The Gibbs Sampler algorithm first generates a completely random set of motifs by randomly taking a single kmer from each sequence of dna. After that, runs through a loop an arbitrary number of times, N both in the pseudocode and in my code. Inside the loop, it first selects a random dna string from the set. It then creates a profile using the set of random motifs, excluding the motif from the selected dna string. A new motif set is then created by inserting a ProfileRandom generated motif where the randomly generated motif had been. The way the ProfileRandom generation works is by using the profile to test the probabilities of each kmer in the dna sequence. It then uses these probabilities as relative weights to randomly choose one. Once the new motif has been generated, the motif is scored and compared to the previous best motif, and replaces it if it has a lower score. The loop then starts again, until it has finished.

Outputs:

Score: 13
AGTAACCG AGGTGCCA TAGTACCG AGGTGCAC TGTTGGCC

Score: 19
GGGGTGTT GTGTAAGT GAGACCGA AGGTGCAC GCAATGTT

Score: 13
CAGTAACC AAGTGCCA TAGTACCG AAGTTTCA ATGTTGGC

Score: 13
GTAACCGG GTGCCAAG GTATACAG GTGCACGT CTCCACGT

Score: 16
CTCGGGGG CCAAGGTG CCGAGACC CGTCGGTG GCTCCACG

Score: 12
CAGTAACC AAGTGCCA AAGTATAC AAGTTTCA CAGCTCCA

Score: 10
CTCGGGGG CCAAGGTG TACAGGCG TTCAGGTG TCCACGTG

Score: 13
CAGTAACC AGGTATGT AAGTATAC CGGTGAAC AGCTCCAC

Score: 14
TTCAGTAA ATGTGTAA GTATACAG TAGATCAA ACGTGCAA

Score: 16
AACCGGCC AAGGTGCC ATACAGGC CGTCGGTG ATGTTGGC

```
import random


def CreateProfile(motifs):
    length = len(motifs[0])
    profile = [None] * 4
    for i in range(4):
        profile[i] = [1] * length
    for motif in motifs:
        for i in range(length):
            base = motif[i]
            match base:
                case 'A':
                    profile[0][i] += 1
                case 'C':
                    profile[1][i] += 1
                case 'G':
                    profile[2][i] += 1
                case 'T':
                    profile[3][i] += 1
    for i in range(4):
        for j in range(length):
            profile[i][j] = profile[i][j] / len(motifs)
    return profile

def ProbabilityCalc(sequence, profile):
    probability = 1
    for i in range(len(sequence)):
        if sequence[i] == 'A':
            probability = probability * profile[0][i]
        if sequence[i] == 'C':
            probability = probability * profile[1][i]
        if sequence[i] == 'G':
            probability = probability * profile[2][i]
        if sequence[i] == 'T':
            probability = probability * profile[3][i]
    return probability

def ScoreMotif(motifs):
    score = 0
    numMotifs = len(motifs)
    length = len(motifs[0])
    count = [None] * 4
    for i in range(4):
        count[i] = [0] * length
    for motif in motifs:
        for i in range(length):
```

```
            base = motif[i]
            match base:
                case 'A':
                    count[0][i] += 1
                case 'C':
                    count[1][i] += 1
                case 'G':
                    count[2][i] += 1
                case 'T':
                    count[3][i] += 1
    for i in range(length):
        column = []
        for j in range(4):
            column.append(count[j][i])
        score = score + numMotifs - max(column)
    return score

def LoadedRandom(numbers):
    total = sum(numbers)
    for i in range(len(numbers)):
        numbers[i] = numbers[i] / total
    r = random.random()
    for i in range(len(numbers)):
        r = r-numbers[i]
        if r <= 0:
            return i

def RandomKmer(sequence, k):
    r = random.randint(0, len(sequence) - k)
    return sequence[r:r+k]

def ProfileRandomKmer(sequence, profile, k):
    probs = []
    for i in range(len(sequence) - k):
        probs.append(ProbabilityCalc(sequence[i:i+k], profile))
    r = LoadedRandom(probs)
    return sequence[r:r+k]

def GibbsSampler(dna, k, t, N):
    motifs = []
    for sequence in dna:
        motifs.append(RandomKmer(sequence, k))
    bestMotifs = motifs
    for i in range(N):
        j = random.randint(0, t-1)
        profile = CreateProfile([x for y,x in enumerate(motifs) if y!=j])
        motifs[j] = ProfileRandomKmer(dna[j], profile, k)
```

```
        if ScoreMotif(motifs) < ScoreMotif(bestMotifs):
            bestMotifs = motifs
    return bestMotifs

k = 8
t = 5
N = 100
sequences = ['CGCCCCTCTCGGGGGTGTTCAGTAACCGGCCA',
'GGGCGAGGTATGTGTAAGTGCCAAGGTGCCAG',
'TAGTACCGAGACCGAAAGAAGTATACAGGCGT',
'TAGATCAAGTTTCAGGTGCACGTCGGTGAACC',
'AATCCACCAGCTCCACGTGCAATGTTGGCCTA']

motifs = GibbsSampler(sequences, k, t, N)
output = "Score: " + str(ScoreMotif(motifs)) + "\n"
for motif in motifs:
    output = output + motif + ' '
print(output)
```