isblab2

January 3, 2024

1 Lokesh Yarramallu - 22055

[53]:
```
codon_dict = {
    'A': ['GCU', 'GCC', 'GCA', 'GCG'],
    'C': ['UGU', 'UGC'],
    'D': ['GAU', 'GAC'],
    'E': ['GAA', 'GAG'],
    'F': ['UUU', 'UUC'],
    'G': ['GGU', 'GGC', 'GGA', 'GGG'],
    'H': ['CAU', 'CAC'],
    'I': ['AUU', 'AUC', 'AUA'],
    'K': ['AAA', 'AAG'],
    'L': ['UUA', 'UUG', 'CUU', 'CUC', 'CUA', 'CUG'],
    'M': ['AUG'],
    'N': ['AAU', 'AAC'],
    'P': ['CCU', 'CCC', 'CCA', 'CCG'],
    'Q': ['CAA', 'CAG'],
    'R': ['CGU', 'CGC', 'CGA', 'CGG', 'AGA', 'AGG'],
    'S': ['UCU', 'UCC', 'UCA', 'UCG', 'AGU', 'AGC'],
    'T': ['ACU', 'ACC', 'ACA', 'ACG'],
    'V': ['GUU', 'GUC', 'GUA', 'GUG'],
    'W': ['UGG'],
    'Y': ['UAU', 'UAC'],
    '_': ['UAA', 'UAG', 'UGA']
}

# insulin dna
dna ="""AGCCCTCCAGGACAGGCTGCATCAGAAGAGGCCATCAAGCAGGTCTGTTCCAAGGGCCTTTGCGTCAGGT
                                ␣
 ↪GGGCTCAGGATTCCAGGGTGGCTGGACCCCAGGCCCCAGCTCTGCAGCAGGGAGGACGTGGCTGGGCTCG
                                ␣
 ↪TGAAGCATGTGGGGGTGAGCCCAGGGGCCCCAAGGCAGGGCACCTGGCCTTCAGCCTGCCTCAGCCCTGC
                                ␣
 ↪CTGTCTCCCAGATCACTGTCCTTCTGCCATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTG
                                ␣
 ↪GCCCTCTGGGGACCTGACCCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAG
```

```
                                ␣
 ↪CTCTCTACCTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGACCT
                                ␣
 ↪GCAGGGTGAGCCAACTGCCCATTGCTGCCCCTGGCCGCCCCCAGCCACCCCCTGCTCCTGGCGCTCCCAC
                                ␣
 ↪CCAGCATGGGCAGAAGGGGGCAGGAGGCTGCCACCCAGCAGGGGGTCAGGTGCACTTTTTTAAAAAGAAG
                                ␣
 ↪TTCTCTTGGTCACGTCCTAAAAGTGACCAGCTCCCTGTGGCCCAGTCAGAATCTCAGCCTGAGGACGGTG
                                ␣
 ↪TTGGCTTCGGCAGCCCCGAGATACATCAGAGGGTGGGCACGCTCCTCCCTCCACTCGCCCCTCAAACAAA
                                ␣
 ↪TGCCCCGCAGCCCATTTCTCCACCCTCATTTGATGACCGCAGATTCAAGTGTTTTGTTAAGTAAAGTCCT
                                ␣
 ↪GGGTGACCTGGGGTCACAGGGTGCCCCACGCTGCCTGCCTCTGGGCGAACACCCCATCACGCCCGGAGGA
                                ␣
 ↪GGGCGTGGCTGCCTGCCTGAGTGGGCCAGACCCCTGTCGCCAGGCCTCACGGCAGCTCCATAGTCAGGAG
                                ␣
 ↪ATGGGGAAGATGCTGGGGACAGGCCCTGGGGAGAAGTACTGGGATCACCTGTTCAGGCTCCCACTGTGAC
                                ␣
 ↪GCTGCCCCGGGGCGGGGGAAGGAGGTGGGACATGTGGGCGTTGGGGCCTGTAGGTCCACACCCAGTGTGG
                                ␣
 ↪GTGACCCTCCCTCTAACCTGGGTCCAGCCCGGCTGGAGATGGGTGGGAGTGCGACCTAGGGCTGGCGGGC
                                ␣
 ↪AGGCGGGCACTGTGTCTCCCTGACTGTGTCCTCCTGTGTCCCTCTGCCTCGCCGCTGTTCCGGAACCTGC
                                ␣
 ↪TCTGCGCGGCACGTCCTGGCAGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGC
