

# GenomeSequence

April 25, 2025

```
[1]: from Bio.SeqUtils import GC
from Bio.Seq import Seq
import matplotlib.pyplot as plt
import numpy as np

# UL52 nucleotide sequence
ul52_sequence = (
    "ATGGGGACGGAAGACTGCGATCACGAAGGGCGGTGCGTTGCGGCTCCCGTGGAGGTTACGGCGCTGTATG"
    "CGACCGACGGGTGCGTTATACCTCCTCGCTCGCCCTCCTCACAACTGCCTGCTGGGGGCCGAGCCGTT"
    "GTATATATTAGCTACGACGCTACCGGCCGATGCGCCCAATGGCCCCACGGGCGCGCCACCGAACAG"
    "GAGAGGTTTCGAGGGGAGCCGGGCGCTCTACCGGATGCGGGGGGGCTAAATGGCGATTCAATTCGGGTGA"
    "CCTTTTGTATTATTGGGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT"
    "GTGCCGCTTCGAGCGAGCGGACGACGTCGCCGTGCTCCAAGACGCCCTGGGCCGCGGGACCCCATGCTC"
    "CCGGCCACATCACAGCAACTCTGGAATTGGAGGCGACGTTTGGCTCCACGCTAACATCATCATGGCTC"
    "TCACCGTGGCCATCGTCCACAACGCCCCCGCCGATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCC"
    "CGGCGAATCGATGCGCTCGTTCGTCGGGCGCATGTCCCTGGGGCAGCGCGGCCTCACCACGCTGTTCTGTG"
    "CACACGAGGCGCGCGTGTGGCGGCGTACCGCGGGCGTATTATGGGAGCGCCCAAAGCCCTTTGGT"
    "TTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGTGTCGGCGCTAGGTACTACCTACTCCAGGCTCC"
    "GCGCTTGGGGGGCGCCGGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATC"
    "CCCCACGACCCACGCCCCGACACCCTCAGTGCCGCGTCCTTGACCTCGTTTCGCCGCCATCACTCGGTTCT"
    "GTTGCACGAGCCAGTACTCCCGCGGGCGCGGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGATCGC"
    "CGCCGACGTACGCGAGACCGGCGCGCTGGAGAAGTTCATCGCCACGATCGCAGCTGCCTGCGCGTGTCC"
    "GACCGGAATTCATTACGTACATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCCGCGCCTGGCCACGC"
    "ATCTCCGGGCGGTGACCACCCAGACCCAGCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCCCTGGGTCCG"
    "GGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAAC"
    "GTCACCCCCAGGAAACCGCCCTGGCGGGAGACGCGGCCCGCCCTACCTGCGCGCGCGCACGTATGCCC"
    "CGGCGGCCCTCACGCCCGCCCCCGCTACTGCGGGGTGCGAGACTCGTCCACCAAAATGATGGGACGTCT"
    "GGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCGGTTTCGCACCAACAACCCCGGGGACGAC"
    "GCGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGG"
    "AGCAGCAGGGCACGACGCCCCCGCGATCGCGGCTCTCATGCAGGACGCGTCGGTCCAAACCCCTGCC"
    "CGTGTAACAGGATTACCATGTCCCCGACCGGCCAGGCGTTTGCCGCGGCGGCGCGGGACGACTGGGCCCCG"
    "GTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTCGCGGACGCGGCGGCGGCGCCGAGCCCGCG"
    "CGCTCGGCGGCGGCTCACGCGCCGATTTGCGCCCGGGGCCCCGCGCTCCCCCGGGCGGCGCTGGCCGT"
    "CGGGGGCCAGATGTACGTGAACCGCAACGAGATCTTCAACGCCGCGTGGCCGTTACGAACATCATCTG"
    "GATCTGGACATCGCCCTGAAGGAGCCCGTCCCCTTTCCCCGGCTCCACGAGGCCCTGGGTCACTTTAGGC"
    "GCGGGGCGCTGGCGGCGGTTACGCTGTTGTTTCCCGCGGCCCGGTAGACCCCGACGCCTATCCCTGTTA"
    "TTTTTTCAAAGCGCCTGTCGGCCCCGCGCGCCCGCTCTGTGCGGGCGACGGGCCCCCTGGCCGTGGC"
```

```

"GACGACGGCGACGGGGACTGGTTCCCCGACGCCGGTGGTCCCGGCGACGAGGAGTGGGAGGAGGACACGG"
"ACCCCATGGACACGACCCACGGCCCCCTCCCGGACGACGAGGCCGCGTACCTCGACCTGCTACACGAACA"
"GATACCAGCGGCGACGCCAGCAACCGGACTCCGTGCTGTGTTCTGCGCCGACAAGATCGGGCTGCGC"
"GTGTGCCTACCGGTCCCCGCCCCGTACGTTGTGCACGGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGA"
"TCCAGCAGGCGGTGCTGTTGGACCGCGACTTCGTGGAGGCCGTAGGGAGCCACGTAAAGAACTTTTGCT"
"GATCGATACGGGCGTGTACGCCACGGCCACAGCCTGCGCTTGCCGTATTTGCGCAAGATCGGCCCCGAC"
"GGCTCCGCGTGCGGCCGTTATTGCCCGTCTTCGTGATCCCCCGCGTGCGAGGACGTTCCGGCGTTCG"
"TCGCCGCGCACGCCGACCCGCGGCGCTTCCACTTTCACGCCCCGCCATGTTTTCCGCGGCCCGCGGGA"
"GATCCGCGTCTCCACAGCCTGGGCGGGGACTATGTCAGCTTTTTTCGAGAAGAAGGCGTCGCGCAACGCC"
"CTGGAGCACTTTGGGCGACGCGAGACCCTGACGGAGGTTCTGGGCCGCTACGATGTGCGGCCGACGCCG"
"GGGAGACCGTGGAGGGGTTTCGCGTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCACTTTC"
"CGAGCACGCGCGGGAATATCAGGCCGTGTCCGTTCCCGGGCCGTCATTAAGGACGACTGGGTCTGCTG"
"CAGCTGATCCCCGGCCGCGCGCCCTGAACCAAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAA"
"GTCGCGGACGCGCCCGACCTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTG"
"TCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATGCGGGCACG"
"CCATGCTCGCGTCCGCTCCCTCCAGCACCTCACACCGTCATCTTCATAA"
)

# Convert to BioPython sequence object
ul52_seq = Seq(ul52_sequence)

# 1. Calculate GC Content
gc_content = GC(ul52_seq)

# 2. Sliding window for GC content
window_size = 100
gc_content_windows = [
    GC(ul52_seq[i:i+window_size]) for i in range(0, len(ul52_seq) - window_size,
↪ 1, window_size)
]

# 3. Plot GC content
plt.figure(figsize=(10, 6))
plt.plot(gc_content_windows, label="GC Content (%)", linewidth=2)
plt.axhline(y=gc_content, color='r', linestyle='--', label=f"Average GC Content:
↪ {gc_content:.2f}%")
plt.xlabel("Window (100 bp steps)", fontsize=12)
plt.ylabel("GC Content (%)", fontsize=12)
plt.title("GC Content Across HSV-2 UL52 Gene", fontsize=14)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

```

-----  
**ImportError**

Traceback (most recent call last)

```

Cell In[1], line 1
----> 1 from Bio.SeqUtils import GC
      2 from Bio.Seq import Seq
      3 import matplotlib.pyplot as plt

ImportError: cannot import name 'GC' from 'Bio.SeqUtils' (C:
↪\Users\Developer\anaconda3\Lib\site-packages\Bio\SeqUtils\__init__.py)

```

```

[3]: from Bio.Seq import Seq
import matplotlib.pyplot as plt
import numpy as np

# UL52 nucleotide sequence
ul52_sequence = (
    "ATGGGGACGGAAGACTGCGATCACGAAGGGCGGTGCGTTGCGGCTCCCGTGGAGGTTACGGCGCTGTATG"
    "CGACCGACGGGTGCGTTATCACCTCCTCGCTCGCCCTCCTCACAAACTGCCTGCTGGGGGCGGAGCCGTT"
    "GTATATATTCAGCTACGACGCGTACCGGCCGATGCGCCCAATGGCCCCACGGGCGCGCCACCGAACAG"
    "GAGAGGTTTCAGGGGAGCCGGGCGCTCTACCGGATGCGGGGGGCTAAATGGCGATTCATTCGGGTGA"
    "CCTTTTGTATTGTTGGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT"
    "GTGCCGCTTCGAGCGAGCGGACGACGTGCGCGTGCTCCAAGACGCCCTGGGCCGCGGGACCCCATGTGCTC"
    "CCGGCCACATCACAGCAACTCTGGACTTGAGGCGACGTTTTCGCTCCACGCTAACATCATCATGGCTC"
    "TCACCGTGGCCATCGTCCACAACGCCCCCGCCGATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCC"
    "CGGCGAATCGATGCGCTCGTTCGTCGCGGCGCATGTCCCTGGGGCAGCGCGGCCTCACCACGCTGTTCTG"
    "CACACGAGGCGCGCGTGTGGCGGCGTACCGCCGGGCGTATTATGGGAGCGCCCAAAGCCCCCTTTTGGT"
    "TTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGCTGGCCGCTAGGTACTACCTACTCCAGGCTCC"
    "GCGCTTGGGGGGCGCCGGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATC"
    "CCCCACGACCCACGCCCCGACACCCTCAGTGCCGCGTCTTTGACCTCGTTTCGCGCCATCACTCGGTTCT"
    "GTTGCACGAGCCAGTACTCCCGCGGGGCGCGGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGC"
    "CGCCGACGTACGCGAGACCGGCGCGCTGGAGAAGTTCATCGCCACGATCGCAGCTGCCTGCGCGTGTCC"
    "GACCGGAATTCATTACGTACATCTACCTGGCCACTTTGAGTGCTTCAGCCCCCGCGCTGGCCACGC"
    "ATCTCCGGGCCGTGACCACCCACGACCCAGCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCCGTGGTTCG"
    "GGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAAC"
    "GTCACCCCCAGGGAACCGCCCTGGCGGGAGACGCGGCCCGCCCTACCTGCGCGCGCGCACGTATGCCC"
    "CGGCGGCCCTCACGCCGCCCCCGCTACTGCGGGTTCGAGACTCGTCCACCAAAATGATGGGACGTCT"
    "GGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCGGTTTCGCACCAACAACCCCCGGGACGAC"
    "GCGGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGG"
    "AGCAGCAGGGCACGACGCCCCCGCGATCGCGGCTCTCATGCAGGACGCGTCGGTCCAAACCCCTGCC"
    "CGTGACAGGATTACCATGTCCCCGACCGGCCAGGCGTTTGGCGGGCGGCGCGGGACGACTGGGCCCGC"
    "GTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTCGCGGACGCGGCGGCGGCGCCGAGCCCGCGC"
    "CGCTCGGCCGCGGCTCACGCGCCGATTTGCGCCCGGGGCCCCGCGCTCCCCCGGGCGGCGCTGGCCGT"
    "CGGGGGCCAGATGTACGTGAACGCAACGAGATCTTCAACGCGCGCTGGCCGTTACGAACATCATCTG"
    "GATCTGGACATCGCCCTGAAGGAGCCCGTCCCCTTTCCCGGCTCCACGAGGCCCTGGGTCACTTTAGGC"
    "GCGGGGCGCTGGCGGCGGTTACGTGTTGTTTCCCGCGGCCCGCTAGACCCCGACGCTATCCCTGTTA"
    "TTTTTTCAAAGCGCCTGTGCGCCCCGCGCGCCCGCTGTGTGCGGGCGACGGGCCCCGTGGCCGTGGC"
    "GACGACGGGACGGGACTGGTTCCCCGACGCGGTGGTCCCGGCGACGAGGAGTGGGAGGAGACACGG"
    "ACCCCATGGACACGACCCACGGCCCCCTCCCGACGACGAGGCCGCTACCTCGACCTGCTACACGAACA"

