

3 4
GCCCAA
GGCCTG
AACCTA
TTCCTT

Output:

GCC
GCC
AAC
TTC

This dataset checks that your code always picks the first-occurring Profile-most Probable k-mer in a given sequence of Dna. In the first sequence ("GCCCAA"), "GCC" and "CCA" are both Profile-most Probable k-mers. However, you must return "GCC" since it occurs earlier than "CCA". Thus, if the first sequence of your output is "CCA", this test case fails your code.

TEST DATASET 2:

Input:

5 8
GAGGCGCACATCATTATCGATAACGATTCGCCGCATTGCC
TCATCGAATCCGATAACTGACACCTGCTCTGGCACCGCTC
TCGGCGGTATAGCCAGAAAGCGTAGTGCCAATAATTTCCT
GAGTCGTGGTGAAGTGTGGGTTATGGGGAAAGGCAGACTG
GACGGCAACTACGGTTACAACGCAGCAACCGAAGAATATT
TCTGTTGTTGCTAACACCGTTAAAGGCGGCGACGGCAACT
AAGCGGCCAACGTAGGCGCGGCTTGGCATCTCGGTGTGTG
AATTGAAAGGCGCATCTTACTCTTTTCGCTTTCAAAAAAA

Output:

GAGGC
TCATC
TCGGC
GAGTC
GCAGC
GCGGC
GCGGC
GCATC

This dataset checks if your code has an off-by-one error at the beginning of each sequence of Dna. Notice that the first four motifs of the solution occur at the beginning of their respective sequences in Dna, so if your code did not check the first k-mer in each sequence of Dna, it would not find these sequences.

TEST DATASET 3:

Input:

6 5

GCAGGTTAATACCGCGGATCAGCTGAGAAACCGGAATGTGCGT

CCTGCATGCCCGGTTTGAGGAACATCAGCGAAGAACTGTGCGT

GCGCCAGTAACCCGTGCCAGTCAGGTTAATGGCAGTAACATTT

AACCCGTGCCAGTCAGGTTAATGGCAGTAACATTTATGCCTTC

ATGCCTTCCGCGCCAATTGTTCGTATCGTCGCCACTTCGAGTG

Output:

GTGCGT

GTGCGT

GCGCCA

GTGCCA

GCGCCA

This dataset checks if your code has an off-by-one error at the end of each sequence of Dna. Notice that the first two motifs of the solution occur at the end of their respective sequences in Dna, so if your code did not check the end k-mer in each sequence of Dna, it would not find these sequences.

TEST DATASET 4:

Input:

5 8

GACCTACGGTTACAACGCAGCAACCGAAGAATATTGGCAA
TCATTATCGATAACGATTCGCCGGAGGCCATTGCCGCACA
GGAGTCTGGTGAAGTGTGGGTTATGGGGCAGACTGGGAAA
GAATCCGATAACTGACACCTGCTCTGGCACCGCTCTCATC
AAGCGCGTAGGCGCGGCTTGGCATCTCGGTGTGTGGCCAA
AATTGAAAGGCGCATCTTACTCTTTTCGCTTAAAATCAAA
GGTATAGCCAGAAAGCGTAGTTAATTTCGGCTCCTGCCAA
TCTGTTGTTGCTAACACCGTTAAAGGCGGCGACGGCAACT

Output:

GCAGC
TCATT
GGAGT
TCATC
GCATC
GCATC
GGTAT
GCAAC

This test dataset checks if your code is correctly breaking ties when calling Profile-most Probable k-mer. Specifically, it makes sure that, when you call Profile-most Probable k-mer, in the event of a tie, you choose the first-occurring k-mer.