                                ␣
 ↪CCTTGGCCCTGGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCT
                                ␣
 ↪CTACCAGCTGGAGAACTACTGCAACTAGACGCAGCCCGCAGGCAGCCCCACACCCGCCGCCTCCTGCACC
                             GAGAGAGATGGAATAAAGCCCTTGAACCAGC"""

rna = dna.replace('T', 'U')

def rna2protein(rna, codon_table):
  protein_sequence = ""
  rna = [rna[i:i+3] for i in range(0, len(rna), 3)]
  for protein in rna:
    for acid, codons in codon_table.items():
      if protein in codons:
        protein_sequence +=acid
        break
  return protein_sequence


protein = rna2protein(rna, codon_dict)
```

```
print("RNA Sequence:")
print(rna)

print("\n\nAmino Acid Sequence:")
print(protein)
```

```
RNA Sequence:
AGCCCUCCAGGACAGGCUGCAUCAGAAGAGGCCAUCAAGCAGGUCUGUUCCAAGGGCCUUUGCGUCAGGU
GGGCUCAGGAUUCCAGGGUGGCUGGACCCCAGGCCCCAGCUCUGCAGCAGGGAGGACGUGGCUGGGCUCG
UGAAGCAUGUGGGGGUGAGCCCAGGGGCCCCAAGGCAGGGCACCUGGCCUUCAGCCUGCCUCAGCCCUGC
CUGUCUCCCAGAUCACUGUCCUUCUGCCAUGGCCCUGUGGAUGCGCCUCCUGCCCCUGCUGGCGCUGCUG
GCCCUCUGGGGACCUGACCCAGCCGCAGCCUUUGUGAACCAACACCUGUGCGGCUCACACCUGGUGGAAG
CUCUCUACCUAGUGUGCGGGGAACGAGGCUUCUUCUACACACCCAAGACCCGCCGGGAGGCAGAGGACCU
GCAGGGUGAGCCAACUGCCCAUUGCUGCCCCUGGCCGCCCCCAGCCACCCCCUGCUCCUGGCGCUCCCAC
CCAGCAUGGGCAGAAGGGGGCAGGAGGCUGCCACCCAGCAGGGGGUCAGGUGCACUUUUUUAAAAAGAAG
UUCUCUUGGUCACGUCCUAAAAGUGACCAGCUCCCUGUGGCCCAGUCAGAAUCUCAGCCUGAGGACGGUG
UUGGCUUCGGCAGCCCCGAGAUACAUCAGAGGGUGGGCACGCUCCUCCCUCCACUCGCCCCUCAAACAAA
UGCCCCGCAGCCCAUUUCUCCACCCUCAUUUGAUGACCGCAGAUUCAAGUGUUUUGUUAAGUAAAGUCCU
GGGUGACCUGGGGUCACAGGGUGCCCCACGCUGCCUGCCUCUGGGCGAACACCCCAUCACGCCCGGAGGA
GGGCGUGGCUGCCUGCCUGAGUGGGCCAGACCCCUGUCGCCAGGCCUCACGGCAGCUCCAUAGUCAGGAG
AUGGGGAAGAUGCUGGGGACAGGCCCUGGGGAGAAGUACUGGGAUCACCUGUUCAGGCUCCCACUGUGAC
GCUGCCCCGGGGCGGGGGAAGGAGGUGGGACAUGUGGGCGUUGGGGCCUGUAGGUCCACACCCAGUGUGG
GUGACCCUCCCUCUAACCUGGGUCCAGCCCGGCUGGAGAUGGGUGGGAGUGCGACCUAGGGCUGGCGGGC
AGGCGGGCACUGUGUCUCCCUGACUGUGUCCUCCUGUGUCCCUCUGCCUCGCCGCUGUUCCGGAACCUGC
UCUGCGCGGCACGUCCUGGCAGUGGGGCAGGUGGAGCUGGGCGGGGGCCCUGGUGCAGGCAGCCUGCAGC
CCUUGGCCCUGGAGGGGUCCCUGCAGAAGCGUGGCAUUGUGGAACAAUGCUGUACCAGCAUCUGCUCCCU
CUACCAGCUGGAGAACUACUGCAACUAGACGCAGCCCGCAGGCAGCCCCACACCCGCCGCCUCCUGCACC
                              GAGAGAGAUGGAAUAAAGCCCUUGAACCAGC


Amino Acid Sequence:
SPPGQAASEEAIKQVCSKGLCVRGLRIPGWLDPRPQLCSREDVAGL_SMWG_AQGPQGRAPGLQPASALLSPRSLSFCHG
PVDAPPAPAGAAALWGPDPAAAFVNQHLCGSHLVELST_CAGNEASSTHPRPAGRQRTAG_ANCPLLPLAAPSHPLLLAL
PPAWAEGGRRLPPSRGSGALF_KEFSWSRPKSDQLPVAQSESQPEDGLASAAPRYIRGWARSSLHSPLKQCPAAHFSTLI
__PQIQVFC_VKSG_PGVTGCPTLPASGRTPHHARRGRGCLPEWARPLSPGLTAAP_SGMGKMLGTGPGEKYWDHLFRLP
L_AAPGRGKEVGHVGVGACRSTPSVVTLPLTWVQPGWRWVGVRPRAGGRRALCLPDCVLLCPSASPLFRNLSARHVLAVG
QVELGGGPGAGSLQPWPWRGPCRSVALWNNAVPASAPLPAGELLQLDAARRQPHTRRLLHERDGIKPLNQ
```