```

```

"GATACCAGCGGCGACGCCAGCGAACCAGGACTCCGTCGTGTGTTCTGCGCCGACAAGATCGGGCTGCGC"
"GTGTGCCTACCGGTCCCCGCCCCGTACGTTGTGCACGGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGA"
"TCCAGCAGGCGGTGCTGTTGACCGCGACTTCGTGGAGGCCGTAGGGAGCCACGTAAAGAACTTTTGCT"
"GATCGATACGGGCGGTGTACGCCACGGCCACAGCCTGCGCTTGCCGTATTTGCGCAAGATCGGCCCCGAC"
"GGCTCCGCGTGCGGCCGTTATTGCCCGTCTTCGTGATCCCCCGCGGTGCGAGGACGTTCCGCGCTTCG"
"TCGCCGCGCACGCCGACCGCGCGCTTCCACTTTCACGCCCCGCCATGTTTTCCGCGCCCCGCGGA"
"GATCCGCGTCCTCCACAGCCTGGGCGGGGACTATGTCAGCTTTTTCGAGAAGAAGGCGTCGCGCAACGCC"
"CTGGAGCACTTTGGGCGACGCGAGACCCTGACGGAGTTCTGGGCCGCTACGATGTGCGGCCGACGCCG"
"GGGAGACCGTGAGGGGTTTCGCGTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCACTTTCC"
"CGAGCACGCGCGGAATATCAGGCCGTGTCGTTCCGCGCGGCCGTCATTAAGGACGACTGGGTCCTGCTG"
"CAGCTGATCCCCGCGCGCGCCCTGAACCAAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAA"
"GTCGCGGACGCGCCCGACCTTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTG"
"TCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATGCGGGCACG"
"CCATGCTCGCGTCCGCTCCCTCCAGCACCTCACGACCGTCATCTTCATAA"
)

# Function to calculate GC content manually
def calculate_gc(sequence):
    gc_count = sequence.count('G') + sequence.count('C')
    return (gc_count / len(sequence)) * 100

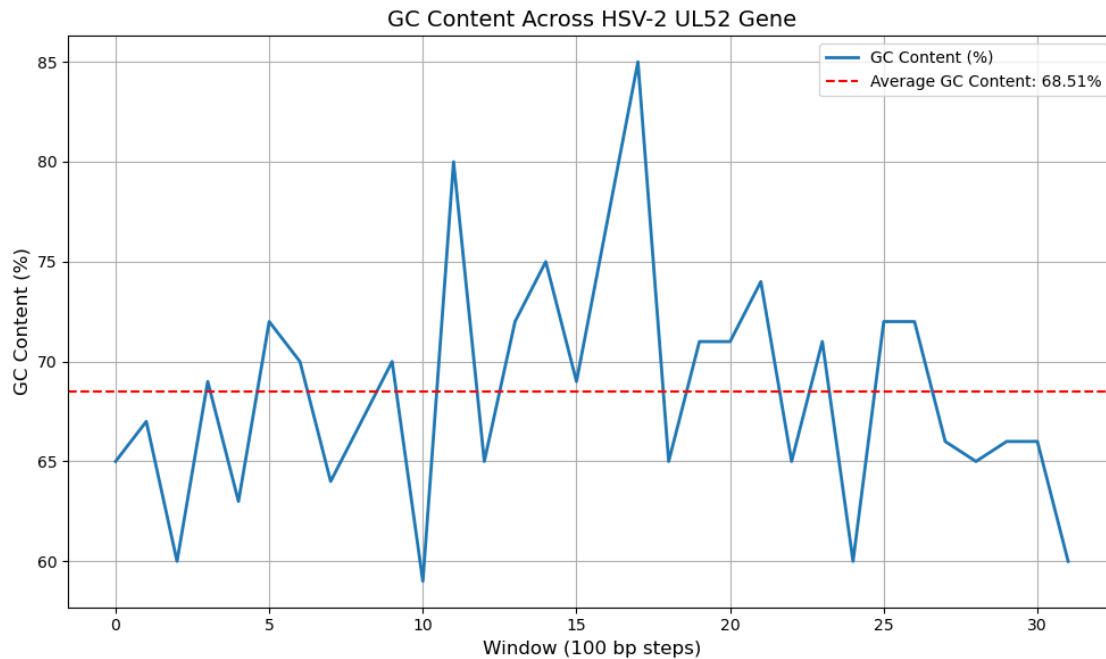
# 1. Calculate GC Content
gc_content = calculate_gc(ul52_sequence)

# 2. Sliding window for GC content
window_size = 100
gc_content_windows = [
    calculate_gc(ul52_sequence[i:i+window_size]) for i in range(0,
↪ len(ul52_sequence) - window_size + 1, window_size)
]

# 3. Plot GC content
plt.figure(figsize=(10, 6))
plt.plot(gc_content_windows, label="GC Content (%)", linewidth=2)
plt.axhline(y=gc_content, color='r', linestyle='--', label=f"Average GC Content:
↪ {gc_content:.2f}%")
plt.xlabel("Window (100 bp steps)", fontsize=12)
plt.ylabel("GC Content (%)", fontsize=12)
plt.title("GC Content Across HSV-2 UL52 Gene", fontsize=14)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

```



[ ]:

[5]: *# Full code to handle the given sequence and fill missing nucleotides to reach a specific size.*

```
def identify_and_fill_gaps(sequence, expected_length):
    """
    Identifies gaps in the sequence based on missing nucleotides and fills them
    symmetrically.

    Args:
        - sequence (str): The original nucleotide sequence.
        - expected_length (int): The expected length of the sequence after filling
        gaps.

    Returns:
        - filled_sequence (str): The sequence after filling the missing nucleotides.
        - gaps (list): List of identified gaps (positions and sizes).
    """
    original_length = len(sequence)
    missing_length = expected_length - original_length

    if missing_length <= 0:
        return sequence, []
```

```

# Step 1: Analyze sequence for gaps (quantum folding rule: gaps where
↳ difference > 1)
gaps = []
for i in range(1, len(sequence)):
    if ord(sequence[i]) - ord(sequence[i - 1]) > 1: # Gap detected
        gaps.append((i, ord(sequence[i]) - ord(sequence[i - 1]) - 1))

# Step 2: Redistribute missing nucleotides symmetrically into the sequence
filled_sequence = list(sequence)
remaining_to_fill = missing_length

for gap_index, gap_size in gaps:
    # Determine how many bases to fill in this gap
    fill_size = min(gap_size, remaining_to_fill)
    filler = 'N' * fill_size # Use 'N' for missing nucleotides
    remaining_to_fill -= fill_size

    # Insert missing nucleotides symmetrically
    filled_sequence.insert(gap_index, filler)

    if remaining_to_fill <= 0:
        break

# If there's still missing data after filling gaps, pad the end
if remaining_to_fill > 0:
    filled_sequence.append('N' * remaining_to_fill)

# Step 3: Combine into a complete sequence
filled_sequence = ''.join(filled_sequence)
return filled_sequence, gaps

# Full Input Sequence
original_sequence = (
    "ATGGGGACGGAAGACTGCGATCACGAAGGGCGGTGCGTTGCGGCTCCCGTGAGGTTACGGCGCTGTATG"
    "CGACCGACGGGTGCGTTATCACCTCCTCGCTCGCCCTCCTCACAACTGCCTGCTGGGGGCCGAGCCGTT"
    "GTATATATTCAGCTACGACGCGTACCGGCCGATGCGCCCAATGGCCCCACGGGCGCGCCACCGAACAG"
    "GAGAGGTTTCGAGGGGAGCCGGGCGCTCTACCGGATGCGGGGGGGCTAAATGGCGATTCAATTCGGGTGA"
    "CCTTTTGTATTATTGGGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT"
    "GTGCCGCTTCGAGCGAGCGGACGACGTCGCCGTGCTCCAAGACGCCCTGGGCCGCGGGACCCATTGCTC"
    "CCGCCCCACATCACAGCAACTCTGGAATTGGAGGCGACGTTTGGCTCCACGCTAACATCATCATGGCTC"
    "TCACCGTGGCCATCGTCCACAACGCCCCCGCCGATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCC"
    "CGGCGAATCGATGCGCTCGTTCGTCGGGCGCATGTCCCTGGGGCAGCGCGGCCTACCCACGCTGTTCTGTG"
    "CACACGAGGCGCGCGTGTGGCGGCGTACCGCCGGGCGTATTATGGGAGCGCCCAAAGCCCCCTTTGGT"
    "TTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGCTGGCCGCTAGGTACTACCTACTCCAGGCTCC"
    "GCGCTTGGGGGGCGCCGGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATC"
    "CCCCACGACCCACGCCCGACACCCTCAGTGCCGCGTCCTTGACCTCGTTGCGCGCCATCACTCGGTTCT"

```

```

"GTTGCACGAGCCAGTACTCCCGCGGGGCCGCGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGC"
"CGCCGACGTACGCGAGACCGGCGCGCTGGAGAAGTTCATCGCCACGATCGCAGCTGCCTGCGCGTGTCC"
"GACCGGAATTCATTACGTACATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCCGCGCCTGGCCACGC"
"ATCTCCGGGCGGTGACCACCCACGACCCAGCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCCCTGGGTCT"
"GGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAAC"
"GTCACCCCCAGGGAACCGCCCTGGCGGGAGACGCGGCCGCCCTACCTGCGCGCGCGCACGTATGCCC"
"CGGCGGCCCTCACGCCCCCCCCGCGTACTGCGGGTTCGAGACTCGTCCACCAAAATGATGGGACGTCT"
"GGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCGGTTTCGCACCAACAACCCCCGGGACGAC"
"GCGGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGG"
"AGCAGCAGGGCACGACGCCCCGCGATCGCGGCTCTCATGCAGACGCGTCGGTCCAAACCCCTGCC"
"CGTGTACAGGATTACCATGTCCCCGACCGGCCAGGCGTTTGCCGCGGCGGCGGGACGACTGGGCCCCG"
"GTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTCGCGGACGCGGCGGCGGCGCCGAGCCCGCG"
"CGCTCGGCGGCGGCTCACGCGCCGATTTGCGCCCGGGGCCCCGCGCTCCCCCGGGCGGCGCTGGCCGT"
"CGGGGGCCAGATGTACGTGAACCGCAACGAGATCTTCAACGCCGCGTGGCCGTTACGAACATCATCTG"
"GATCTGGACATCGCCCTGAAGGAGCCCGTCCCCTTTCCCGGCTCCACGAGGCCCTGGGTCACTTTAGGC"
"GCGGGGCGCTGGCGGCGGTTTACGTGTTGTTTCCCGCGGCCCGCTAGACCCCGACGCCTATCCCTGTTA"
"TTTTTTCAAAGCGCCTGTGCGCCCCGCGCGCCGCGCCGCTGTGCGGGCGACGGGCCCTGGCCGTGGC"
"GACGACGGCGACGGGACTGGTTCCCCGACGCCGTTGGTCCCGCGACGAGGAGTGGGAGGAGACACGG"
"ACCCCATGGACACGACCCACGGCCCCCTCCCGACGACGAGGCCGCTACCTCGACCTGCTACACGAACA"
"GATACCAGCGGCGACGCCAGCGAACCAGGACTCCGTCGTGTGTTCTGCGCCGACAAGATCGGGCTGCGC"
"GTGTGCCTACCGGTCCCCGCCCCGTACGTTGTGCACGGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGA"
"TCCAGCAGGCGGTGCTGTTGACCGCGACTTCGTGGAGGCCGTAGGAGCCACGTAAAGAACTTTTGCT"
"GATCGATACGGCGTGTACGCCACGGCCACAGCCTGCGCTTGCCGTATTTGCGCAAGATCGGCCCCGAC"
"GGCTCCGCGTGCGGCCGTTATTGCCCGTCTTCGTGATCCCCCGCGTGCGAGGACGTTCCGGCGTTCG"
"TCGCCGCGCACGCCGACCCGCGGCGCTTCCACTTTCACGCCCCGCCATGTTTTCCGCGGCCCGCGGGA"
"GATCCGCGTCTCCACAGCCTGGGCGGGACTATGTCAGCTTTTCGAGAAGAAGCGTCGCGCAACGCC"
"CTGGAGCACTTTGGGCGACGCGAGACCCTGACGGAGTTCTGGGCCGCTACGATGTGCGGCCGACGCCG"
"GGGAGACCGTGGAGGGGTTCCGCTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCCACTTCC"
"CGAGCACGCGCGGGAATATCAGGCCGTGTCGTTTCGCCGGGCCGTCATTAAGGACGACTGGGTCTGCTG"
"CAGCTGATCCCCGGCCGCGCGCCCTGAACCAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAA"
"GTCGCGGACGCGCCCGACCTTTCTCGCGTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTG"
"TCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATGCGGGCAG"
"CCATGCTCGCGTCCGCTCCCTCCAGCACCTCACGACCGTCATCTTCATAA"

```

```

)
expected_sequence_length = 4096

# Run the function
filled_sequence, detected_gaps = identify_and_fill_gaps(original_sequence,
↳ expected_sequence_length)

# Output results
print("Original Sequence Length:", len(original_sequence))
print("Expected Sequence Length:", expected_sequence_length)
print("Filled Sequence Length:", len(filled_sequence))
print("Detected Gaps:", detected_gaps)
print("Filled Sequence (First 500 bases):", filled_sequence)

```

Original Sequence Length: 3201

Expected Sequence Length: 4096

Filled Sequence Length: 4096

Detected Gaps: [(1, 18), (7, 1), (8, 3), (12, 5), (14, 1), (15, 16), (18, 3), (20, 18), (23, 1), (24, 3), (27, 5), (31, 3), (33, 12), (35, 3), (37, 12), (41, 3), (44, 16), (48, 3), (49, 12), (53, 5), (55, 12), (58, 1), (59, 3), (62, 3), (64, 16), (66, 12), (68, 18), (71, 3), (73, 1), (75, 3), (77, 1), (78, 3), (81, 12), (84, 3), (85, 12), (88, 18), (91, 1), (93, 16), (96, 16), (98, 3), (100, 16), (102, 3), (106, 16), (109, 16), (112, 1), (116, 1), (117, 16), (121, 16), (124, 16), (132, 3), (134, 5), (137, 3), (138, 12), (141, 12), (143, 18), (145, 18), (147, 18), (151, 5), (153, 16), (155, 1), (156, 3), (158, 1), (159, 3), (161, 3), (162, 12), (164, 1), (166, 3), (171, 3), (173, 18), (176, 3), (182, 18), (190, 1), (191, 3), (195, 3), (197, 3), (202, 1), (204, 3), (207, 1), (209, 5), (212, 5), (214, 5), (216, 12), (219, 3), (221, 5), (226, 5), (229, 3), (233, 3), (235, 16), (237, 16), (239, 1), (241, 3), (245, 18), (248, 3), (256, 16), (260, 18), (264, 3), (266, 18), (270, 18), (274, 3), (277, 12), (280, 1), (282, 16), (287, 12), (291, 18), (298, 1), (299, 3), (303, 5), (304, 12), (309, 3), (310, 12), (313, 1), (317, 1), (320, 1), (323, 3), (327, 5), (331, 3), (334, 1), (337, 3), (343, 18), (345, 12), (349, 12), (351, 12), (355, 3), (357, 16), (360, 3), (362, 5), (364, 3), (366, 5), (368, 3), (371, 1), (372, 3), (374, 1), (375, 3), (376, 12), (378, 3), (381, 3), (382, 12), (385, 16), (390, 5), (392, 1), (393, 3), (397, 16), (403, 3), (405, 3), (409, 1), (414, 18), (418, 16), (422, 3), (428, 1), (430, 18), (433, 1), (435, 5), (439, 1), (440, 16), (442, 16), (446, 1), (447, 16), (452, 5), (455, 3), (457, 1), (458, 3), (459, 12), (464, 3), (466, 16), (470, 1), (471, 3), (473, 16), (476, 1), (478, 18), (481, 18), (484, 18), (488, 16), (490, 16), (493, 1), (495, 3), (496, 12), (502, 18), (504, 3), (505, 12), (509, 1), (512, 1), (513, 3), (519, 3), (523, 3), (526, 18), (528, 3), (532, 5), (534, 3), (538, 5), (541, 1), (543, 3), (550, 16), (552, 12), (554, 18), (557, 5), (561, 3), (564, 3), (567, 18), (569, 3), (571, 18), (574, 3), (576, 16), (578, 3), (580, 12), (582, 3), (583, 12), (585, 3), (589, 3), (592, 18), (594, 12), (598, 16), (605, 5), (607, 3), (609, 3), (613, 16), (616, 1), (619, 1), (620, 3), (622, 16), (624, 12), (627, 3), (628, 12), (632, 1), (635, 1), (636, 3), (638, 5), (641, 3), (643, 3), (645, 3), (646, 12), (649, 16), (653, 3), (656, 3), (657, 12), (659, 1), (661, 3), (664, 3), (668, 3), (669, 12), (671, 18), (674, 18), (679, 5), (681, 3), (688, 5), (693, 16), (699, 12), (703, 16), (706, 5), (711, 18), (714, 3), (719, 3), (722, 1), (723, 3), (728, 5), (730, 5), (733, 16), (736, 12), (739, 16), (744, 3), (746, 16), (748, 5), (750, 12), (752, 1), (753, 16), (755, 1), (757, 16), (759, 1), (760, 16), (764, 5), (767, 16), (770, 3), (772, 3), (774, 16), (783, 3), (786, 3), (789, 5), (793, 1), (794, 3), (795, 12), (797, 1), (798, 3), (800, 18), (802, 16), (806, 5), (810, 3), (811, 12), (816, 5), (818, 1), (820, 18), (822, 16), (825, 3), (827, 3), (829, 1), (831, 16), (833, 1), (834, 3), (836, 3), (838, 18), (845, 1), (846, 3), (848, 1), (852, 1), (853, 3), (858, 3), (860, 1), (862, 1), (865, 16), (868, 5), (869, 12), (873, 3), (875, 3), (876, 12), (879, 16), (883, 1), (885, 16), (887, 3), (888, 12), (891, 3), (894, 3), (898, 18), (901, 1), (902, 16), (904, 3), (906, 12), (909, 16), (911, 12), (916, 1), (917, 3), (919, 5), (923, 5), (924, 12), (926, 1), (927, 16), (931, 3), (933, 3), (939, 3), (941, 3), (945, 3), (947, 16), (951, 12), (956, 3), (958, 16), (960, 12), (962, 18), (964, 12), (968, 5), (970, 3), (973, 3), (976, 18), (978, 3), (981, 3),



(984, 3), (986, 1), (987, 3), (988, 12), (990, 1), (991, 3), (993, 3), (995, 5),  
 (997, 1), (999, 3), (1002, 3), (1004, 3), (1006, 16), (1010, 5), (1013, 5),  
 (1014, 12), (1018, 18), (1020, 3), (1025, 1), (1026, 3), (1028, 18), (1030, 3),  
 (1033, 5), (1035, 16), (1039, 16), (1042, 3), (1044, 3), (1045, 12), (1047, 12),  
 (1050, 3), (1052, 1), (1054, 3), (1059, 18), (1063, 18), (1066, 1), (1067, 3),  
 (1068, 12), (1070, 1), (1072, 18), (1074, 16), (1076, 1), (1078, 16), (1085, 1),  
 (1086, 16), (1091, 5), (1092, 12), (1095, 16), (1099, 5), (1106, 3), (1108, 3),  
 (1111, 16), (1117, 1), (1118, 3), (1121, 18), (1123, 16), (1126, 3), (1131, 3),  
 (1132, 12), (1135, 1), (1138, 1), (1142, 1), (1143, 3), (1145, 1), (1150, 5),  
 (1155, 3), (1157, 3), (1162, 5), (1165, 1), (1166, 3), (1169, 5), (1172, 5),  
 (1176, 16), (1178, 3), (1183, 16), (1187, 12), (1189, 3), (1193, 5), (1196, 3),  
 (1198, 12), (1203, 1), (1205, 5), (1206, 12), (1209, 16), (1213, 3), (1217, 1),  
 (1218, 3), (1219, 12), (1222, 3), (1224, 3), (1229, 5), (1231, 16), (1235, 1),  
 (1237, 18), (1240, 3), (1242, 3), (1244, 5), (1245, 12), (1247, 1), (1248, 3),  
 (1249, 12), (1253, 5), (1259, 1), (1260, 3), (1261, 12), (1264, 1), (1270, 5),  
 (1276, 1), (1278, 3), (1282, 16), (1286, 3), (1290, 5), (1292, 1), (1293, 3),  
 (1295, 3), (1299, 3), (1302, 3), (1305, 16), (1307, 1), (1309, 16), (1312, 3),  
 (1314, 3), (1316, 3), (1318, 3), (1321, 1), (1322, 3), (1323, 12), (1325, 18),  
 (1331, 3), (1334, 3), (1339, 16), (1342, 1), (1343, 3), (1347, 3), (1353, 3),  
 (1355, 3), (1356, 12), (1358, 1), (1359, 16), (1362, 3), (1366, 12), (1368, 3),  
 (1371, 5), (1373, 1), (1374, 16), (1376, 3), (1377, 12), (1381, 1), (1387, 18),  
 (1390, 18), (1395, 1), (1396, 3), (1397, 12), (1399, 16), (1403, 3), (1407, 5),  
 (1410, 5), (1414, 5), (1417, 16), (1420, 16), (1422, 5), (1423, 12), (1430, 1),  
 (1431, 3), (1434, 16), (1440, 3), (1442, 3), (1443, 12), (1446, 3), (1449, 1),  
 (1453, 1), (1456, 1), (1461, 3), (1466, 1), (1467, 3), (1469, 1), (1470, 3),  
 (1472, 3), (1479, 3), (1483, 1), (1484, 16), (1488, 3), (1496, 5), (1498, 1),  
 (1500, 16), (1503, 3), (1507, 18), (1509, 3), (1510, 12), (1514, 5), (1516, 3),  
 (1519, 16), (1522, 16), (1526, 5), (1528, 16), (1533, 3), (1537, 1), (1538, 3),  
 (1541, 5), (1544, 5), (1547, 5), (1552, 1), (1553, 3), (1555, 1), (1556, 3),  
 (1562, 3), (1565, 3), (1567, 18), (1569, 3), (1571, 3), (1574, 16), (1576, 16),  
 (1579, 18), (1583, 5), (1586, 1), (1587, 3), (1589, 3), (1590, 12), (1592, 3),  
 (1594, 12), (1600, 1), (1606, 16), (1611, 3), (1612, 12), (1614, 12), (1616, 1),  
 (1618, 5), (1621, 18), (1624, 1), (1627, 18), (1629, 12), (1634, 3), (1636, 1),  
 (1638, 3), (1643, 5), (1646, 3), (1647, 12), (1653, 3), (1655, 3), (1658, 3),  
 (1661, 3), (1663, 3), (1667, 1), (1668, 3), (1670, 1), (1671, 16), (1678, 3),  
 (1680, 3), (1681, 12), (1684, 1), (1685, 3), (1687, 3), (1691, 1), (1692, 3),  
 (1694, 3), (1696, 3), (1700, 3), (1703, 3), (1707, 5), (1709, 3), (1711, 1),  
 (1713, 3), (1714, 12), (1717, 12), (1719, 3), (1721, 3), (1724, 1), (1725, 3),  
 (1727, 3), (1730, 3), (1733, 3), (1736, 3), (1740, 3), (1742, 5), (1746, 3),  
 (1749, 3), (1751, 3), (1753, 16), (1755, 3), (1759, 3), (1762, 3), (1765, 16),  
 (1768, 1), (1769, 3), (1771, 3), (1774, 3), (1777, 18), (1782, 3), (1786, 3),  
 (1794, 3), (1796, 3), (1798, 16), (1805, 3), (1809, 3), (1813, 16), (1818, 3),  
 (1819, 12), (1821, 3), (1829, 5), (1831, 18), (1833, 12), (1835, 1), (1836, 3),  
 (1837, 12), (1841, 1), (1843, 3), (1847, 1), (1848, 3), (1850, 5), (1852, 18),  
 (1854, 16), (1859, 1), (1860, 3), (1863, 3), (1865, 3), (1867, 16), (1872, 3),  
 (1873, 12), (1876, 1), (1877, 3), (1880, 1), (1882, 18), (1885, 18), (1888, 16),  
 (1892, 18), (1894, 16), (1898, 1), (1900, 18), (1902, 3), (1906, 16), (1910, 5),  
 (1913, 5), (1917, 3), (1918, 12), (1923, 16), (1930, 3), (1933, 16), (1937, 1),