[54]:
```
fasta_content = """>O00626|HUMAN Small inducible cytokine A22.
MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS
CPRPGVVLLTFRDKEICADPR
VPWVKMILNKLSQ"""

file_path = "example.fasta"

with open(file_path, "w") as file:
```

```
    file.write(fasta_content)

# a. Read the file, extract the header information, and print it.
def extract_header(file_path):
    with open(file_path, "r") as file:
        header = file.readline().strip()[1:]
        print("Header:", header)

# b. Read and print the sequence from the file.
def extract_sequence(file_path):
    with open(file_path, "r") as f:
        f.readline()
        while True:
            line = f.readline()
            if not line:
                break
            print(line.strip())


extract_header(file_path)
extract_sequence(file_path)



# c. Append molecular weight of the sequence at the end of the file.
acid_weights = {
    'A': 89.093, 'C': 121.158, 'D': 133.103, 'E': 147.129,
    'F': 165.192, 'G': 75.067, 'H': 155.155, 'I': 131.175,
    'K': 146.189, 'L': 131.175, 'M': 149.208, 'N': 132.118,
    'P': 115.132, 'Q': 146.146, 'R': 174.203, 'S': 105.093,
    'T': 119.119, 'V': 117.146, 'W': 204.228, 'Y': 181.191
}

seq=""
with open(file_path, "r") as f:
  f.readline()
  while True:
    line = f.readline()
    if not line:
      break
    seq+=line.strip()

print(f"Amino Acid Sequence: {seq}")
mweight=0
for i in seq:
  mweight+=acid_weights[i]
```

```
print(f"\nMolecular Weight of the Amino Acid Sequence: {round(mweight, 2)}u.")

with open(file_path, "a") as f:
  f.write(f"\nMolecular Weight of the Amino Acid Sequence: {round(mweight, 2)}u.
  ↪")
```

```
Header: O00626|HUMAN Small inducible cytokine A22.
MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS
CPRPGVVLLTFRDKEICADPR
VPWVKMILNKLSQ
Amino Acid Sequence: MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS
CPRPGVVLLTFRDKEICADPRVPWVKMILNKLSQ

Molecular Weight of the Amino Acid Sequence: 12237.93u.
```

[55]:
```
with open(file_path, "r") as f:
  print(f.read())
```

```
>O00626|HUMAN Small inducible cytokine A22.
MARLQTALLVVLVLLAVALQATEAGPYGANMEDSVCCRDYVRYRLPLRVVKHFYWTSDS
CPRPGVVLLTFRDKEICADPR
VPWVKMILNKLSQ
Molecular Weight of the Amino Acid Sequence: 12237.93u.
```

[56]:
```
# vibrio cholerae genome

# reading a txt file
with open("VibrioCholeraGenome.txt", "r") as file:
    genome = file.read()

with open("VibrioCholeraOric.txt", "r") as file:
    oric = file.read()

oric= oric.upper().replace("\n", "")
```

[57]:
```
# Ques 3

def PatternCount(dna, Pattern):
  k=len(pattern)
  cnt=0
  for i in range(len(dna)-k):
    if dna[i:i+k] == pattern:
      cnt+=1
  return cnt


pattern = "ATGATCAAG"
```

```
output = PatternCount(oric, pattern)
print("Output:", output)
```

```
Output: 3
```

[58]:
```
# Ques 4

def PatternMatching(Pattern, Genome):
    positions = []
    k = len(Pattern)
    for i in range(len(Genome) - k + 1):
        if Genome[i:i+k] == Pattern:
            positions.append(i)
    return positions


pattern = "ATGATCAAG"

output = PatternMatching(pattern, genome)
print("Output:", "\n".join(map(str, output)))
```

```
Output: 116556
149355
151913
152013
152394
186189
194276
200076
224527
307692
479770
610980
653338
679985
768828
878903
985368
```

[59]:
```
# Ques 5

def frequent_words(text, k):
    kmer_counts = {}

    for i in range(len(text) - k + 1):
```

```
        kmer = text[i:i+k]
        kmer_counts[kmer] = kmer_counts.get(kmer, 0) + 1

    max_count = max(kmer_counts.values())

    most_frequent_kmers = [kmer for kmer, count in kmer_counts.items() if count
 ↪== max_count]

    return most_frequent_kmers


text = "Vibrio Cholerae Oric DataSet"
k = 9


output = frequent_words(oric, k)
print("Real Output:", " ".join(output))
```

```
Real Output: ATGATCAAG CTCTTGATC TCTTGATCA CTTGATCAT
```