(1938, 3), (1940, 5), (1945, 16), (1949, 12), (1952, 1), (1953, 16), (1957, 5),  
 (1960, 3), (1962, 3), (1967, 3), (1969, 16), (1973, 3), (1976, 3), (1978, 12),  
 (1982, 5), (1984, 16), (1986, 12), (1989, 12), (1995, 3), (1997, 3), (2002, 3),  
 (2004, 3), (2005, 12), (2007, 5), (2009, 1), (2013, 3), (2015, 1), (2016, 3),  
 (2019, 16), (2021, 18), (2025, 16), (2027, 12), (2030, 18), (2041, 5), (2043,  
 3), (2046, 16), (2048, 12), (2050, 3), (2056, 3), (2058, 3), (2060, 3), (2063,  
 3), (2067, 3), (2068, 12), (2070, 16), (2072, 12), (2075, 3), (2079, 3), (2081,  
 1), (2082, 3), (2089, 16), (2094, 3), (2096, 12), (2100, 3), (2102, 1), (2103,  
 3), (2105, 1), (2106, 3), (2109, 3), (2111, 1), (2112, 3), (2117, 1), (2118,  
 16), (2121, 12), (2127, 3), (2129, 1), (2130, 3), (2133, 3), (2135, 12), (2138,  
 12), (2142, 3), (2145, 3), (2147, 1), (2148, 3), (2150, 5), (2153, 5), (2154,  
 12), (2159, 5), (2162, 5), (2165, 1), (2167, 1), (2168, 3), (2171, 1), (2176,  
 18), (2180, 1), (2182, 1), (2183, 3), (2185, 1), (2189, 1), (2190, 3), (2197,  
 16), (2201, 3), (2204, 1), (2205, 3), (2207, 1), (2208, 3), (2210, 5), (2214,  
 3), (2216, 3), (2217, 12), (2219, 1), (2221, 16), (2223, 3), (2225, 1), (2227,  
 16), (2230, 16), (2232, 1), (2234, 1), (2235, 3), (2238, 1), (2240, 5), (2242,  
 18), (2244, 1), (2247, 5), (2249, 3), (2252, 3), (2254, 1), (2255, 3), (2260,  
 5), (2262, 3), (2265, 1), (2267, 3), (2270, 1), (2271, 16), (2274, 3), (2275,  
 12), (2277, 3), (2278, 12), (2280, 12), (2282, 12), (2286, 16), (2289, 3),  
 (2292, 3), (2294, 1), (2297, 5), (2299, 18), (2301, 3), (2305, 16), (2308, 3),  
 (2310, 3), (2311, 12), (2313, 12), (2317, 16), (2319, 1), (2321, 3), (2323, 12),  
 (2328, 3), (2333, 3), (2334, 12), (2336, 1), (2337, 3), (2338, 12), (2341, 12),  
 (2345, 1), (2346, 3), (2349, 16), (2353, 16), (2356, 1), (2357, 3), (2359, 18),  
 (2362, 3), (2363, 12), (2368, 12), (2372, 3), (2374, 5), (2377, 12), (2380, 18),  
 (2384, 5), (2387, 5), (2390, 3), (2392, 12), (2395, 16), (2397, 12), (2402, 1),  
 (2404, 3), (2406, 3), (2408, 1), (2409, 16), (2412, 3), (2413, 12), (2417, 5),  
 (2421, 3), (2422, 12), (2424, 5), (2428, 5), (2432, 1), (2433, 3), (2434, 12),  
 (2438, 5), (2441, 1), (2442, 16), (2449, 16), (2452, 18), (2454, 3), (2456, 18),  
 (2458, 1), (2459, 3), (2463, 3), (2464, 12), (2466, 12), (2468, 1), (2469, 3),  
 (2474, 1), (2475, 3), (2480, 1), (2482, 5), (2485, 16), (2488, 3), (2490, 16),  
 (2495, 3), (2496, 12), (2498, 18), (2502, 3), (2507, 5), (2509, 18), (2511, 3),  
 (2517, 3), (2519, 1), (2520, 3), (2523, 16), (2526, 3), (2528, 3), (2529, 12),  
 (2532, 3), (2536, 3), (2538, 12), (2541, 18), (2547, 3), (2548, 12), (2550, 16),  
 (2553, 3), (2554, 12), (2557, 18), (2565, 3), (2567, 3), (2568, 12), (2571, 3),  
 (2573, 5), (2576, 1), (2577, 3), (2578, 12), (2582, 3), (2585, 3), (2586, 12),  
 (2589, 3), (2590, 12), (2592, 3), (2595, 3), (2597, 3), (2600, 1), (2601, 3),  
 (2604, 3), (2606, 1), (2609, 3), (2611, 3), (2614, 3), (2616, 16), (2621, 1),  
 (2622, 16), (2627, 1), (2628, 3), (2633, 3), (2638, 18), (2640, 12), (2646, 3),  
 (2648, 3), (2654, 3), (2656, 3), (2660, 5), (2662, 18), (2665, 3), (2667, 3),  
 (2668, 12), (2671, 16), (2675, 1), (2677, 5), (2680, 16), (2685, 3), (2690, 1),  
 (2691, 16), (2693, 18), (2695, 12), (2698, 5), (2700, 16), (2706, 3), (2708, 5),  
 (2711, 5), (2714, 5), (2717, 3), (2718, 12), (2720, 3), (2722, 3), (2726, 1),  
 (2727, 3), (2731, 16), (2735, 5), (2738, 1), (2739, 16), (2746, 3), (2748, 1),  
 (2749, 3), (2751, 3), (2753, 5), (2755, 1), (2758, 16), (2761, 1), (2762, 3),  
 (2765, 5), (2767, 12), (2770, 16), (2776, 3), (2778, 16), (2780, 1), (2781, 3),  
 (2783, 18), (2785, 12), (2788, 3), (2793, 3), (2795, 1), (2796, 3), (2799, 3),  
 (2804, 5), (2806, 1), (2808, 3), (2809, 12), (2813, 5), (2817, 12), (2820, 3),  
 (2822, 3), (2823, 12), (2826, 5), (2829, 1), (2830, 16), (2833, 16), (2839, 3),

(2842, 18), (2844, 5), (2845, 12), (2847, 3), (2849, 3), (2850, 12), (2854, 18), (2856, 3), (2858, 5), (2864, 1), (2865, 16), (2871, 3), (2873, 5), (2876, 1), (2877, 3), (2879, 3), (2881, 3), (2886, 18), (2888, 18), (2891, 5), (2895, 3), (2896, 12), (2898, 12), (2901, 3), (2902, 12), (2905, 3), (2908, 3), (2913, 3), (2914, 12), (2917, 18), (2921, 5), (2924, 1), (2925, 3), (2927, 1), (2928, 16), (2932, 12), (2935, 16), (2938, 16), (2942, 5), (2944, 16), (2947, 18), (2952, 3), (2956, 3), (2958, 3), (2961, 3), (2965, 16), (2969, 1), (2974, 5), (2977, 16), (2979, 16), (2981, 3), (2982, 12), (2984, 12), (2986, 16), (2989, 3), (2991, 16), (2996, 5), (2999, 1), (3000, 3), (3004, 5), (3010, 5), (3011, 12), (3013, 3), (3015, 3), (3017, 3), (3019, 1), (3020, 3), (3025, 3), (3028, 1), (3030, 16), (3034, 16), (3036, 3), (3038, 3), (3040, 16), (3043, 5), (3045, 3), (3046, 12), (3048, 3), (3052, 1), (3056, 1), (3059, 1), (3061, 3), (3064, 16), (3066, 18), (3069, 3), (3071, 3), (3072, 12), (3076, 16), (3078, 12), (3080, 12), (3083, 5), (3086, 5), (3087, 12), (3090, 16), (3097, 1), (3098, 16), (3102, 18), (3105, 3), (3107, 18), (3110, 1), (3113, 1), (3115, 3), (3118, 16), (3122, 1), (3124, 1), (3125, 3), (3127, 16), (3129, 12), (3133, 1), (3135, 3), (3136, 12), (3138, 3), (3140, 18), (3143, 3), (3148, 1), (3149, 3), (3153, 18), (3156, 16), (3158, 3), (3160, 3), (3162, 12), (3165, 3), (3167, 16), (3171, 16), (3175, 5), (3178, 1), (3180, 16), (3183, 1), (3184, 3), (3186, 1), (3188, 3), (3189, 12), (3192, 18), (3194, 16), (3198, 18)]

Filled Sequence (First 500 bases): ANNNNNNNNNNNNNNNNNNTGGGGNNNNACGNNNNNGNNNNNNNN  
NNNNNNNNNAANNNGNNNNNNNNNNNNNNNNNACNNNTGNNNNNCGANNNTNNNNNNNNNNNNCNNNANNNNNNNNN  
NNCGANNNAGNNNNNNNNNNNNNNNGGCNNNNNNNNNNNNNNNGGTNNNNNCNNNNNNNNNNNNNGGNNNTTNNNGNN  
NNNNNNNNNNNNNCNNNNNNNNNNNNNGNNNNNNNNNNNNNNNNNGCNNNTNCNNNCNNNNCGNNNNNNNNNNNTGN  
NNNNNNNNNNNGANNNNNNNNNNNNNNNNNNNGGNTNNNNNNNNNNNNNNNTANNNNNNNNNNNNNNNNC  
NNNNNNNNNNNGNNCGCNNNNNNNNNNNNNNNTGNNNNNNNNNNNNNNTANTGCNNNNNNNNNNNNNNNGACNN  
NNNNNNNNNNNNNCGNNNNNNNNNNNNNNNACGGGTGNNNCNNNNNGTNNNNNNNNNNNNNTANNNNNNNNNNNT  
NNNNNNNNNNNNNNNNNNCNNNNNNNNNNNNNNNNNANNNNNNNNNNNNNNNNNCCTNNNNNCNNNNNNNNNNNN  
NCNNNTTNNNNCNNNNNNNNNNNNNNNGNCNNNTCGCNNNCNNNNNNNNNNNNNNNNCTNNNCCTCANNNNNNNNNN  
NNNNNNNCAAACTGNNNNCCTNNNGNNCTGGNGNNNGGNCNNNNNCGNNNNANNNNGNNNNNNNNNNNCCNNNGNN  
NNNTTGTNNNNNATNNNATANNNTNNNNNNNNNNNNNNNTNNNNNNNNNNNNNNNCNANNNGCTNNNNNNNNNNNN  
NNNNACNNNGACGCGTNNNNNNNNNNNNNNNACNNNNNNNNNNNNNNNNNGGCNNNCNNNNNNNNNNNNNNNNNCG  
ANNNNNNNNNNNNNNNNTGCNNNGCNNNNNNNNNNNNCCNANNNNNNNNNNNNNNNATGGNNNNNNNNNNNCCCN  
NNNNNNNNNNNNNNNNCACGGGNNNNCGCNNNNNNNNNNNNNNNGCCNNNNNNNNNNNNNNNACNCGANACNAGNN  
GAGNNNNNAGNNNTTNCGNNNAGGGNNNNNNNAGCCGGGCGCTCTACCGGATGCGGGGGGCTAAATGGCGATTCA  
TTTCGGGTGACCTTTTGTATTGTTGAGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT  
GTGCCGCTTCGAGCGAGCGGACGACGTGCGCGTGCTCCAAGACGCCCTGGGCCGCGGACCCCATGCTCCCGGCCACA  
TCACAGCAACTCTGGAATTGGAGGCGACGTTTGCCTCCACGCTAACATCATATGGCTCTCACCGTGGCCATCGTCCAC  
AACGCCCCCGCCGCATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCCCGGCGAATCGATGCGCTCGGTCTCGGGCG  
CATGTCCCTGGGGCAGCGGGCCTCACCACGCTGTTCTGTGACCACGAGGCGCGCTGCTGGCGGCGTACCGCCGGCGT  
ATTATGGGAGCGCCAAAGCCCTTTTGGTTTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGTGCGCGTAGG  
TACTACCTACTCCAGGCTCCGCGCTTGGGGGGCGCCGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGAC  
CTACGCGATCCCCACGACCCACGCCCCGACCCCTCAGTGCCGCGTCTTGACCTCGTTCGCCGCCATCACTCGTTCT  
GTTGCACGAGCCAGTACTCCCGCGGGGCGCGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGCCGCCGACGTA  
CGCGAGACCGCGCGCTGGAGAAGTTCATCGCCACGATCGCAGCTGCCTGCGCGTGTCCGACCGGAATTCATTACGTA  
CATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCCGCGCCTGGCCACGCATCTCCGGGCGGTGACCACCCACGACCCCA  
GCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCTGGGTGCGGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAG  
CTGAACATCCGCGAGTACGTAAAGCAAAACGTACCCCCAGGGAAACCGCCCTGGCGGGAGACGCGGCCGCGCCTACCT

```

GCGCGCGCGCACGTATGCCCCGGCGGCCCTCACGCCCGCCCCGCGTACTGCGGGGTGCGAGACTCGTCCACCAAAATGA
TGGGACGTCTGGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCCGCGTTTCGCACCAACAACCCCCGGGGACGAC
GCGGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGGAGCAGCAGGG
CACGACGCCCCCGGCGATCGCGGCTCTCATGCAGGACGCGTCGGTCCAAACCCCCCTGCCCGTGTACAGGATTACCATGT
CCCCGACCGGCCAGGCGTTTGCCGCGGCGGCGCGGGACGACTGGGCCCGCGTGACGCGGGACGCGCGCCCCGCCGGAAGCG
ACCGTGTCGCGGACGCGGCGGCGGCGCCCCGAGCCCGGCGCGCTCGGCCGCGGCTACGCGCCGCATTTGCGCCCCGGG
CCCCGCGCTCCCCCGGGCGGCCTGGCCGTGCGGGGCCAGATGTACGTGAACCGCAACGAGATCTTCAACGCCGCGCTGG
CCGTTACGAACATCATCCTGGATCTGGACATCGCCCTGAAGGAGCCCGTCCCCCTTCCCCGGCTCCACGAGGCCCTGGGT
CACTTTAGCGCGGGGCGCTGGCGGCGGTTACAGTGTGTTTCCGCGGCCCGCGTAGACCCCCGACGCTATCCCTGTTA
TTTTTTCAAAAGCGCCTGTGCGCCCCGCGCGCCCGCTCTGTGCGGGCGACGGGCCCTGGCCGGTGGCGACGACGGCG
ACGGGGACTGGTTCCCCGACGCCGGTGGTCCCGGCGACGAGGAGTGGGAGGAGACACGGACCCCATGGACACGACCCAC
GGCCCCCTCCCGGACGACGAGGCCGCGTACCTCGACCTGCTACACGAACAGATACCAGCGGCGACGCCAGCGAACCGGA
CTCCGTCGTGTGTTCTGCGCCGACAAGATCGGGCTGCGCGTGTGCCTACCGTCCCCGCCCCGTACGTTGTGCACGGCT
CCCTGACGATGCGTGGGGTGGCGAGGTTGATCCAGCAGGCGGTGCTGTTGGACCGCGACTTCGTGGAGGCCGTAGGGAGC
CACGTAAAGAACTTTTTGCTGATCGATACGGGCGTGACGCCCACGGCCACAGCCTGCGCTTGCCGTATTTTCGCCAAGAT
CGGCCCCGACGGCTCCGCGTGCGGCCGTTATTGCCCGTCTTCGTGATCCCCCGCGTGCGAGGACGTTCCGGCGTTCG
TCGCCGCGCACGCCGACCCGCGGCGCTTCCACTTTCACGCCCCGCCATGTTTTCCGCGGCCCGCGGGAGATCCGCGTC
CTCCACAGCCTGGGCGGGGACTATGTCAGCTTTTTTCGAGAAGAAGGCGTCGCGCAACGCCCTGGAGCACTTTGGGCGACG
CGAGACCCTGACGGAGGTTCTGGGCCGCTACGATGTGCGGCCCGACGCCGGGAGACCGTGGAGGGGTTCCGCTCAGAAC
TGCTGGGGCGAATAGTCGCGTGCATCGAGGCCACTTTCCCGAGCACGCGCGGAATATCAGGCCGTGTCCGTTCCGCCG
GCCGTCAATTAAGGACGACTGGGTCTGTGTCAGCTGATCCCCGGCCGCGGCGCCCTGAACCAAAGCCTCTCGTGTCTGCG
CTTCAAGCACGGCAGGGCAAGTCGCGGACGCGCCGGACCTTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCG
CGTCCCTGTGTCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATGCGGGCACG
CCATGCTCGCGGTCCGCTCCCTCCAGCACCTCACGACCGTCATCTTCATAA

```

```

[7]: # Function to identify and fill gaps, highlighting added nucleotides
def identify_and_fill_gaps_with_highlight(sequence, expected_length):
    """
    Identifies gaps in the sequence based on missing nucleotides, fills them
    ↪symmetrically,
    and highlights the added nucleotides.

    Args:
    - sequence (str): The original nucleotide sequence.
    - expected_length (int): The expected length of the sequence after filling
    ↪gaps.

    Returns:
    - filled_sequence (str): The sequence after filling the missing nucleotides.
    - highlighted_sequence (str): The sequence with added nucleotides
    ↪highlighted.
    - total_pairs (int): Total number of pairs in the filled sequence.
    - added_pairs (int): Number of pairs formed by the inserted nucleotides.
    """
    original_length = len(sequence)
    missing_length = expected_length - original_length

```

```

if missing_length <= 0:
    return sequence, sequence, original_length // 2, 0

# Analyze sequence for gaps
gaps = []
for i in range(1, len(sequence)):
    if ord(sequence[i]) - ord(sequence[i - 1]) > 1: # Gap detected
        gaps.append((i, ord(sequence[i]) - ord(sequence[i - 1]) - 1))

# Redistribute missing nucleotides symmetrically into the sequence
filled_sequence = list(sequence)
highlighted_sequence = list(sequence) # For storing highlighted sequence
remaining_to_fill = missing_length

for gap_index, gap_size in gaps:
    fill_size = min(gap_size, remaining_to_fill)
    filler = 'N' * fill_size # Use 'N' for missing nucleotides
    remaining_to_fill -= fill_size

    # Insert missing nucleotides symmetrically
    filled_sequence.insert(gap_index, filler)
    highlighted_sequence.insert(gap_index, f"[{filler}]") # Highlight
    ↪ added bases

    if remaining_to_fill <= 0:
        break

# If there's still missing data after filling gaps, pad the end
if remaining_to_fill > 0:
    filler = 'N' * remaining_to_fill
    filled_sequence.append(filler)
    highlighted_sequence.append(f"[{filler}]") # Highlight added padding

# Combine into complete sequences
filled_sequence = ''.join(filled_sequence)
highlighted_sequence = ''.join(highlighted_sequence)

# Calculate pairs
total_pairs = len(filled_sequence) // 2
added_pairs = missing_length // 2

return filled_sequence, highlighted_sequence, total_pairs, added_pairs

# Full Input Sequence
original_sequence = (
    "ATGGGGACGGAAGACTGCGATCACGAAGGGCGGTCGGTTGCGGCTCCCGTGGAGGTTACGGCGCTGTATG"
    "CGACCGACGGGTGCGTTATCACCTCCTCGCTCGCCCTCCTCAGAACTGCCTGCTGGGGGCCGAGCCGTT"

```

```

"GTATATATTTCAGCTACGACGCGTACCGGCCCGATGCGCCCAATGGCCCCACGGGCGCGCCACCGAACAG"
"GAGAGGTTTCGAGGGGAGCCGGGCGCTCTACCGGGATGCGGGGGGGCTAAATGGCGATTTCATTCGGGTGA"
"CCTTTTGTATTGTTGGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT"
"GTGCCGCTTCGAGCGAGCGGACGACGTGCGCGTGCTCCAAGACGCCCTGGGCGCGGGACCCCATGTGCTC"
"CCGGCCACATCACAGCAACTCTGGACTTGGAGGCGACGTTTGCCTCCACGCTAACATCATCATGGCTC"
"TCACCGTGGCCATCGTCCACAACGCCCCCGCCGCATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCC"
"CGGCGAATCGATGCGCTCGTCTCGGGCGCATGTCCCTGGGGCAGCGCGGCTCACCACGCTGTTCTGTG"
"CACCACGAGGCGCGCTGCTGGCGGCGTACCGCCGGGCGTATTATGGGAGCGCCCAAAGCCCCCTTTTGGT"
"TTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGCTGGCCGCTAGGTACTACCTACTCCAGGCTCC"
"GCGCTTGGGGGGCGCCGGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATC"
"CCCCACGACCCACGCCCGACACCCTCAGTGCCGCGTCCCTTGACCTCGTTGCGCGCCATCACTCGGTTCT"
"GTTGCACGAGCCAGTACTCCCGCGGGGCGCGGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGC"
"CGCCGACGTACGCGAGACCGGCGCGCTGGAGAAGTTCATCGCCCACGATCGCAGCTGCCTGCGCGTGCTCC"
"GACCGGAATTCATTACGTACATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCCGCGCTGGCCACGC"
"ATCTCCGGGCGGTGACCACCCACGACCCAGCCCGCGGCCAGCACGAGCAGCCCTCGCCCCGTTGGTTCG"
"GGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAAC"
"GTCACCCCCAGGGAACCGCCCTGGCGGGAGACGCGGCCGCGCCCTACCTGCGCGCGCGCACGTATGCCC"
"CGGCGGCCCTCACGCCCGCCCCGCGTACTGCGGGTTCGAGACTCGTCCACCAAAATGATGGGACGTCT"
"GGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCGGTTTCGCACCAACAACCCCGGGGACGAC"
"GCGGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGG"
"AGCAGCAGGGCAGCAGCCCCCGCGATCGCGGCTCTCATGCAGGACGCGTCGGTCCAAACCCCTGCC"
"CGTGATACAGGATTACCATGTCCCCGACCGGCCAGGCGTTTGGCGGGCGGGCGGGGACGACTGGGCCCG"
"GTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTTCGCGGACGCGGCGGGCGGCCGAGCCCGCG"
"CGCTCGGCCGGCGGCTCACGCGCCGATTTGCGCCCGGGGCCCCGCGCTCCCCCGGGCGGCGCTGGCCGT"
"CGGGGGCCAGATGTACGTGAACCGCAACGAGATCTTCAACGCCGCGCTGGCCGTTACGAACATCATCCTG"
"GATCTGGACATCGCCCTGAAGGAGCCCGTCCCCTTTCCCGGCTCCACGAGGCCCTGGGTCACTTTAGGC"
"GCGGGGCGCTGGCGGCGGTTACGTGTTGTTTCCCGCGGCCCGGTAGACCCCGACGCCTATCCCTGTTA"
"TTTTTTCAAAGCGCCTGTGCGCCCCGCGCGCCCGCCGCTGTGTGCGGGCGACGGGCCCCCTGGCCGTGGC"
"GACGACGGCGACGGGACTGGTTCCCCGACGCCGTTGGTCCCAGGACGAGGAGTGGGAGGAGGACACGG"
"ACCCCATGGACACGACCCACGGCCCCCTCCCGACGACGAGGCCGCGTACCTCGACCTGCTACACGAACA"
"GATACCAGCGGCGACGCCAGCGAACCAGACTCCGTCGTGTGTTCTGCGCCGACAAGATCGGGCTGCGC"
"GTGTGCTACCGGTCCCCGCCCCGTACGTTGTGCACGGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGA"
"TCCAGCAGGCGGTGCTGTTGGACCGGACTTCGTGGAGGCCGTAGGGAGCCACGTAAAGAACTTTTGCT"
"GATCGATACGGGCGGTACGCCCACGGCCACAGCCTGCGCTTGCCGTATTTGCGCAAGATCGGCCCCGAC"
"GGCTCCGCGTGCGGCCGTTATTGCCCGTCTTCGTGATCCCCCGCGTGCGAGGACGTTCCGGCGTTTCG"
"TCGCGCGCACGCCGACCCGCGGCGCTTCCACTTTCACGCCCCGCCATGTTTTCCGCGGCCCGCGGGA"
"GATCCGCGTCTCCACAGCCTGGGCGGGGACTATGTGAGCTTTTTCGAGAAGAAGGCGTCGCGCAACGCC"
"CTGGAGCACTTTGGGCGACGCGAGACCCTGACGGAGTTCTGGGCCGCTACGATGTGCGGCCGACGCCG"
"GGGAGACCGTGGAGGGGTTTCGCGTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCCACTTTC"
"CGAGCACGCGCGGGAATATCAGGCCGTGTCCGTTCCCGGGCCGTCATTAAGGACGACTGGGTCTGCTG"
"CAGCTGATCCCCGGCCGCGGCCCTGAACCAAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAA"
"GTCGCGGACGCGCCCGACCTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTG"
"TCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATCGGGGACG"
"CCATGCTCGCGTCCGCTCCCTCCAGCACCTCACACCGTCATCTTCATAA")

```

```
expected_sequence_length = 4096
```

```

# Run the function
filled_sequence, highlighted_sequence, total_pairs, added_pairs = identify_and_fill_gaps_with_highlight(
    original_sequence, expected_sequence_length
)

# Display results
print("Original Sequence Length:", len(original_sequence))
print("Expected Sequence Length:", expected_sequence_length)
print("Filled Sequence Length:", len(filled_sequence))
print("Total Pairs in Filled Sequence:", total_pairs)
print("Added Pairs:", added_pairs)
print("\nHighlighted Sequence (First 500 bases):\n", highlighted_sequence)

# Optionally save the highlighted sequence to a file for further review
with open("highlighted_sequence.txt", "w") as file:
    file.write(highlighted_sequence)

```

Original Sequence Length: 3201  
 Expected Sequence Length: 4096  
 Filled Sequence Length: 4096  
 Total Pairs in Filled Sequence: 2048  
 Added Pairs: 447

Highlighted Sequence (First 500 bases):

A [NNNNNNNNNNNNNNNNNNNN] TGGGG [N] [NNN] ACG [NNNNN] G [N] [NNNNNNNNNNNNNNNNNN] AA [NNN] G [NNNN  
 NNNNNNNNNNNNNNNNNNN] AC [N] [NNN] TG [NNNNN] CGA [NNN] T [NNNNNNNNNNNNNNNNNN] C [NNN] A [NNNNNNNNNNNNNNNNNN] CG  
 A [NNN] AG [NNNNNNNNNNNNNNNNNN] GGC [NNN] [NNNNNNNNNNNNNNNNNN] GGT [NNNNN] C [NNNNNNNNNNNNNNNNNN] GG [N] [N  
 NN] TT [NNN] G [NNNNNNNNNNNNNNNNNN] C [NNNNNNNNNNNNNNNNNN] G [NNNNNNNNNNNNNNNNNNNN] GC [NNN] T [N] C [NN  
 N] C [N] [NNN] CG [NNNNNNNNNNNNNNNNNN] TG [NNN] [NNNNNNNNNNNNNNNNNN] GA [NNNNNNNNNNNNNNNNNNNN] GG [N] T [NNN  
 NNNNNNNNNNNNNNNNNNN] TA [NNNNNNNNNNNNNNNNNN] C [NNN] G [NNNNNNNNNNNNNNNNNN] G [NNN] CGC [NNNNNNNNNNNN  
 NNNNN] TG [NNNNNNNNNNNNNNNNNN] TA [N] TGC [N] [NNNNNNNNNNNNNNNNNN] GAC [NNNNNNNNNNNNNNNNNN] CG [N  
 NNNNNNNNNNNNNNNNNNN] ACGGGTG [NNN] C [NNNNN] GT [NNN] [NNNNNNNNNNNNNNNNNN] TA [NNNNNNNNNNNNNNNNNN] T [NNNNN  
 NNNNNNNNNNNNNNNNNNN] C [NNNNNNNNNNNNNNNNNNNN] A [NNNNNNNNNNNNNNNNNNNN] CCT [NNNNN] C [NNNNNNNNNNNNNNNNNN  
 NNNN] C [N] [NNN] T [N] [NNN] C [NNN] [NNNNNNNNNNNNNNNNNN] G [N] C [NNN] TCGC [NNN] C [NNNNNNNNNNNNNNNNNN  
 NN] CT [NNN] CCTCA [NNNNNNNNNNNNNNNNNNNN] CAAACTG [N] [NNN] CCT [NNN] G [NNN] CTGG [N] G [NNN] GG [N  
 N] C [NNNNN] CG [NNNNN] A [NNNNN] G [NNNNNNNNNNNNNNNNNN] CC [NNN] G [NNNNN] TTGT [NNNNN] AT [NNN] ATA [N  
 NN] T [NNNNNNNNNNNNNNNNNN] T [NNNNNNNNNNNNNNNNNN] C [N] A [NNN] GCT [NNNNNNNNNNNNNNNNNNNN] AC [NNN  
 ] GACGCGT [NNNNNNNNNNNNNNNNNNNN] ACC [NNNNNNNNNNNNNNNNNNNN] GGC [NNN] C [NNNNNNNNNNNNNNNNNNNN] CG  
 A [NNNNNNNNNNNNNNNNNNNN] TGC [NNN] GC [NNNNNNNNNNNNNNNNNN] CC [N] A [NNNNNNNNNNNNNNNNNNNN] ATGG [NNNNNN  
 NNNNN] CCC [NNNNNNNNNNNNNNNNNNNN] CACGGG [N] [NNN] CGC [NNNNN] [NNNNNNNNNNNNNNNNNN] GCCC [NNN] [NN  
 NNNNNNNNNNN] AC [N] CGA [N] AC [N] AG [NNN] GAG [NNNNN] AGG [NNN] TT [N] CG [NNN] AGGGG [NNNNNNNN] A  
 GCCGGGCGCTCTACCGGGATGCGGGGGGGCTAAATGGCGATTCTTTTCGGGTGACCTTTTGTATTGGGGACGGAAGTG  
 GGCGTGACCCACCAACCCGAAAGGGCGCACCCGGCCCATGTTTGTGTGCGGCTTCGAGCGAGCGGACGACGTCGCCGTGCT  
 CCAAGACGCCCTGGGCGCGGGACCCCATGTCTCCCGGCCACATCACAGCAACTCTGGACTTGAGGCGACGTTTGCGC  
 TCCACGCTAACATCATCATGGCTCTCACCGTGCCATCGTCCACAACGCCCCCGCCGCATCGGCAGCGGCAGCACCGCC  
 CCCCTGTATGAGCCCGGCGAATCGATGCGCTCGGTGCGTGGGCGCATGTCCCTGGGGCAGCGCGGCCTCACCACGCTGTT  
 CGTGACACGAGGCGCGCGTGCTGGCGGCGTACCGCCGGGCGTATTATGGGAGCGCCCAAAGCCCCTTTTGGTTTCTGA

GCAAATTCGGCCCGGACGAAAAGAGCCTGGTGTGGCCGCTAGGTACTACCTACTCCAGGCTCCGCGCTTGGGGGGCGCC  
GGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATCCCCACGACCCACGCCCCGACACCCT  
CAGTGCCGCGTCTTGACCTCGTTCCGCGCCATCACTCGGTTCTGTTGCACGAGCCAGTACTCCGCGGGGGCGCGCCG  
CTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGCCGCCGACGTACGCGAGACCGCGCGCTGGAGAAGTTCATCGCCAC  
GATCGCAGCTGCCTGCGCGTGTCCGACCGGGAATTCATTACGTACATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCC  
GCGCCTGGCCACGCATCTCCGGGCGGTGACCACCACGACCCAGCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCTGG  
GTCGGGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAACGTACC  
CCCAGGGAAACCGCCCTGGCGGGAGACGCGGCCGCCCTACCTGCGCGCGCGACGTATGCCCCGGCGGCCCTCACGCC  
CGCCCCCGCTACTGCGGGGTGCGAGACTCGTCCACCAAAATGATGGGACGTCTGGCGGAAGCAGAAAGGCTCCTAGTCC  
CCCACGGCTGGCCCGGTTGCGACCAACAACCCCGGGGACGACGCGGGGGCGGCACTGCCGCCCCCAGACCTGCGGA  
ATCGTCAAGCGCTCCTCAAGCTGGCCGCCACGAGCAGAGGACGACGCCCCCGCGATCGCGGCTCTCATGCAGGA  
CGCGTCGGTCCAAACCCCTGCCCCTGTACAGGATTACCATGTCCCGACCGGCCAGGCGTTTGCCGCGCGCGCGGG  
ACGACTGGGCCCCGCTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTCGCGGACGCGGGCGGCGCCCGAGCCC  
GGCGCGCTCGGCCGCGGCTCACGCGCCGATTGCGCCCCGGGGCCCCGCGCTCCCCCGGGCGGCGCTGGCCGTGGGGG  
CCAGATGTACGTGAACCGCAACGAGATCTTCAACGCGCGCTGGCCGTTACGAACATCATCCTGGATCTGGACATCGCC  
TGAAGGAGCCCGTCCCTTTCCCGGCTCCACGAGGCCCTGGGTCACTTTAGGCGGGGGCGCTGGCGGCGGTTACAGTG  
TTGTTTCCCGCGGCCCGCTAGACCCCGACGCTATCCCTGTTATTTTTTCAAAAGCGCCTGTGCGCCCCGCGCGCCGCC  
CGTCTGTGCGGGCGACGGGCCCCCTGGCCGTTGGCGACGACGGCGACGGGACTGGTTCCCCGACGCGGTTGGTCCCCGCG  
ACGAGGAGTGGGAGGAGACACGACCCCATGGACACGACCCACGGCCCCCTCCCGACGACGAGGCCGCGTACCTCGAC  
CTGCTACACGAACAGATACAGCGGCGACGCCAGCAACCGACTCCGTCGTGTGTTCTGCGCCGACAAGATCGGGCT  
GCGCGTGTGCTACCGGTCCCCGCCCCGTACGTTGTGACGCGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGATCCAGC  
AGGCGGTGCTGTTGGACCGCGACTTCGTGGAGCCGTAGGGAGCCACGTAAAGAACTTTTGTGATCGATACGGGCGTG  
TACGCCCACGGCCACAGCCTGCGCTTGCCGATTTCGCCAAGATCGGCCCGACGGCTCCGCGTGC GGCCGTTATTGCC  
CGTCTTCGTGATCCCCCGCGTGCAGGACGTTCCGGCGTTCGTGCGCGCGACGCCGACCCGCGGCGCTTCCACTTTC  
ACGCCCCGCCATGTTTTCCGCGGCCCGCGGGAGATCCGCGTCTCCACAGCCTGGGCGGGGACTATGTCAGCTTTTTC  
GAGAAGAAGGCGTCGCGCAACGCCCTGGAGCACTTTGGGCGACGCGAGACCCTGACGGAGGTTCTGGGCCGCTACGATGT  
GCGGCCCCGACCGGGGAGACCGTGGAGGGTTTCGCGTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCACT  
TTCCCGAGCACGCGCGGGAATATCAGGCCGTGTCGTTTCGCGGGCCGTCATTAAGGACGACTGGGTCTGCTGCAGCTG  
ATCCCCGGCGCGGCGCCCTGAACCAAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAAGTCGCGCGACGGCCG  
GACCTTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTGTGACGAGTGCTTTGCCACTAAATGCG  
ATAACAACCGCTGCACACGCTGTTTACCGTCGATGCGGGCACGCCATGCTCGCGGTCCGCTCCCTCCAGCACCTCACGA  
CCGTCATCTTCATAA

[37]: *# Refined function to address gaps properly and minimize padding while grouping  
→ the sequence into pairs.*

```
def identify_and_fill_gaps_properly(sequence, expected_length):
    """
    Refined version to detect gaps, fill them symmetrically, and minimize
    →padding,
    with the output grouped in pairs for readability.

    Args:
    - sequence (str): The original nucleotide sequence.
    - expected_length (int): The expected length of the sequence after filling
    →gaps.
```



```

Returns:
- grouped_sequence (str): The sequence grouped in pairs for readability.
- filled_sequence (str): The sequence after filling the missing nucleotides.
- total_pairs (int): Total number of pairs in the filled sequence.
- added_pairs (int): Number of pairs formed by the inserted nucleotides.
"""

original_length = len(sequence)
missing_length = expected_length - original_length

if missing_length <= 0:
    grouped_sequence = " ".join([sequence[i:i+2] for i in range(0,
↳len(sequence), 2)])
    return grouped_sequence, sequence, original_length // 2, 0

# Detect gaps and distribute missing nucleotides
gaps = []
for i in range(1, len(sequence)):
    if ord(sequence[i]) - ord(sequence[i - 1]) > 1: # Detect gaps
        gaps.append((i, ord(sequence[i]) - ord(sequence[i - 1]) - 1))

filled_sequence = list(sequence)
remaining_to_fill = missing_length

# Fill gaps symmetrically
for gap_index, gap_size in gaps:
    if remaining_to_fill <= 0:
        break
    fill_size = min(gap_size, remaining_to_fill)
    filler = 'N' * fill_size # Use 'N' for missing nucleotides
    remaining_to_fill -= fill_size
    filled_sequence.insert(gap_index, filler)

# If any nucleotides are still missing, add them to the end
if remaining_to_fill > 0:
    filled_sequence.append('N' * remaining_to_fill)

# Combine into complete sequence
filled_sequence = ''.join(filled_sequence)

# Group into pairs
grouped_sequence = " ".join([filled_sequence[i:i+2] for i in range(0,
↳len(filled_sequence), 2)])

# Calculate pairs
total_pairs = len(filled_sequence) // 2
added_pairs = missing_length // 2

```

```
return grouped_sequence, filled_sequence, total_pairs, added_pairs
```

```
# Full Input Sequence
```

```
original_sequence = (
```

```
"ATGGGGACGGAAGACTGCGATCACGAAGGGCGGTGCGTTGCGGCTCCCGTGGAGGTTACGGCGCTGTATG"  
"CGACCGACGGGTGCGTTATCACCTCCTCGCTCGCCCTCCTCACAACTGCCTGCTGGGGGCCGAGCCGTT"  
"GTATATATTTCAGCTACGACGCGTACCGGCCCGATGCGCCCAATGGCCCCACGGGCGCGCCACCGAACAG"  
"GAGAGGTTTCGAGGGGAGCCGGGCGCTCTACCGGGATGCGGGGGGGCTAAATGGCGATTCAATTCGGGTGA"  
"CCTTTTGTATTGTTGGGACGGAAGTGGGCGTGACCCACCACCCGAAAGGGCGCACCCGGCCCATGTTTGT"  
"GTGCCGCTTCGAGCGAGCGGACGACGTGCGCGTGCTCCAAGACGCCCTGGGCGCGGGACCCCATGTGCTC"  
"CCGGCCACATCACAGCAACTCTGGAATTGGAGGCGACGTTTTCGCTCCACGCTAACATCATCATGGCTC"  
"TCACCGTGGCCATCGTCCACAACGCCCCCGCCGCATCGGCAGCGGCAGCACCGCCCCCTGTATGAGCC"  
"CGGCGAATCGATGCGCTCGTCTCGGGCGCATGTCCCTGGGGCAGCGCGGCTCACCACGCTGTTCTGTG"  
"CACACGAGGCGCGCGTGTGGCGGCGTACCGCCGGGCGTATTATGGGAGCGCCAAAGCCCCCTTTGGT"  
"TTCTGAGCAAATTCGGCCCGGACGAAAAGAGCCTGGTGTGCGCGCTAGGTACTACCTACTCCAGGCTCC"  
"GCGCTTGGGGGGCGCCGGAGCCACGTACGATCTGCAGGCCGTGAAAGACATCTGCGCGACCTACGCGATC"  
"CCCCACGACCCACGCCCCGACACCTCAGTGCCGCGTCTTGACCTCGTTGCGCGCCATCACTCGGTTCT"  
"GTTGCACGAGCCAGTACTCCCGCGGGGCGCGGCCGCTGGGTTTCCGCTGTATGTGGAGCGCCGCATCGC"  
"CGCCGACGTACGCGAGACCGGCGCGCTGGAGAAGTTCATCGCCACGATCGCAGCTGCCTGCGCGTGTCC"  
"GACCGGAATTCATTACGTACATCTACCTGGCCCACTTTGAGTGCTTCAGCCCCCGCGCCTGGCCACGC"  
"ATCTCCGGGCGGTGACCACCCACGACCCAGCCCCGCGGCCAGCACGGAGCAGCCCTCGCCCTGGGTG"  
"GGAGGCGGTGGAACAGTTCTTCCGGCACGTGCGCGCCAGCTGAACATCCGCGAGTACGTAAAGCAAAAC"  
"GTCACCCCCAGGAAACCGCCCTGGCGGGAGACGCGGCCCGCCCTACCTGCGCGCGCGCACGTATGCCC"  
"CGGCGGCCCTCACGCCCCCCCCGCGTACTGCGGGGTGCGAGACTCGTCCACCAAAATGATGGGACGTCT"  
"GGCGGAAGCAGAAAGGCTCCTAGTCCCCACGGCTGGCCCGGTTGCGACCAACAACCCCGGGGACGAC"  
"GCGGGGGGCGGCACTGCCGCCCCCAGACCTGCGGAATCGTCAAGCGCCTCCTCAAGCTGGCCGCCACGG"  
"AGCAGCAGGGCACGACGCCCCCGCGATCGCGGCTCTCATGCAGGACGCGTCCGTTCCAAACCCCTGCC"  
"CGTGTACAGGATTACCATGTCCCCGACCGGCCAGGCGTTTGGCCGGCGGGCGCGGGGACGACTGGGCCCGC"  
"GTGACGCGGGACGCGCGCCCGCGGAAGCGACCGTGGTTCGCGACGCGGCGGGCGGCCGAGCCCGCG"  
"CGCTCGGCCGGCGGCTCACGCGCCGATTTGCGCCCGGGGCCCCGCGCTCCCCCGGGCGGCGTGGCCGT"  
"CGGGGGCCAGATGTACGTGAACCGCAACGAGATCTTCAACGCCGCGTGGCCGTTACGAACATCATCTG"  
"GATCTGGACATCGCCCTGAAGGAGCCCGTCCCCTTTCCCGGCTCCACGAGGCCCTGGGTCACTTAGGC"  
"GCGGGGCGTGGCGCGGTTACGTGTTGTTTCCCGCGCCCGGTAGACCCCGACGCCTATCCCTGTTA"  
"TTTTTTCAAAGCGCCTGTGCGCCCCGCGCGCCCGCTGTGTGCGGGGACGGGCCCTGGCCGTTGGC"  
"GACGACGGGCGACGGGGACTGGTTCCCCGACGCCGTTGGTCCCGGCGACGAGGAGTGGGAGGAGACACGG"  
"ACCCCATGGACACGACCCACGGCCCCCTCCCGGACGACGAGGCCGCTACCTCGACCTGCTACACGAACA"  
"GATACGAGCGGCGACGCCAGGAACCGGACTCCGTGTTGTTTCTGCGCCGACAAGATCGGGCTGCGC"  
"GTGTGCTACCGGTCCCCGCCCCGTACGTTGTGACGGCTCCCTGACGATGCGTGGGGTGGCGAGGGTGA"  
"TCCAGCAGGCGGTGCTGTTGGACCGCGACTTCGTGGAGGCCGTAGGGAGCCACGTAAAGAACTTTTGT"  
"GATCGATACGGGCGTGTACGCCACGGCCACAGCCTGCGCTTGCCGTATTTGCGCAAGATCGGCCCGAC"  
"GGCTCCGCGTGGCGCCGTTATTGCCCCTTCGTGATCCCCCGCGTGCGAGGACGTTCCGCGGTTGCG"  
"TCGCGCGCACGCCGACCCGCGGCGCTTCCACTTTCACGCCCCCGCCATGTTTTCCGCGCCCCCGGGA"  
"GATCCGCGTCTCCACAGCCTGGGCGGGGACTATGTGAGCTTTTTCGAGAAGAAGGCGTCGCGCAACGCC"  
"CTGGAGCACTTTGGGCGACGCGAGACCTGACGGAGTTCTGGGCCGCTACGATGTGCGGCCGACGCGG"  
"GGGAGACCGTGGAGGGGTTGCGGTCAGAACTGCTGGGGCGAATAGTCGCGTGCATCGAGGCCACTTTC"  
"CGAGCACGCGCGGGAATATCAGGCCGTGTCGTTCCCGGGCCGTCATTAAGGACGACTGGGTCCTGCTG"
```

```

"CAGCTGATCCCCGGCCGCGGCCCTGAACCAAAGCCTCTCGTGTCTGCGCTTCAAGCACGGCAGGGCAA"
"GTCGCGCGACGGCCCGGACCTTTCTCGCGCTGAGCGTCGGGACCAACAACCGCCTATGCGCGTCCCTGTG"
"TCAGCAGTGCTTTGCCACTAAATGCGATAACAACCGCCTGCACACGCTGTTTACCGTCGATGCGGGCAG"
"CCATGCTCGCGGTCCGCTCCCTCCAGCACCTCACGACGTCATCTTCATAA")

expected_sequence_length = 4096
expected_sequence_length = 4096

# Run the refined function
grouped_sequence, filled_sequence, total_pairs, added_pairs = \
    identify_and_fill_gaps_properly(
        original_sequence, expected_sequence_length
    )

# Display results
print("Original Sequence Length:", len(original_sequence))
print("Expected Sequence Length:", expected_sequence_length)
print("Filled Sequence Length:", len(filled_sequence))
print("Total Pairs in Filled Sequence:", total_pairs)
print("Added Pairs:", added_pairs)
print("\nGrouped Sequence (First 200 pairs):\n", grouped_sequence) # Show
    first 200 pairs

# Optionally save the grouped sequence to a file
with open("grouped_sequence_refined.txt", "w") as file:
    file.write(grouped_sequence)

```

Original Sequence Length: 3201  
 Expected Sequence Length: 4096  
 Filled Sequence Length: 4096  
 Total Pairs in Filled Sequence: 2048  
 Added Pairs: 447

Grouped Sequence (First 200 pairs):

```

AN NN NN NN NN NN NN NN NN NN NT GG GG NN NN AC GN NN NN GN NN NN NN NN NN NN NN
NN AA NN NG NN NN NN NN NN NN NN NN NN NN NN AC NN NN TG NN NN NC GA NN NT NN NN NN NN
NN NN CN NN AN NN NN NN NN NN NN NC GA NN NA GN NN NN NN NN NN NN NN NG GC NN NN NN NN
NN NN NN NN NG GT NN NN NN NC NN NN NN NN NN NN NN GG NN NN TT NN NG NN NN NN NN NN NN
NN NN CN NN NN NN NN NN NN NG NN NN NN NN NN NN NN NN NN NN GC NN NT NC NN NC NN NN CG
NN NN NN NN NN NN NN TG NN NN NN NN NN NN NN NN NN NN NG AN NN NN NN NN NN NN NN NG GN TN
NN NN NN NN NN NN NN NT AN NN NN NN NN NN NN NN NN NN NC NN NG NN NN NN NN NN NN NN NN
GN NN CG CN NN NN NN NN NN NN NN NN NT GN NN NN NN NN NN NN NN NN NT AN TG CN NN NN NN
NN NN NN NN NN GA CN NN NN NN NN NN NN NN NN NN NC GN NN NN NN NN NN NN NN NA CG GG TG
NN NC NN NN NG TN NN NN NN NN NN NN NN NN TA NN NN NN NN NN NN NN TN NN NN NN NN NN NN
NN NN NC NN NN NN NN NN NN NN NN NN NN AN NN NN NN NN NN NN NN NN NN NC CT NN NN NC NN
NN NN NN NN NN NN NN CN NN NT NN NN CN NN NN NN NN NN NN NN NN GN CN NN TC GC NN NC
NN NN NN NN NN NN NN NN NN NN CT NN NC CT CA NN NN NN NN NN NN NN NN NN NN CA AA CT GN

```

NN NC CT NN NG NN NC TG GN GN NN GG NC NN NN NC GN NN NN AN NN NN GN NN NN NN NN  
 NN NC CN NN GN NN NN TT GT NN NN NA TN NN AT AN NN TN NN NN NN NN NN NN NT NN  
 NN NN NN NN NN NN NN CN AN NN GC TN NN NN NN NN NN NN NN NN NA CN NN GA CG CG TN  
 NN NN NN NN NN NN NN NA CC NN NN NN NN NN NN NN NN NN GG CN NN CN NN NN NN NN  
 NN NN NN NC GA NN NN NN NN NN NN NN NN NN TG CN NN GC NN NN NN NN NN NN CC NA NN  
 NN NN NN NN NN NN NN AT GG NN NN NN NN NN NN CC CN NN NN NN NN NN NN NN NC AC  
 GG GN NN NC GC NN NN NN NN NN NN NN NN NG CC CN NN NN NN NN NN NN AC NC GA NA  
 CN AG NN NG AG NN NN NA GG NN NT TN CG NN NA GG GG NN NN NN NN AG CC GG GC GC TC  
 TA CC GG GA TG CG GG GG GG CT AA AT GG CG AT TC AT TT CG GG TG AC CT TT TG TT TA  
 TT GG GG AC GG AA GT GG GC GT GA CC CA CC AC CC GA AA GG GC GC AC CC GG CC CA TG  
 TT TG TG TG CC GC TT CG AG CG AG CG GA CG AC GT CG CC GT GC TC CA AG AC GC CC TG  
 GG CC GC GG GA CC CC AT TG CT CC CG GC CC AC AT CA CA GC AA CT CT GG AC TT GG AG  
 GC GA CG TT TG CG CT CC AC GC TA AC AT CA TC AT GG CT CT CA CC GT GG CC AT CG TC  
 CA CA AC GC CC CC GC CC GC AT CG GC AG CG GC AG CA CC GC CC CC CT GT AT GA GC CC  
 GG CG AA TC GA TG CG CT CG GT CG TC GG GC GC AT GT CC CT GG GG CA GC GC GG CC TC  
 AC CA CG CT GT TC GT GC AC CA CG AG GC GC GC GT GC TG GC GG CG TA CC GC CG GG CG  
 TA TT AT GG GA GC GC CC AA AG CC CC TT TT GG TT TC TG AG CA AA TT CG GC CC GG AC  
 GA AA AG AG CC TG GT GC TG GC CG CT AG GT AC TA CC TA CT CC AG GC TC CG CG CT TG  
 GG GG GC GC CG GA GC CA CG TA CG AT CT GC AG GC CG TG AA AG AC AT CT GC GC GA CC  
 TA CG CG AT CC CC CA CG AC CC AC GC CC CG AC AC CC TC AG TG CC GC GT CC TT GA CC  
 TC GT TC GC CG CC AT CA CT CG GT TC TG TT GC AC GA GC CA GT AC TC CC GC GG GG CC  
 GC GG CC GC TG GG TT TC CG CT GT AT GT GG AG CG CC GC AT CG CC GC CG AC GT AC GC  
 GA GA CC GG CG CG CT GG AG AA GT TC AT CG CC CA CG AT CG CA GC TG CC TG CG CG TG  
 TC CG AC CG GG AA TT CA TT AC GT AC AT CT AC CT GG CC CA CT TT GA GT GC TT CA GC  
 CC CC CG CG CC TG GC CA CG CA TC TC CG GG CC GT GA CC AC CC AC GA CC CC AG CC CC  
 GC GG CC AG CA CG GA GC AG CC CT CG CC CC TG GG TC GG GA GG CG GT GG AA CA GT TC  
 TT CC GG CA CG TG CG CG CC CA GC TG AA CA TC CG CG AG TA CG TA AA GC AA AA CG TC  
 AC CC CC AG GG AA AC CG CC CT GG CG GG AG AC GC GG CC GC CG CC TA CC TG CG CG CG  
 CG CA CG TA TG CC CC GG CG GC CC TC AC GC CC GC CC CC GC GT AC TG CG GG GT CG CA  
 GA CT CG TC CA CC AA AA TG AT GG GA CG TC TG GC GG AA GC AG AA AG GC TC CT AG TC  
 CC CC AC GG CT GG CC CG CG TT CG CA CC AA CA AC CC CC GG GG AC GA CG CG GG GG GC  
 GG CA CT GC CG CC CC CC AG AC CT GC GG AA TC GT CA AG CG CC TC CT CA AG CT GG CC  
 GC CA CG GA GC AG CA GG GC AC GA CG CC CC CG GC GA TC GC GG CT CT CA TG CA GG AC  
 GC GT CG GT CC AA AC CC CC CT GC CC GT GT AC AG GA TT AC CA TG TC CC CG AC CG GC  
 CA GG CG TT TG CC GC GG CG GC GC GG GA CG AC TG GG CC CG CG TG AC GC GG GA CG CG  
 CG CC CG CC GG AA GC GA CC GT GG TC GC GG AC GC GG CG GC GG CG CC CG AG CC CG GC  
 GC GC TC GG CC GG CG GC TC AC GC GC CG CA TT TG CG CC CG GG GC CC CG CG CT CC CC  
 CC GG GC GG CC TG GC CG TC GG GG GC CA GA TG TA CG TG AA CC GC AA CG AG AT CT TC  
 AA CG CC GC GC TG GC CG TT AC GA AC AT CA TC CT GG AT CT GG AC AT CG CC CT GA AG  
 GA GC CC GT CC CC TT TC CC CG GC TC CA CG AG GC CC TG GG TC AC TT TA GG CG CG GG  
 GC GC TG GC GG CG GT TC AG CT GT TG TT TC CC GC GG CC CG CG TA GA CC CC GA CG CC  
 TA TC CC TG TT AT TT TT TC AA AA GC GC CT GT CG GC CC CG CG CG CC GC CC GT CT GT  
 GC GG GC GA CG GG CC CC TG GC CG GT GG CG AC GA CG GC GA CG GG GA CT GG TT CC CC  
 GA CG CC GG TG GT CC CG GC GA CG AG GA GT GG GA GG AG GA CA CG GA CC CC AT GG AC  
 AC GA CC CA CG GC CC CC TC CC GG AC GA CG AG GC CG CG TA CC TC GA CC TG CT AC AC  
 GA AC AG AT AC CA GC GG CG AC GC CC AG CG AA CC GG AC TC CG TC GT GT GT TC CT GC  
 GC CG AC AA GA TC GG GC TG CG CG TG TG CC TA CC GG TC CC CG CC CC GT AC GT TG TG  
 CA CG GC TC CC TG AC GA TG CG TG GG GT GG CG AG GG TG AT CC AG CA GG CG GT GC TG

TT GG AC CG CG AC TT CG TG GA GG CC GT AG GG AG CC AC GT AA AG AA CT TT TT GC TG  
 AT CG AT AC GG GC GT GT AC GC CC AC GG CC AC AG CC TG CG CT TG CC GT AT TT CG CC  
 AA GA TC GG CC CC GA CG GC TC CG CG TG CG GC CG GT TA TT GC CC GT CT TC GT GA TC  
 CC CC CC GC GT GC GA GG AC GT TC CG GC GT TC GT CG CC GC GC AC GC CG AC CC GC GG  
 CG CT TC CA CT TT CA CG CC CC GC CC AT GT TT TC CG CG GC CC CG CG GG AG AT CC GC  
 GT CC TC CA CA GC CT GG GC GG GG AC TA TG TC AG CT TT TT CG AG AA GA AG GC GT CG  
 CG CA AC GC CC TG GA GC AC TT TG GG CG AC GC GA GA CC CT GA CG GA GG TT CT GG GC  
 CG CT AC GA TG TG CG GC CC GA CG CC GG GG AG AC CG TG GA GG GG TT CG CG TC AG AA  
 CT GC TG GG GC GA AT AG TC GC GT GC AT CG AG GC CC AC TT TC CC GA GC AC GC GC GG  
 GA AT AT CA GG CC GT GT CC GT TC GC CG GG CC GT CA TT AA GG AC GA CT GG GT CC TG  
 CT GC AG CT GA TC CC CG GC CG CG GC GC CC TG AA CC AA AG CC TC TC GT GT CT GC GC  
 TT CA AG CA CG GC AG GG CA AG TC GC GC GA CG GC CC GG AC CT TT CT CG CG CT GA GC  
 GT CG GG AC CA AC AA CC GC CT AT GC GC GT CC CT GT GT CA GC AG TG CT TT GC CA CT  
 AA AT GC GA TA AC AA CC GC CT GC AC AC GC TG TT TA CC GT CG AT GC GG GC AC GC CA  
 TG CT CG CG GT CC GC TC CC TC CA GC AC CT CA CG AC CG TC AT CT TC AT AA

[ ]: