19BIO201
Intelligence Of Biological Systems 3
Assignment-4
Topic: String reconstruction using debruijn graph

1. Write a program in any programming language (Java or Python) to carry out string reconstruction using DeBruijn graph for the sequences given below (Attempt for k=3 & k=5).

a) TAATGCT
b) GCGGTAATGCAGTCGAC
c) TTGAGTGCCAGCGAGCTAGGTCAGT

CODE:

```
# Amruth
# BL.EN.U4AIE20002
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import pylab

def kmers(read, k):
    KList=[]
    num_kmers = len(read) - k + 1
    for i in range(num_kmers):
        kmer = read[i:i+k]
        if i==0:
            KList.append(kmer)
        else:
            KList.append(kmer)
    return KList

input1="TAATGCCATGGGATGTT"
Kmers=kmers(input1,3)

print(f"Kmers={Kmers}")
print(" ")

def FindNodesFromEdges(KmerList):
    NodesListOfLists=[]
    for edge in KmerList:
        NodesListOfLists.append([edge[:-1],edge[1:]])

    return NodesListOfLists

NodesListOfLists = FindNodesFromEdges(Kmers)
print(f"Nodes = {NodesListOfLists}")
```

```
Kmers=['TAA', 'AAT', 'ATG', 'TGC', 'GCC', 'CCA', 'CAT', 'ATG', 'TGG', 'GGG', 'GGA', 'GAT', 'ATG', 'TGT', 'GTT']

Nodes = [['TA', 'AA'], ['AA', 'AT'], ['AT', 'TG'], ['TG', 'GC'], ['GC', 'CC'], ['CC', 'CA'], ['CA', 'AT'], ['AT', '
TG'], ['TG', 'GG'], ['GG', 'GG'], ['GG', 'GA'], ['GA', 'AT'], ['AT', 'TG'], ['TG', 'GT'], ['GT', 'TT']]
```

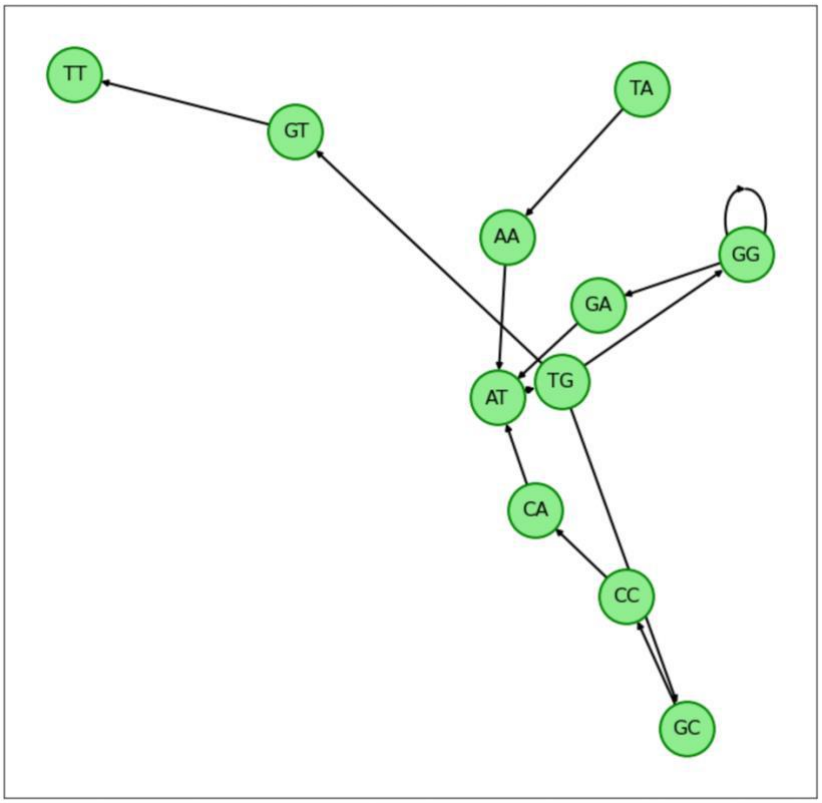
```
G = nx.MultiDiGraph()
G.add_edges_from(NodesListOfLists)
totalNodes=G.nodes()
pos = nx.spring_layout(G)
options = {
    "font_size": 16,
    "node_size": 2000,
    "node_color": "lightgreen",
    "edgecolors": "green",
    "linewidths": 2,
    "width": 2,
    "edge_vmin":5
}
nodes=G.nodes()
edges=G.edges()
print("         ---   DE-BRUIJN Graph   ---         ")
print()
print(f"nodes = {nodes()}")
print(" ")
print(f"edges = {edges}")

plt.figure(2,figsize=(15,15))
nx.draw_networkx(G,pos,**options)
plt.show()
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['TA', 'AA', 'AT', 'TG', 'GC', 'CC', 'CA', 'GG', 'GA', 'GT', 'TT']

edges = [('TA', 'AA'), ('AA', 'AT'), ('AT', 'TG'), ('AT', 'TG'), ('AT', 'TG'), ('TG', 'GC'), ('TG', 'GG'), ('TG', '
GT'), ('GC', 'CC'), ('CC', 'CA'), ('CA', 'AT'), ('GG', 'GG'), ('GG', 'GA'), ('GA', 'AT'), ('GT', 'TT')]
```




```
# converting list of lists in dictionary format
def ConvertListToDict(List):
    Dict={}
    for i in List:
        if i[0] in Dict.keys():
            newList = Dict[i[0]]
            newList.extend([i[1]])
            Dict.update({i[0]: newList})
        else:
            Dict[i[0]] = [i[1]]
    return Dict

Output=ConvertListToDict(edges)
```

```python
def eulerian_cycle(edge_dict):
    tempo1=list(edge_dict.keys())
    current_node = tempo1[0]
    path = [current_node]
    while True:
        path.append(edge_dict[current_node][0])

        if len(edge_dict[current_node]) == 1:
            del edge_dict[current_node]
        else:
            edge_dict[current_node] = edge_dict[current_node][1:]

        if path[-1] in edge_dict:
            current_node = path[-1]
        else:
            break
    while len(edge_dict) > 0:
        for i in range(len(path)):
            if path[i] in edge_dict:
                current_node = path[i]
                cycle = [current_node]
                while True:
                    cycle.append(edge_dict[current_node][0])

                    if len(edge_dict[current_node]) == 1:
                        del edge_dict[current_node]
                    else:
                        edge_dict[current_node] = edge_dict[current_node][1:]

                    if cycle[-1] in edge_dict:
                        current_node = cycle[-1]
                    else:
                        break

                path = path[:i] + cycle + path[i+1:]
                break
    return path
```

```python
print()
print(" Final Eulerian Path from the de-bruijn graph: ")
print()
path = eulerian_cycle(Output)
print ('->'.join(map(str,path)))
print(path)
```

```
 Final Eulerian Path from the de-bruijn graph:

TA->AA->AT->TG->GC->CC->CA->AT->TG->GG->GG->GA->AT->TG->GT->TT
['TA', 'AA', 'AT', 'TG', 'GC', 'CC', 'CA', 'AT', 'TG', 'GG', 'GG', 'GA', 'AT', 'TG', 'GT', 'TT']
```

```python
def reconstructString(List):
    String=""
    String=String+List[0]

    for i in range(1,len(List)):
        newWord = List[i]
        String=String+newWord[-1]

    return String
```

```python
print(f"    Original String   : {input1}")
print(f" Reconstructed String : {reconstructString(path)}")
```

```
    Original String   : TAATGCCATGGGATGTT
 Reconstructed String : TAATGCCATGGGATGTT
```

a) TAATGCT

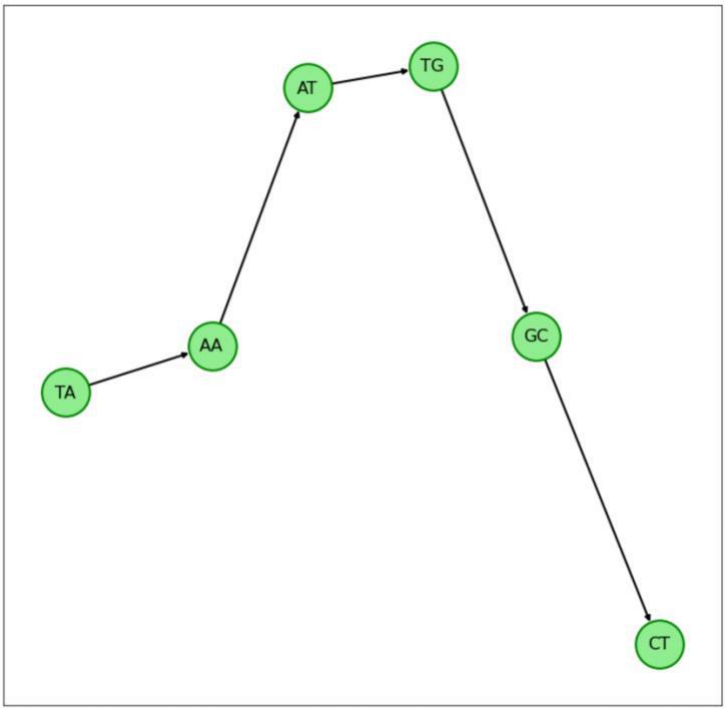
K=3

```
input1="TAATGCT"
Kmers=kmers(input1,3)
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['TA', 'AA', 'AT', 'TG', 'GC', 'CT']

edges = [('TA', 'AA'), ('AA', 'AT'), ('AT', 'TG'), ('TG', 'GC'), ('GC', 'CT')]
```




```
 Final Eulerian Path from the de-bruijn graph:

TA->AA->AT->TG->GC->CT
['TA', 'AA', 'AT', 'TG', 'GC', 'CT']
```

```
   Original String   : TAATGCT
Reconstructed String : TAATGCT
```

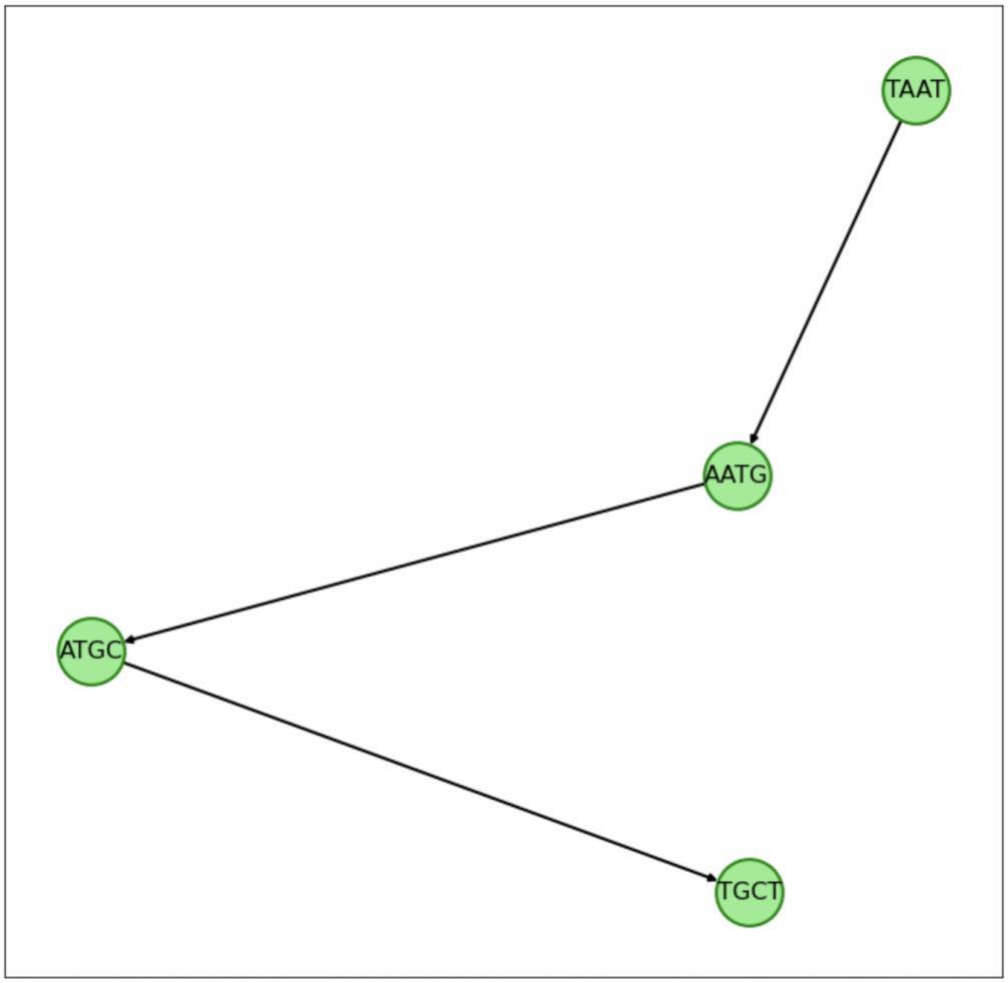
K=5

```
input1="TAATGCT"
Kmers=kmers(input1,5)
```

```
          ---   DE-BRUIJN Graph   ---

nodes = ['TAAT', 'AATG', 'ATGC', 'TGCT']

edges = [('TAAT', 'AATG'), ('AATG', 'ATGC'), ('ATGC', 'TGCT')]
```




```
 Final Eulerian Path from the de-bruijn graph:

TAAT->AATG->ATGC->TGCT
['TAAT', 'AATG', 'ATGC', 'TGCT']
```

```
   Original String   : TAATGCT
Reconstructed String : TAATGCT
```

b)GCGGTAATGCAGTCGAC

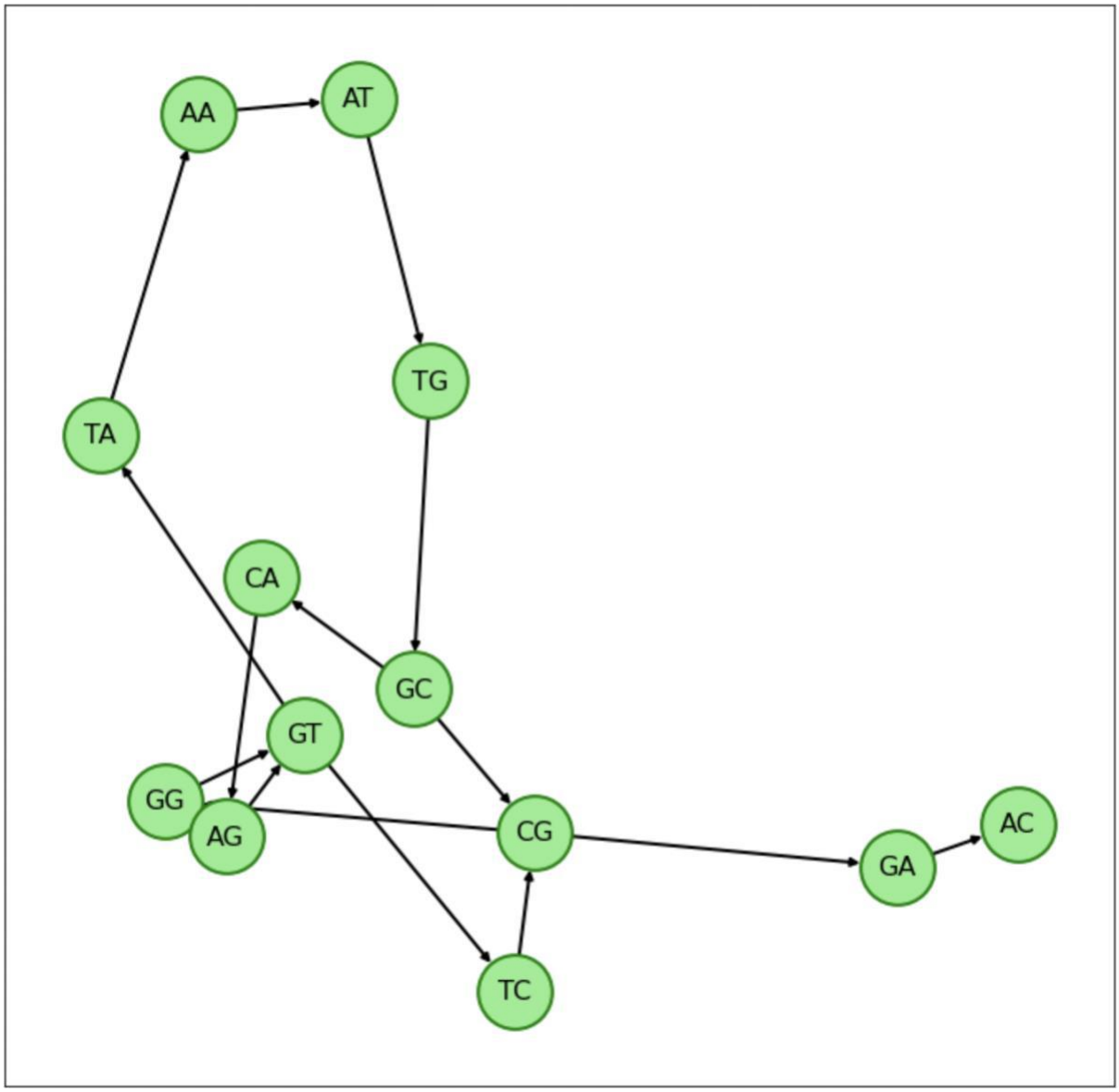
K=3

```
input1='GCGGTAATGCAGTCGAC'
Kmers=kmers(input1,3)
```

```
        ---   DE-BRUIJN Graph   ---

nodes = ['GC', 'CG', 'GG', 'GT', 'TA', 'AA', 'AT', 'TG', 'CA', 'AG', 'TC', 'GA', 'AC']

edges = [('GC', 'CG'), ('GC', 'CA'), ('CG', 'GG'), ('CG', 'GA'), ('GG', 'GT'), ('GT', 'TA'), ('GT', 'TC'), ('TA', '
AA'), ('AA', 'AT'), ('AT', 'TG'), ('TG', 'GC'), ('CA', 'AG'), ('AG', 'GT'), ('TC', 'CG'), ('GA', 'AC')]
```




```
 Final Eulerian Path from the de-bruijn graph:

GC->CG->GG->GT->TA->AA->AT->TG->GC->CA->AG->GT->TC->CG->GA->AC
['GC', 'CG', 'GG', 'GT', 'TA', 'AA', 'AT', 'TG', 'GC', 'CA', 'AG', 'GT', 'TC', 'CG', 'GA', 'AC']
```

```
    Original String   : GCGGTAATGCAGTCGAC
 Reconstructed String : GCGGTAATGCAGTCGAC
```

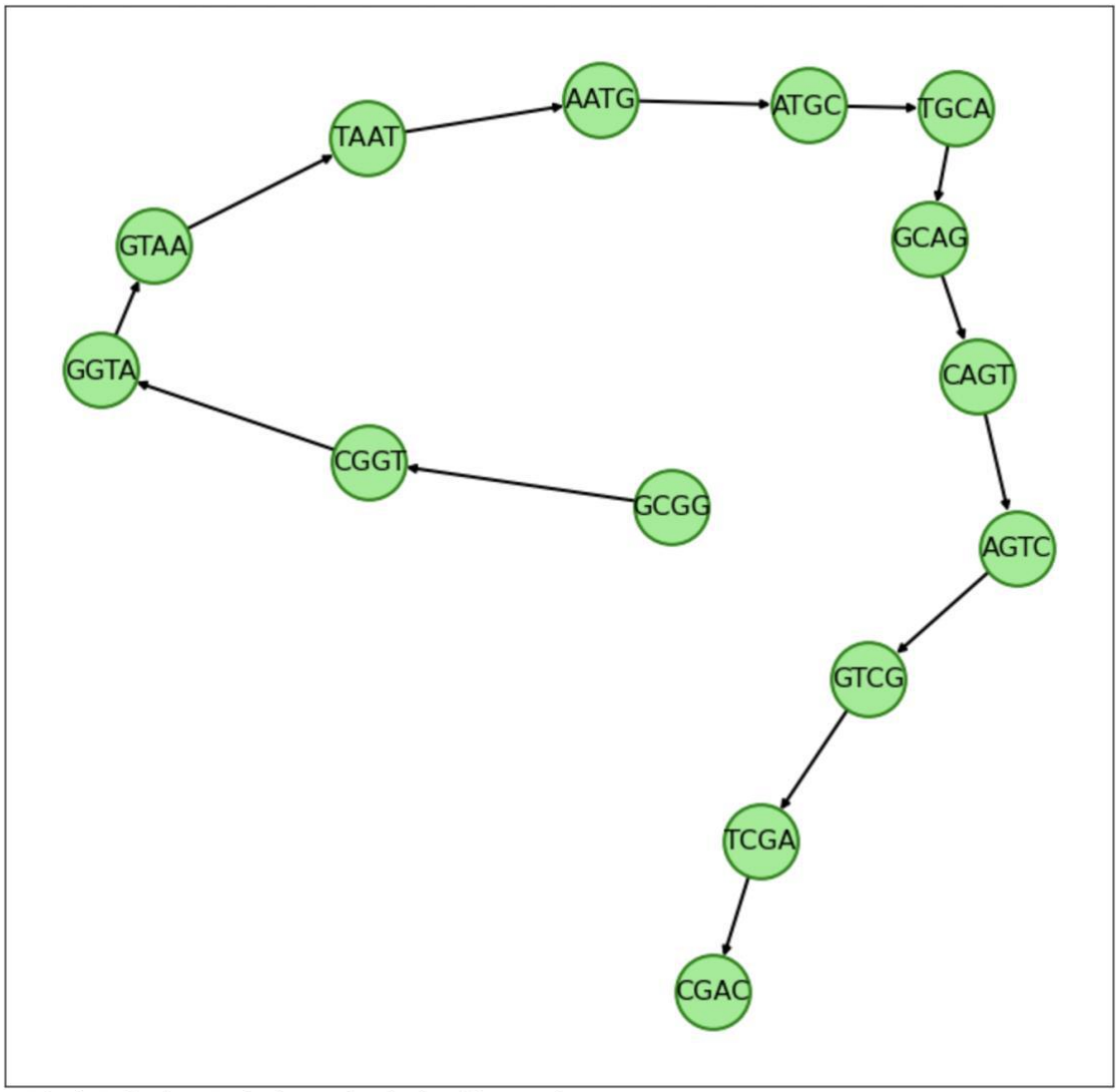
K=5

```
input1='GCGGTAATGCAGTCGAC'
Kmers=kmers(input1,5)
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['GCGG', 'CGGT', 'GGTA', 'GTAA', 'TAAT', 'AATG', 'ATGC', 'TGCA', 'GCAG', 'CAGT', 'AGTC', 'GTCG', 'TCGA', 'C
GAC']

edges = [('GCGG', 'CGGT'), ('CGGT', 'GGTA'), ('GGTA', 'GTAA'), ('GTAA', 'TAAT'), ('TAAT', 'AATG'), ('AATG', 'ATGC')
, ('ATGC', 'TGCA'), ('TGCA', 'GCAG'), ('GCAG', 'CAGT'), ('CAGT', 'AGTC'), ('AGTC', 'GTCG'), ('GTCG', 'TCGA'), ('TCG
A', 'CGAC')]
```




```
 Final Eulerian Path from the de-bruijn graph:

GCGG->CGGT->GGTA->GTAA->TAAT->AATG->ATGC->TGCA->GCAG->CAGT->AGTC->GTCG->TCGA->CGAC
['GCGG', 'CGGT', 'GGTA', 'GTAA', 'TAAT', 'AATG', 'ATGC', 'TGCA', 'GCAG', 'CAGT', 'AGTC', 'GTCG', 'TCGA', 'CGAC']
```

```
    Original String    : GCGGTAATGCAGTCGAC
 Reconstructed String : GCGGTAATGCAGTCGAC
```

C) TTGAGTGCCAGCGAGCTAGGTCAGT

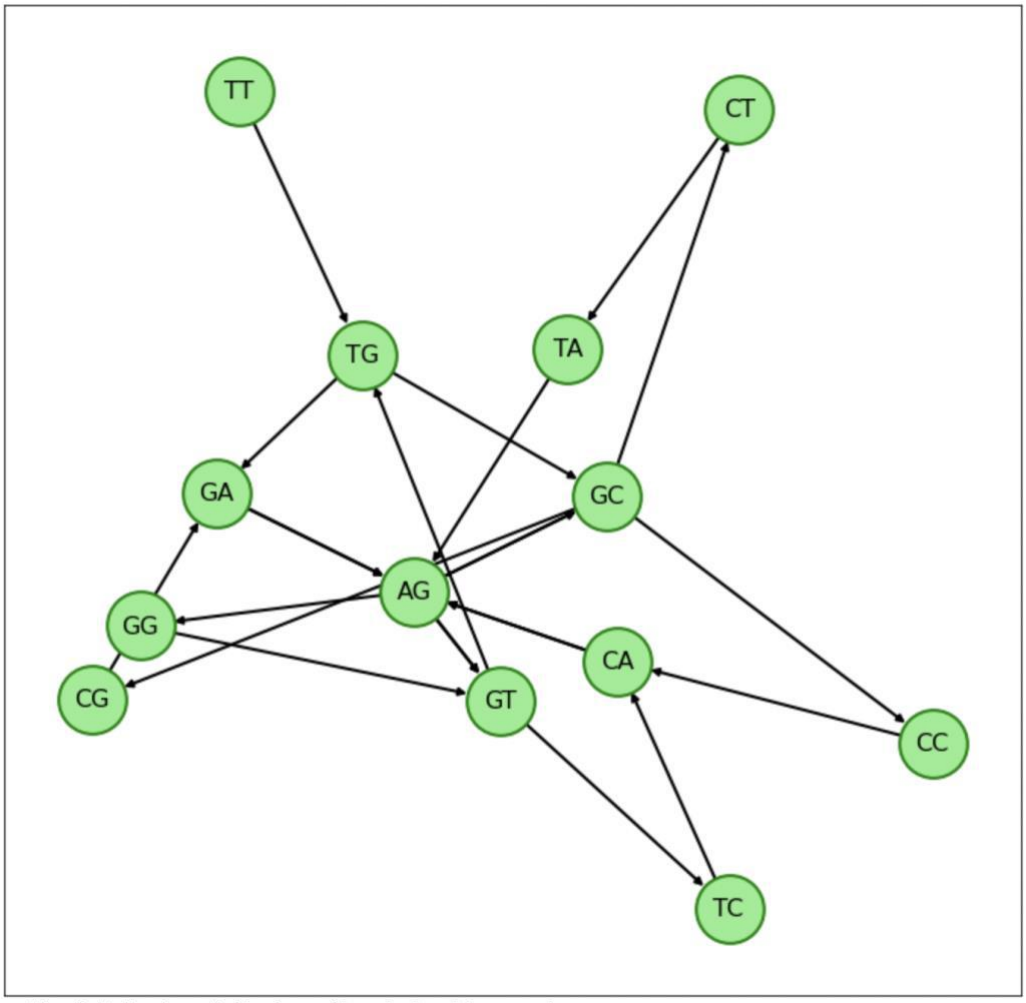
K=3

```
input1='TTGAGTGCCAGCGAGCTAGGTCAGT'
Kmers=kmers(input1,3)
```

```
---   DE-BRUIJN Graph   ---

nodes = ['TT', 'TG', 'GA', 'AG', 'GT', 'GC', 'CC', 'CA', 'CG', 'CT', 'TA', 'GG', 'TC']

edges = [('TT', 'TG'), ('TG', 'GA'), ('TG', 'GC'), ('GA', 'AG'), ('GA', 'AG'), ('AG', 'GT'), ('AG', 'GT'), ('AG', '
GC'), ('AG', 'GC'), ('AG', 'GG'), ('GT', 'TG'), ('GT', 'TC'), ('GC', 'CC'), ('GC', 'CG'), ('GC', 'CT'), ('CC', 'CA'
), ('CA', 'AG'), ('CA', 'AG'), ('CG', 'GA'), ('CT', 'TA'), ('TA', 'AG'), ('GG', 'GT'), ('TC', 'CA')]
```




```
Final Eulerian Path from the de-bruijn graph:

TT->TG->GA->AG->GT->TG->GC->CC->CA->AG->GT->TC->CA->AG->GC->CG->GA->AG->GC->CT->TA->AG->GG->GT
['TT', 'TG', 'GA', 'AG', 'GT', 'TG', 'GC', 'CC', 'CA', 'AG', 'GT', 'TC', 'CA', 'AG', 'GC', 'CG', 'GA', 'AG', 'GC',
'CT', 'TA', 'AG', 'GG', 'GT']

   Original String   : TTGAGTGCCAGCGAGCTAGGTCAGT
Reconstructed String : TTGAGTGCCAGTCAGCGAGCTAGGT
```

NOTE: In this, we find tht original and reconstructed strings are not matching, this is because the debruijn graph contains multiple eulerian paths, and one among them is being reconstructed

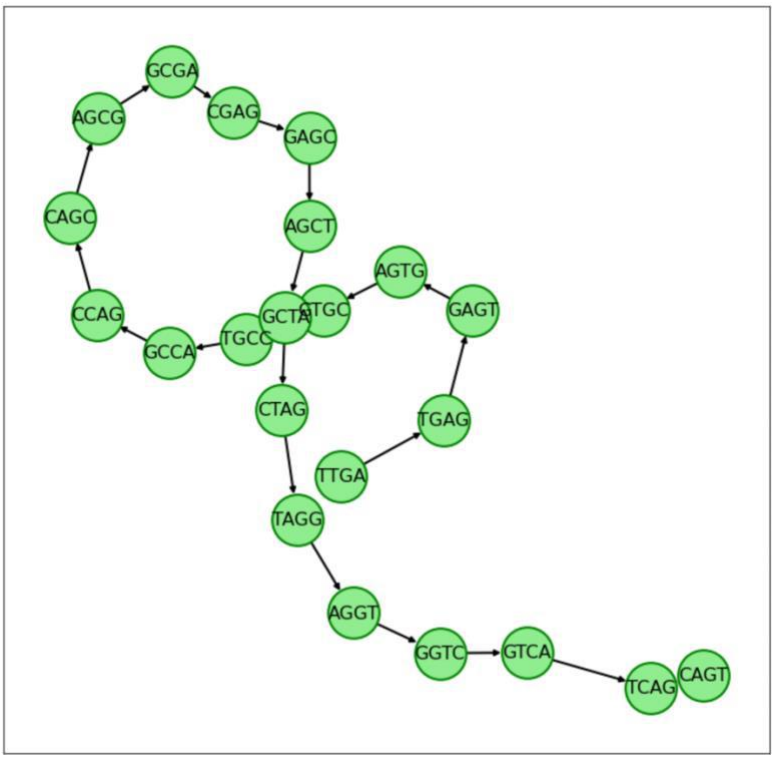
K=5

```
input1='TTGAGTGCCAGCGAGCTAGGTCAGT'
Kmers=kmers(input1,5)
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['TTGA', 'TGAG', 'GAGT', 'AGTG', 'GTGC', 'TGCC', 'GCCA', 'CCAG', 'CAGC', 'AGCG', 'GCGA', 'CGAG', 'GAGC', 'A
GCT', 'GCTA', 'CTAG', 'TAGG', 'AGGT', 'GGTC', 'GTCA', 'TCAG', 'CAGT']

edges = [('TTGA', 'TGAG'), ('TGAG', 'GAGT'), ('GAGT', 'AGTG'), ('AGTG', 'GTGC'), ('GTGC', 'TGCC'), ('TGCC', 'GCCA')
, ('GCCA', 'CCAG'), ('CCAG', 'CAGC'), ('CAGC', 'AGCG'), ('AGCG', 'GCGA'), ('GCGA', 'CGAG'), ('CGAG', 'GAGC'), ('GAG
C', 'AGCT'), ('AGCT', 'GCTA'), ('GCTA', 'CTAG'), ('CTAG', 'TAGG'), ('TAGG', 'AGGT'), ('AGGT', 'GGTC'), ('GGTC', 'GT
CA'), ('GTCA', 'TCAG'), ('TCAG', 'CAGT')]
```




```
Final Eulerian Path from the de-bruijn graph:

TTGA->TGAG->GAGT->AGTG->GTGC->TGCC->GCCA->CCAG->CAGC->AGCG->GCGA->CGAG->GAGC->AGCT->GCTA->CTAG->TAGG->AGGT->GGTC->G
TCA->TCAG->CAGT
['TTGA', 'TGAG', 'GAGT', 'AGTG', 'GTGC', 'TGCC', 'GCCA', 'CCAG', 'CAGC', 'AGCG', 'GCGA', 'CGAG', 'GAGC', 'AGCT', 'G
CTA', 'CTAG', 'TAGG', 'AGGT', 'GGTC', 'GTCA', 'TCAG', 'CAGT']

    Original String   : TTGAGTGCCAGCGAGCTAGGTCAGT
 Reconstructed String : TTGAGTGCCAGCGAGCTAGGTCAGT
```

2. Each member of the team has to select 20bp long unique segment from the sequence that has been selected from NCBI website. Mention the starting and stopping position of the unique segment selected in the report. (**eg.** Team member 1 can take first 20, Team member 2 can take next 20 and so on.)

String selected by our team

ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCC
AGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAA
GAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTTGACA
CAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGA
TACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCAACAAAACTGCTCGCCAGAACACTACGAGC
CACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

Output:

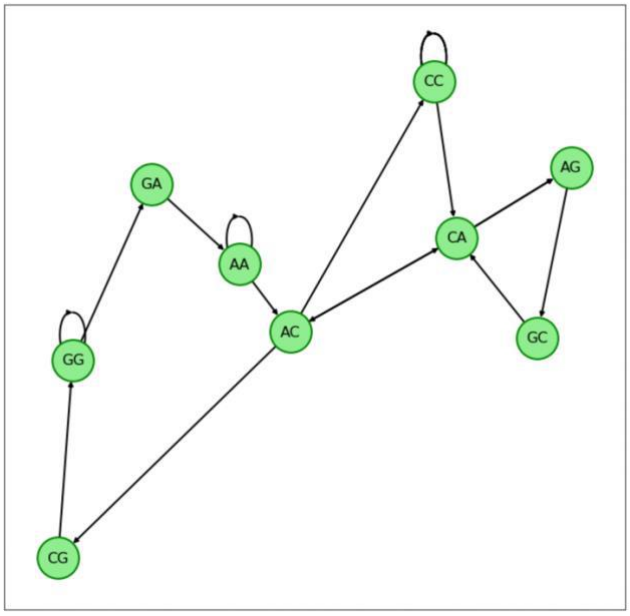
K=3

```
input1="ACCCCCACGGGAAACAGCAG"
Kmers=kmers(input1,3)
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['AC', 'CC', 'CA', 'CG', 'GG', 'GA', 'AA', 'AG', 'GC']

edges = [('AC', 'CC'), ('AC', 'CG'), ('AC', 'CA'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CA'), ('CA', '
AC'), ('CA', 'AG'), ('CA', 'AG'), ('CG', 'GG'), ('GG', 'GG'), ('GG', 'GA'), ('GA', 'AA'), ('AA', 'AA'), ('AA', 'AC'
), ('AG', 'GC'), ('GC', 'CA')]
```




```
 Final Eulerian Path from the de-bruijn graph:

AC->CC->CC->CC->CC->CA->AC->CG->GG->GG->GA->AA->AA->AC->CA->AG->GC->CA->AG
['AC', 'CC', 'CC', 'CC', 'CC', 'CA', 'AC', 'CG', 'GG', 'GG', 'GA', 'AA', 'AA', 'AC', 'CA', 'AG', 'GC', 'CA', 'AG']

   Original String    : ACCCCCACGGGAAACAGCAG
Reconstructed String : ACCCCCACGGGAAACAGCAG
```

Output:

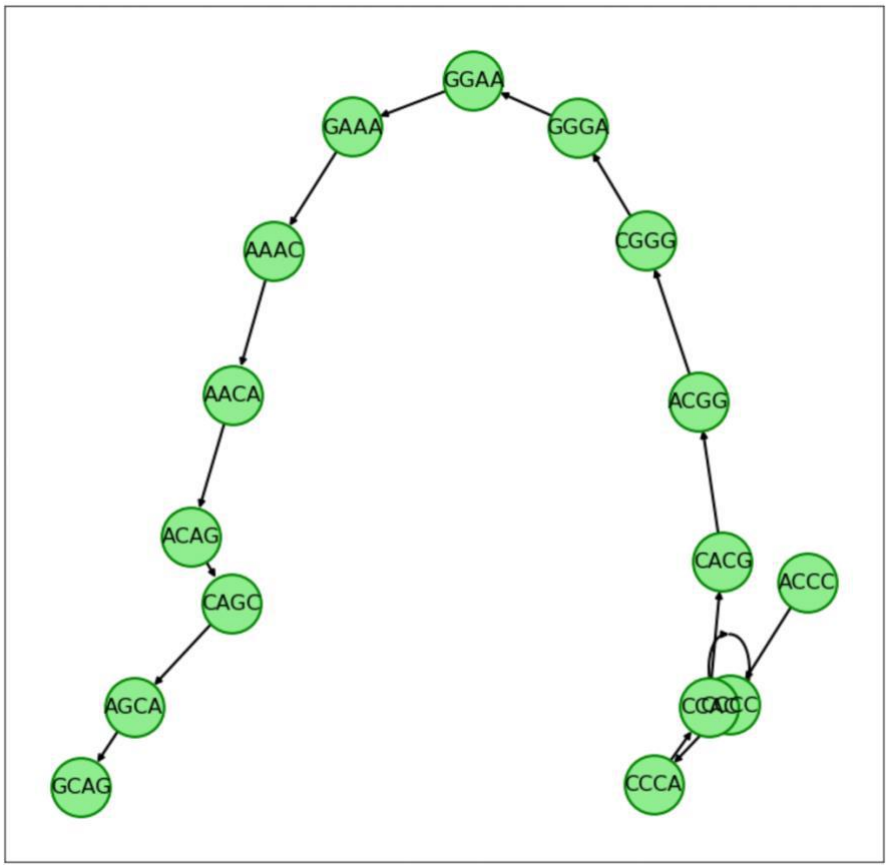
K=5

```
input1="ACCCCCACGGGAAACAGCAG"
Kmers=kmers(input1,5)
```

```
         ---   DE-BRUIJN Graph   ---

nodes = ['ACCC', 'CCCC', 'CCCA', 'CCAC', 'CACG', 'ACGG', 'CGGG', 'GGGA', 'GGAA', 'GAAA', 'AAAC', 'AACA', 'ACAG', 'C
AGC', 'AGCA', 'GCAG']

edges = [('ACCC', 'CCCC'), ('CCCC', 'CCCC'), ('CCCC', 'CCCA'), ('CCCA', 'CCAC'), ('CCAC', 'CACG'), ('CACG', 'ACGG')
, ('ACGG', 'CGGG'), ('CGGG', 'GGGA'), ('GGGA', 'GGAA'), ('GGAA', 'GAAA'), ('GAAA', 'AAAC'), ('AAAC', 'AACA'), ('AAC
A', 'ACAG'), ('ACAG', 'CAGC'), ('CAGC', 'AGCA'), ('AGCA', 'GCAG')]
```




```
Final Eulerian Path from the de-bruijn graph:

ACCC->CCCC->CCCC->CCCA->CCAC->CACG->ACGG->CGGG->GGGA->GGAA->GAAA->AAAC->AACA->ACAG->CAGC->AGCA->GCAG
['ACCC', 'CCCC', 'CCCC', 'CCCA', 'CCAC', 'CACG', 'ACGG', 'CGGG', 'GGGA', 'GGAA', 'GAAA', 'AAAC', 'AACA', 'ACAG', 'C
AGC', 'AGCA', 'GCAG']
```

```
   Original String   : ACCCCCACGGGAAACAGCAG
Reconstructed String : ACCCCCACGGGAAACAGCAG
```

Write a program in the programming language you prefer (Java or Python) to carry out string reconstruction using DeBruijn graph. (Attempt for k=3 & k=5). Also run the program for the whole sequence you have selected and include the output as well as the total run-time in the report. **(5 marks)**

String:

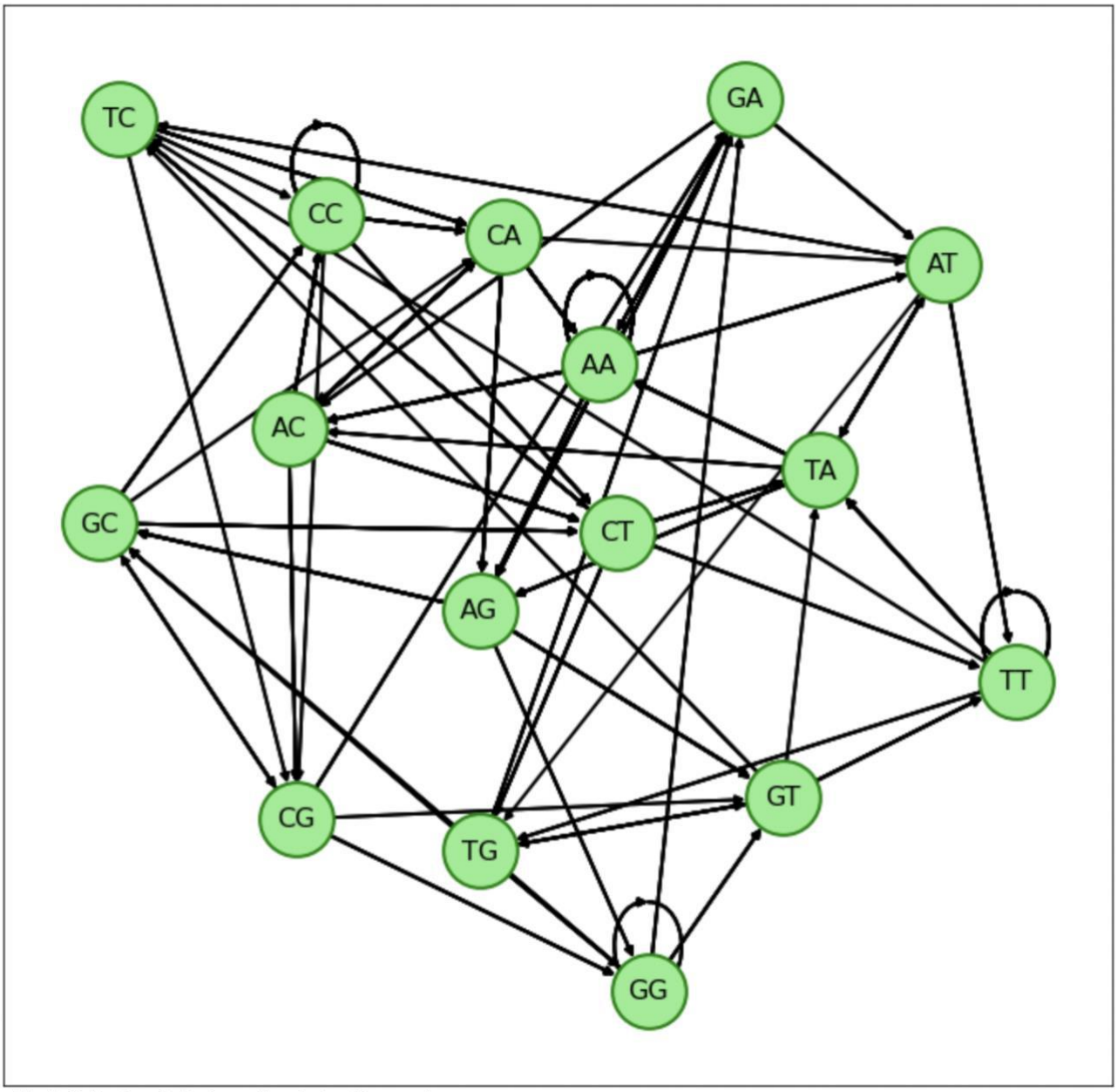
ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCC
AGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAA
GAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTTGACA
CAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGA
TACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCAACAAAACTGCTCGCCAGAACACTACGAGC
CACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

K=3

--- DE-BRUIJN Graph ---

nodes = ['AC', 'CC', 'CA', 'CG', 'GG', 'GA', 'AA', 'AG', 'GC', 'GT', 'TG', 'AT', 'TT', 'TA', 'CT', 'TC']

edges = [('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', '
CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CC'), ('AC', 'CG'), ('AC', 'CG'), ('AC', 'CG'), ('AC', 'CG'), ('AC', 'CG'
), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'),
('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CA'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('A
C', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('AC', 'CT'), ('CC', 'CC'), ('CC', 'CC'), ('CC',
'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC
'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CC'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'),
('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('CC', 'CA'), ('C
C', 'CT'), ('CC', 'CT'), ('CC', 'CT'), ('CC', 'CT'), ('CC', 'CT'), ('CC', 'CT'), ('CC', 'CT'), ('CC', 'CG'), ('CC',
'CG'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC
'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AC'), ('CA', 'AG'), ('CA', 'AG'), ('CA', 'AG'), ('CA', 'AG'),
('CA', 'AG'), ('CA', 'AG'), ('CA', 'AG'), ('CA', 'AG'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('C
A', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA', 'AA'), ('CA',
'AT'), ('CA', 'AT'), ('CG', 'GG'), ('CG', 'GG'), ('CG', 'GG'), ('CG', 'GG'), ('CG', 'GA'), ('CG', 'GA'), ('CG', 'GA
'), ('CG', 'GA'), ('CG', 'GT'), ('CG', 'GT'), ('CG', 'GC'), ('CG', 'GC'), ('GG', 'GG'), ('GG', 'GG'), ('GG', 'GG'),
('GG', 'GA'), ('GG', 'GA'), ('GG', 'GA'), ('GG', 'GT'), ('GG', 'GT'), ('GG', 'GT'), ('GG', 'GT'), ('GG', 'GC'), ('G
G', 'GC'), ('GG', 'GC'), ('GA', 'AA'), ('GA', 'AA'), ('GA', 'AA'), ('GA', 'AA'), ('GA', 'AA'), ('GA', 'AA'), ('GA',
'AT'), ('GA', 'AT'), ('GA', 'AT'), ('GA', 'AT'), ('GA', 'AT'), ('GA', 'AG'), ('GA', 'AG'), ('GA', 'AG'), ('GA', 'AG
'), ('GA', 'AC'), ('GA', 'AC'), ('GA', 'AC'), ('GA', 'AC'), ('GA', 'AC'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'),
('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('A
A', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA', 'AA'), ('AA',
'AA'), ('AA', 'AA'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC
'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'), ('AA', 'AC'),
('AA', 'AC'), ('AA', 'AC'), ('AA', 'AT'), ('AA', 'AT'), ('AA', 'AT'), ('AA', 'AT'), ('AA', 'AT'), ('AA', 'AT'), ('A
A', 'AT'), ('AA', 'AG'), ('AA', 'AG'), ('AA', 'AG'), ('AA', 'AG'), ('AA', 'AG'), ('AA', 'AG'), ('AA', 'AG'), ('AA',
'AG'), ('AA', 'AG'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC
'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GC'), ('AG', 'GT'), ('AG', 'GT'), ('AG', 'GT'), ('AG', 'GT'), ('AG', 'GT'),
('AG', 'GT'), ('AG', 'GT'), ('AG', 'GT'), ('AG', 'GG'), ('AG', 'GG'), ('AG', 'GG'), ('AG', 'GA'), ('AG', 'GA'), ('A
G', 'GA'), ('AG', 'GA'), ('AG', 'GA'), ('AG', 'GA'), ('AG', 'GA'), ('AG', 'GA'), ('GC', 'CA'), ('GC', 'CA'), ('GC',
'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CT'), ('GC', 'CC
'), ('GC', 'CC'), ('GC', 'CC'), ('GC', 'CC'), ('GC', 'CC'), ('GC', 'CC'), ('GC', 'CG'), ('GC', 'CG'), ('GC', 'CG'),
('GT', 'TG'), ('GT', 'TG'), ('GT', 'TG'), ('GT', 'TG'), ('GT', 'TG'), ('GT', 'TT'), ('GT', 'TT'), ('GT', 'TT'), ('G
T', 'TT'), ('GT', 'TT'), ('GT', 'TT'), ('GT', 'TC'), ('GT', 'TC'), ('GT', 'TC'), ('GT', 'TA'), ('GT', 'TA'), ('TG',
'GA'), ('TG', 'GA'), ('TG', 'GA'), ('TG', 'GA'), ('TG', 'GG'), ('TG', 'GG'), ('TG', 'GG'), ('TG', 'GG'), ('TG', 'GC
'), ('TG', 'GC'), ('TG', 'GC'), ('TG', 'GC'), ('TG', 'GT'), ('TG', 'GT'), ('AT', 'TT'), ('AT', 'TT'), ('AT', 'TT'),
('AT', 'TT'), ('AT', 'TA'), ('AT', 'TA'), ('AT', 'TA'), ('AT', 'TA'), ('AT', 'TA'), ('AT', 'TA'), ('AT', 'TA'), ('A
T', 'TA'), ('AT', 'TA'), ('AT', 'TC'), ('AT', 'TC'), ('AT', 'TC'), ('AT', 'TC'), ('AT', 'TG'), ('TT', 'TA'), ('TT',
'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA'), ('TT', 'TA
'), ('TT', 'TT'), ('TT', 'TT'), ('TT', 'TT'), ('TT', 'TT'), ('TT', 'TT'), ('TT', 'TT'), ('TT', 'TG'), ('TT', 'TG'),
('TT', 'TG'), ('TT', 'TC'), ('TT', 'TC'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('T
A', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA', 'AA'), ('TA',
'AA'), ('TA', 'AA'), ('TA', 'AG'), ('TA', 'AG'), ('TA', 'AG'), ('TA', 'AG'), ('TA', 'AG'), ('TA', 'AG'), ('TA', 'AG
'), ('TA', 'AG'), ('TA', 'AT'), ('TA', 'AT'), ('TA', 'AT'), ('TA', 'AT'), ('TA', 'AC'), ('TA', 'AC'), ('TA', 'AC'),
('TA', 'AC'), ('CT', 'TT'), ('CT', 'TT'), ('CT', 'TT'), ('CT', 'TT'), ('CT', 'TT'), ('CT', 'TA'), ('CT', 'TA'), ('C
T', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT', 'TA'), ('CT',
'TC'), ('CT', 'TC'), ('CT', 'TC'), ('CT', 'TC'), ('CT', 'TC'), ('CT', 'TC'), ('CT', 'TC'), ('CT', 'TG'), ('CT', 'TG
'), ('CT', 'TG'), ('CT', 'TG'), ('CT', 'TG'), ('TC', 'CA'), ('TC', 'CA'), ('TC', 'CA'), ('TC', 'CA'), ('TC', 'CA'),
('TC', 'CA'), ('TC', 'CA'), ('TC', 'CA'), ('TC', 'CA'), ('TC', 'CG'), ('TC', 'CG'), ('TC', 'CC'), ('TC', 'CC'), ('T
C', 'CC'), ('TC', 'CT'), ('TC', 'CT')]



Final Eulerian Path from the de-bruijn graph:

AC->CT->TA->AT->TA->AT->TA->AT->TA->AT->TA->AC->CT->TA->AC->CT->TC->CA->AA->AG->GA->AT->TC->CA->AA->AG->GA->AT->TC-
>CA->AA->AG->GA->AT->TC->CA->AA->AG->GA->AG->GA->AG->GA->AG->GA->AG->GA->AC->CT->TC->CA->AT->TC->CA->AT->TG->GC->CG
->GC->CG->GC->CC->CT->TC->CG->GA->AC->CT->TC->CG->GA->AC->CT->TC->CC->CT->TC->CC->CT->TC->CC->CT->TG->GC->CC->CG->G
A->AC->CT->TG->GC->CC->CG->GT->TC->CT->TG->GC->CG->GT->TC->CT->TG->GT->TA->AC->CT->TG->GT->TA->AC->CC->CC->CC->CC->
CC->CC->CC->CC->CC->CC->CC->CC->CC->CC->CC->CC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CC->CA-
>AC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CC->CA->AC->CG->GG->GG->GG->GG->GA->AA->AA->AA->AA->AA->AA->AA
->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AA->AC->CG->GG->GA->AA->AC->CG->GG->GA->AA->AC->CG->G
G->GT->TG->GA->AA->AC->CG->GA->AA->AC->CA->AC->CA->AG->GC->CA->AG->GC->CA->AG->GC->CT->TT->TA->AA->AC->CA->AG->GC->
CT->TT->TA->AA->AC->CA->AG->GC->CT->TT->TA->AA->AC->CA->AG->GC->CT->TT->TA->AA->AC->CA->AG->GC->CT->TT->TA->AA->AC-
>CA->AG->GC->CT->TA->AA->AC->CA->AA->AC->CA->AA->AC->CA->AA->AC->CA->AA->AC->CA->AA->AC->CT->TA->AA->AC->CT->TA->AA
->AT->TT->TA->AA->AT->TT->TA->AA->AT->TT->TA->AA->AT->TT->TA->AA->AT->TA->AA->AT->TA->AA->AT->TA->AA->AG->GC->CT->T
A->AG->GC->CT->TA->AG->GT->TG->GA->AA->AG->GT->TG->GA->AT->TA->AG->GT->TG->GA->AT->TA->AG->GT->TG->GG->GT->TT->TA->
AG->GT->TT->TT->TT->TT->TT->TT->TT->TG->GG->GT->TT->TG->GG->GT->TT->TG->GG->GC->CC->CT->TA->AG->GT->TT->TC->CA->AA-
>AG->GT->TT->TC->CA->AA->AG->GT->TC->CA->AA->AG->GG->GC->CC->CT->TA->AG->GG->GC->CC->CT->TA->AG->GG

Original String : ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCC
AGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTT
GACACAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCA
ACAAAACTGCTCGCCAGAACACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

Reconstructed String : ACTATATATATACTACTCAAGATCAAGATCAAGATCAAGAGAGAGAGACTCATCATGCGCGCCTCGACTCGACTCCTCCTCCTGCCGACTG
CCGTCTGCGTCTGTACTGTACCCCCCCCCCCCCCCCCCACCACCACCACCACCACCACCACCACCACCACGGGGGAAAAAAAAAAAAAAAAAAAAAAAAAACGGAACGGAACGGTGA
ACGAACACAGCAGCAGCTTAACAGCTTAACAGCTTAACAGCTTAACAGCTTAACAGCTAACAACAACAACAACAACTAACTAATTAATTAATTAATTAATAATAATAAGCTAGCT
AGTGAAGTGATAGTGATAGTGGTTAGTTTTTTTTGGTTGGTTGGCCTAGTTCAAGTTCAAGTCAAGGCCTAGGCCTAGG
Run Time: 1.6111957910002275

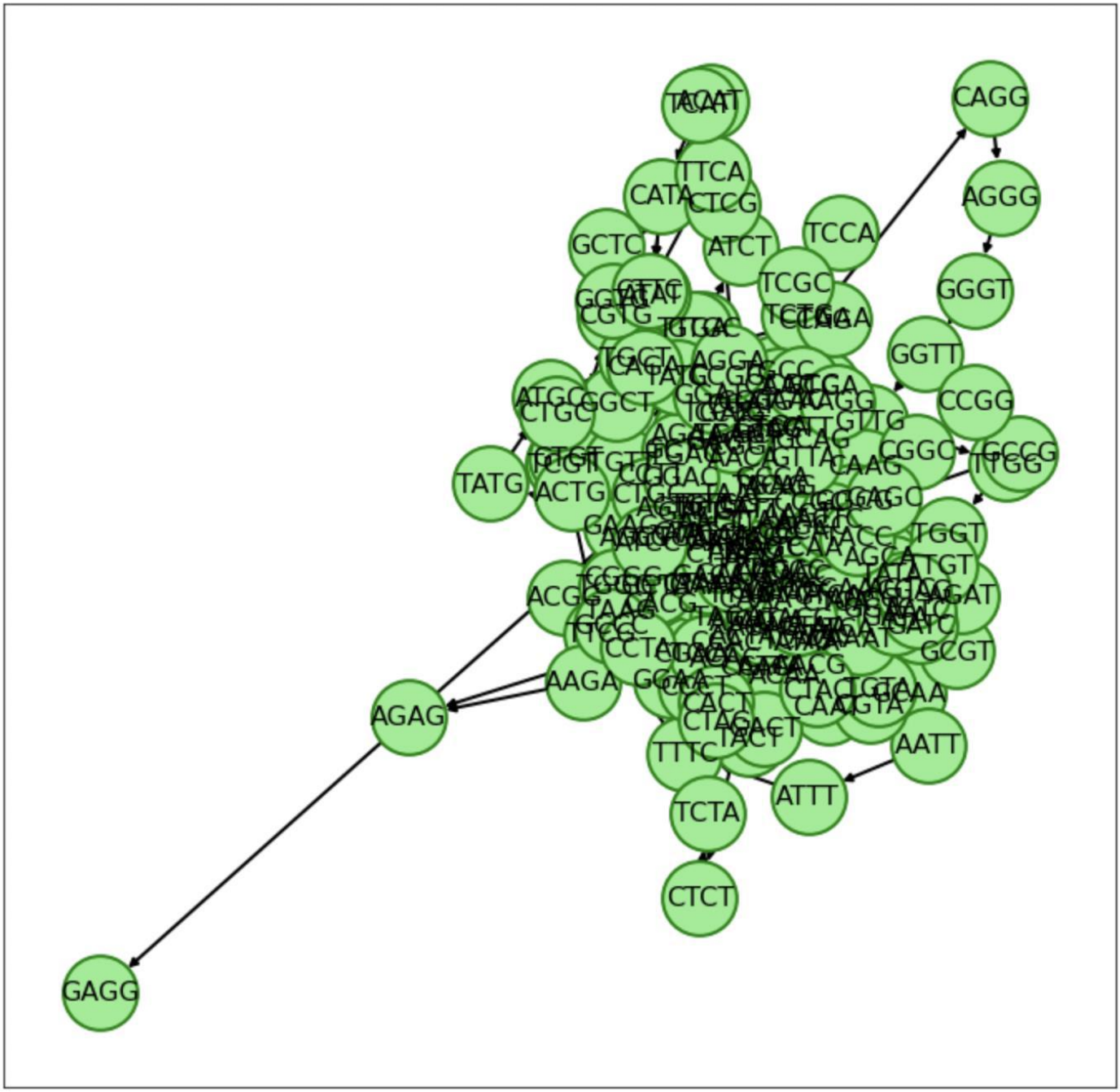
NOTE: In this, we find tht original and reconstructed strings are not matching, this is because the debruijn graph contains multiple eulerian paths, and one among them is being reconstructed

K=5

```
         ---   DE-BRUIJN Graph   ---

nodes = ['ACCC', 'CCCC', 'CCCA', 'CCAC', 'CACG', 'ACGG', 'CGGG', 'GGGA', 'GGAA', 'GAAA', 'AAAC', 'AACA', 'ACAG', 'C
AGC', 'AGCA', 'GCAG', 'CAGT', 'AGTG', 'GTGA', 'TGAT', 'GATT', 'ATTA', 'TTAA', 'TAAC', 'AACC', 'ACCT', 'CCTT', 'CTTT
', 'TTTA', 'TTAG', 'TAGC', 'GCAA', 'CAAT', 'AATA', 'ATAA', 'TAAA', 'AACG', 'ACGA', 'CGAA', 'AAAG', 'AAGT', 'AGTT',
'GTTT', 'AACT', 'ACTA', 'CTAA', 'TAAG', 'AAGC', 'AGCT', 'GCTA', 'CTAT', 'TATA', 'ATAC', 'TACT', 'CCAG', 'CAGG', 'AG
GG', 'GGGT', 'GGTT', 'GTTG', 'TTGG', 'TGGT', 'GGTC', 'GTCA', 'TCAA', 'AATT', 'ATTT', 'TTTC', 'TTCG', 'TCGT', 'CGTG'
, 'GTGC', 'TGCC', 'GCCA', 'AGCC', 'CACC', 'ACCG', 'CCGC', 'CGCG', 'GCGG', 'CGGT', 'TCAC', 'CACA', 'ACAC', 'CGAT', '
CCAA', 'CAAG', 'AGTC', 'ATAG', 'TAGA', 'AGAA', 'GAAG', 'GCCG', 'CCGG', 'CGGC', 'GGCG', 'GCGT', 'CGTA', 'GTAA', 'AAG
A', 'AGAG', 'GAGT', 'GTGT', 'TGTT', 'TTTT', 'AGAT', 'GATC', 'ATCA', 'CCCT', 'CCTC', 'CTCC', 'TCCC', 'AAAA', 'ACTC',
'CTCA', 'CCTG', 'CTGA', 'TGAG', 'TTGT', 'TGTA', 'TCCA', 'TTGA', 'TGAC', 'GACA', 'ACAA', 'CAAA', 'AAAT', 'AGAC', 'GA
CT', 'CTAC', 'TACG', 'GTGG', 'TGGC', 'GGCT', 'GCTT', 'ACAT', 'CATA', 'ATAT', 'TATC', 'ATCT', 'TCTG', 'TGAA', 'GAAC'
, 'GACC', 'ACTG', 'CTGG', 'TGGG', 'GGAT', 'GATA', 'TACC', 'CACT', 'TATG', 'ATGC', 'TGCT', 'CTTA', 'GCCC', 'CCTA', '
CAAC', 'GTTA', 'AATC', 'CTGC', 'GCTC', 'CTCG', 'TCGC', 'CGCC', 'CAGA', 'CGAG', 'GAGC', 'AAGG', 'AGGA', 'GGAC', 'GGT
G', 'CTTC', 'TTCA', 'TCAT', 'ATCC', 'CTCT', 'TCTA', 'CTAG', 'GAGG']

edges = [('ACCC', 'CCCC'), ('ACCC', 'CCCC'), ('ACCC', 'CCCC'), ('ACCC', 'CCCC'), ('ACCC', 'CCCA'), ('ACCC', 'CCCA')
, ('CCCC', 'CCCC'), ('CCCC', 'CCCC'), ('CCCC', 'CCCA'), ('CCCC', 'CCCA'), ('CCCC', 'CCCA'), ('CCCC', 'CCCA'), ('CCC
C', 'CCCT'), ('CCCA', 'CCAC'), ('CCCA', 'CCAC'), ('CCCA', 'CCAG'), ('CCCA', 'CCAA'), ('CCCA', 'CCAA'), ('CCCA', 'CC
AA'), ('CCAC', 'CACG'), ('CCAC', 'CACC'), ('CCAC', 'CACT'), ('CCAC', 'CACA'), ('CACG', 'ACGG'), ('CACG', 'ACGA'), (
'ACGG', 'CGGG'), ('CGGG', 'GGGA'), ('GGGA', 'GGAA'), ('GGGA', 'GGAT'), ('GGAA', 'GAAA'), ('GAAA', 'AAAC'), ('GAAA',
'AAAG'), ('GAAA', 'AAAG'), ('AAAC', 'AACA'), ('AAAC', 'AACG'), ('AAAC', 'AACT'), ('AAAC', 'AACT'), ('AAAC', 'AACT')
, ('AAAC', 'AACT'), ('AAAC', 'AACT'), ('AAAC', 'AACC'), ('AACA', 'ACAG'), ('AACA', 'ACAG'), ('AACA', 'ACAT'), ('AAC
A', 'ACAC'), ('AACA', 'ACAC'), ('AACA', 'ACAA'), ('ACAG', 'CAGC'), ('ACAG', 'CAGC'), ('ACAG', 'CAGT'), ('CAGC', 'AG
CA'), ('CAGC', 'AGCC'), ('CAGC', 'AGCT'), ('AGCA', 'GCAG'), ('AGCA', 'GCAA'), ('GCAG', 'CAGT'), ('CAGT', 'AGTG'), (
'CAGT', 'AGTT'), ('CAGT', 'AGTT'), ('AGTG', 'GTGA'), ('AGTG', 'GTGT'), ('AGTG', 'GTGG'), ('GTGA', 'TGAT'), ('TGAT',
'GATT'), ('GATT', 'ATTA'), ('GATT', 'ATTA'), ('GATT', 'ATTA'), ('ATTA', 'TTAA'), ('ATTA', 'TTAA'), ('ATTA', 'TTAG')
, ('TTAA', 'TAAC'), ('TTAA', 'TAAC'), ('TTAA', 'TAAC'), ('TTAA', 'TAAC'), ('TTAA', 'TAAA'), ('TTAA', 'TAAA'), ('TAA
C', 'AACC'), ('TAAC', 'AACC'), ('TAAC', 'AACC'), ('TAAC', 'AACT'), ('TAAC', 'AACA'), ('AACC', 'ACCT'), ('AACC', 'AC
CT'), ('AACC', 'ACCC'), ('AACC', 'ACCC'), ('ACCT', 'CCTT'), ('ACCT', 'CCTG'), ('ACCT', 'CCTG'), ('ACCT', 'CCTC'), (
'CCTT', 'CTTT'), ('CTTT', 'TTTA'), ('CTTT', 'TTTA'), ('TTTA', 'TTAG'), ('TTTA', 'TTAG'), ('TTTA', 'TTAA'), ('TTTA',
'TTAA'), ('TTAG', 'TAGC'), ('TTAG', 'TAGC'), ('TTAG', 'TAGA'), ('TTAG', 'TAGA'), ('TAGC', 'AGCA'), ('TAGC', 'AGCT')
, ('TAGC', 'AGCC'), ('GCAA', 'CAAT'), ('CAAT', 'AATA'), ('CAAT', 'AATA'), ('CAAT', 'AATA'), ('CAAT', 'AATA'), ('CAA
T', 'AATT'), ('AATA', 'ATAA'), ('AATA', 'ATAA'), ('AATA', 'ATAG'), ('AATA', 'ATAG'), ('AATA', 'ATAG'), ('ATAA', 'TA
AA'), ('ATAA', 'TAAA'), ('TAAA', 'AAAC'), ('TAAA', 'AAAC'), ('TAAA', 'AAAG'), ('TAAA', 'AAAG'), ('TAAA', 'AAAA'), (
'TAAA', 'AAAA'), ('TAAA', 'AAAA'), ('TAAA', 'AAAT'), ('AACG', 'ACGA'), ('ACGA', 'CGAA'), ('ACGA', 'CGAA'), ('ACGA',
'CGAT'), ('ACGA', 'CGAG'), ('CGAA', 'GAAA'), ('CGAA', 'GAAA'), ('AAAG', 'AAGT'), ('AAAG', 'AAGT'), ('AAAG', 'AAGA')
, ('AAAG', 'AAGC'), ('AAAG', 'AAGG'), ('AAGT', 'AGTT'), ('AAGT', 'AGTC'), ('AAGT', 'AGTG'), ('AGTT', 'GTTT'), ('AGT
T', 'GTTG'), ('AGTT', 'GTTG'), ('AGTT', 'GTTA'), ('GTTT', 'TTTA'), ('GTTT', 'TTTT'), ('AACT', 'ACTA'), ('AACT', 'AC
TC'), ('AACT', 'ACTC'), ('AACT', 'ACTC'), ('AACT', 'ACTG'), ('AACT', 'ACTG'), ('ACTA', 'CTAA'), ('ACTA', 'CTAA'), (
'ACTA', 'CTAC'), ('ACTA', 'CTAC'), ('ACTA', 'CTAT'), ('CTAA', 'TAAG'), ('CTAA', 'TAAG'), ('CTAA', 'TAAC'), ('CTAA',
'TAAA'), ('CTAA', 'TAAA'), ('TAAG', 'AAGC'), ('TAAG', 'AAGA'), ('AAGC', 'AGCT'), ('AAGC', 'AGCT'), ('AAGC', 'AGCC')
, ('AGCT', 'GCTA'), ('AGCT', 'GCTA'), ('AGCT', 'GCTA'), ('AGCT', 'GCTT'), ('GCTA', 'CTAT'), ('GCTA', 'CTAA'), ('GCT
A', 'CTAA'), ('CTAT', 'TATA'), ('CTAT', 'TATG'), ('TATA', 'ATAC'), ('ATAC', 'TACT'), ('ATAC', 'TACC'), ('TACT', 'AC
TA'), ('CCAG', 'CAGG'), ('CCAG', 'CAGC'), ('CCAG', 'CAGT'), ('CCAG', 'CAGA'), ('CAGG', 'AGGG'), ('AGGG', 'GGGT'), (
'GGGT', 'GGTT'), ('GGTT', 'GTTG'), ('GTTG', 'TTGG'), ('GTTG', 'TTGT'), ('GTTG', 'TTGA'), ('TTGG', 'TGGT'), ('TGGT',
'GGTC'), ('GGTC', 'GTCA'), ('GGTC', 'GTCA'), ('GTCA', 'TCAA'), ('GTCA', 'TCAA'), ('GTCA', 'TCAC'), ('TCAA', 'CAAT')
, ('TCAA', 'CAAT'), ('TCAA', 'CAAC'), ('TCAA', 'CAAC'), ('TCAA', 'CAAA'), ('AATT', 'ATTT'), ('ATTT', 'TTTC'), ('TTT
C', 'TTCG'), ('TTCG', 'TCGT'), ('TCGT', 'CGTG'), ('CGTG', 'GTGC'), ('GTGC', 'TGCC'), ('GTGC', 'TGCT'), ('TGCC', 'GC
CA'), ('GCCA', 'CCAG'), ('GCCA', 'CCAG'), ('GCCA', 'CCAC'), ('GCCA', 'CCAC'), ('AGCC', 'GCCA'), ('AGCC', 'GCCA'), (
'AGCC', 'GCCG'), ('AGCC', 'GCCC'), ('CACC', 'ACCG'), ('CACC', 'ACCC'), ('CACC', 'ACCT'), ('ACCG', 'CCGC'), ('CCGC',
'CGCG'), ('CGCG', 'GCGG'), ('GCGG', 'CGGT'), ('GCGG', 'CGGT'), ('CGGT', 'GGTC'), ('CGGT', 'GGTG'), ('TCAC', 'CACA')
, ('TCAC', 'CACC'), ('TCAC', 'CACC'), ('CACA', 'ACAC'), ('CACA', 'ACAC'), ('CACA', 'ACAA'), ('CACA', 'ACAA'), ('CAC
A', 'ACAG'), ('ACAC', 'CACG'), ('ACAC', 'CACA'), ('ACAC', 'CACA'), ('ACAC', 'CACA'), ('ACAC', 'CACT'), ('CGAT', 'GA
TT'), ('CCAA', 'CAAG'), ('CCAA', 'CAAT'), ('CCAA', 'CAAA'), ('CAAG', 'AAGT'), ('AGTC', 'GTCA'), ('ATAG', 'TAGA'), (
'ATAG', 'TAGA'), ('ATAG', 'TAGC'), ('TAGA', 'AGAA'), ('TAGA', 'AGAT'), ('TAGA', 'AGAT'), ('TAGA', 'AGAC'), ('TAGA',
'AGAG'), ('AGAA', 'GAAG'), ('AGAA', 'GAAC'), ('GAAG', 'AAGC'), ('GCCG', 'CCGG'), ('CCGG', 'CGGC'), ('CGGC', 'GGCG')
, ('GGCG', 'GCGT'), ('GGCG', 'GCGG'), ('GCGT', 'CGTA'), ('CGTA', 'GTAA'), ('GTAA', 'TAAA'), ('GTAA', 'TAAA'), ('AAG
A', 'AGAG'), ('AAGA', 'AGAC'), ('AGAG', 'GAGT'), ('AGAG', 'GAGG'), ('GAGT', 'AGTG'), ('GAGT', 'AGTT'), ('GTGT', 'TG
TT'), ('TGTT', 'GTTT'), ('TTTT', 'TTTA'), ('AGAT', 'GATC'), ('AGAT', 'GATA'), ('GATC', 'ATCA'), ('ATCA', 'TCAC'), (
'ATCA', 'TCAA'), ('CCCT', 'CCTC'), ('CCCT', 'CCTC'), ('CCCT', 'CCTA'), ('CCTC', 'CTCC'), ('CCTC', 'CTCA'), ('CCTC',
'CTCT'), ('CTCC', 'TCCC'), ('CTCC', 'TCCA'), ('TCCC', 'CCCC'), ('TCCC', 'CCCT'), ('AAAA', 'AAAC'), ('AAAA', 'AAAC')
, ('AAAA', 'AAAC'), ('AAAA', 'AAAC'), ('AAAA', 'AAAA'), ('AAAA', 'AAAA'), ('AAAA', 'AAAT'), ('ACTC', 'CTCA'), ('ACT
C', 'CTCA'), ('ACTC', 'CTCC'), ('CTCA', 'TCAC'), ('CTCA', 'TCAA'), ('CTCA', 'TCAA'), ('CCTG', 'CTGA'), ('CCTG', 'CT
GG'), ('CTGA', 'TGAG'), ('CTGA', 'TGAA'), ('TGAG', 'GAGT'), ('TTGT', 'TGTA'), ('TGTA', 'GTAA'), ('TCCA', 'CCAG'), (
'TTGA', 'TGAC'), ('TGAC', 'GACA'), ('GACA', 'ACAC'), ('ACAA', 'CAAA'), ('ACAA', 'CAAA'), ('ACAA', 'CAAT'), ('CAAA',
'AAAA'), ('CAAA', 'AAAA'), ('CAAA', 'AAAC'), ('CAAA', 'AAAG'), ('AAAT', 'AATA'), ('AAAT', 'AATC'), ('AGAC', 'GACT')
, ('AGAC', 'GACC'), ('GACT', 'ACTA'), ('CTAC', 'TACG'), ('CTAC', 'TACG'), ('TACG', 'ACGA'), ('TACG', 'ACGA'), ('GTG
G', 'TGGC'), ('TGGC', 'GGCT'), ('TGGC', 'GGCG'), ('GGCT', 'GCTT'), ('GCTT', 'CTTT'), ('GCTT', 'CTTA'), ('GCTT', 'CT
TA'), ('GCTT', 'CTTC'), ('ACAT', 'CATA'), ('CATA', 'ATAT'), ('CATA', 'ATAT'), ('ATAT', 'TATC'), ('ATAT', 'TATC'), (
'TATC', 'ATCT'), ('TATC', 'ATCC'), ('ATCT', 'TCTG'), ('TCTG', 'CTGA'), ('TGAA', 'GAAC'), ('GAAC', 'AACA'), ('GAAC',
'AACA'), ('GACC', 'ACCC'), ('GACC', 'ACCT'), ('ACTG', 'CTGG'), ('ACTG', 'CTGC'), ('CTGG', 'TGGG'), ('CTGG', 'TGGC')
, ('TGGG', 'GGGA'), ('GGAT', 'GATT'), ('GATA', 'ATAC'), ('TACC', 'ACCC'), ('CACT', 'ACTA'), ('CACT', 'ACTA'), ('TAT
G', 'ATGC'), ('ATGC', 'TGCT'), ('TGCT', 'GCTT'), ('TGCT', 'GCTT'), ('TGCT', 'GCTC'), ('CTTA', 'TTAG'), ('CTTA', 'TT
AA'), ('GCCC', 'CCCT'), ('CCTA', 'CTAA'), ('CAAC', 'AACA'), ('CAAC', 'AACA'), ('GTTA', 'TTAA'), ('AATC', 'ATCA'), (
'CTGC', 'TGCT'), ('GCTC', 'CTCG'), ('CTCG', 'TCGC'), ('TCGC', 'CGCC'), ('CGCC', 'GCCA'), ('CAGA', 'AGAA'), ('CGAG',
'GAGC'), ('GAGC', 'AGCC'), ('AAGG', 'AGGA'), ('AGGA', 'GGAC'), ('GGAC', 'GACC'), ('GGTG', 'GTGC'), ('CTTC', 'TTCA')
, ('TTCA', 'TCAT'), ('TCAT', 'CATA'), ('ATCC', 'TCCC'), ('CTCT', 'TCTA'), ('TCTA', 'CTAG'), ('CTAG', 'TAGA')]
```



Final Eulerian Path from the de-bruijn graph:

ACCC->CCCC->CCCC->CCCC->CCCA->CCAC->CACG->ACGG->CGGG->GGGA->GGAA->GAAA->AAAC->AACA->ACAG->CAGC->AGCA->GCAG->CAGT->AGTG->GTGA->TGAT->GATT->ATTA->TTAA->TAAC->AACC->ACCT->CCTT->CTTT->TTTA->TTAG->TAGC->AGCA->GCAA->CAAT->AATA->ATAA->TAAA->AAAC->AACG->ACGA->CGAA->GAAA->AAAG->AAGT->AGTT->GTTT->TTTA->TTAG->TAGC->AGCT->GCTA->CTAT->TATA->ATAC->TACT->ACTA->CTAA->TAAG->AAGC->AGCT->GCTA->CTAA->TAAG->AAGA->AGAG->GAGT->AGTG->GTGT->TGTT->GTTT->TTTT->TTTA->TTAA->TAAC->AACC->ACCT->CCTG->CTGA->TGAG->GAGT->AGTT->GTTG->TTGG->TGGT->GGTC->GTCA->TCAA->CAAT->AATA->ATAA->TAAA->AAAC->AACT->ACTA->CTAA->TAAC->AACC->ACCC->CCCC->CCCA->CCAC->CACC->ACCG->CCGC->CGCG->GCGG->CGGT->GGTC->GTCA->TCAA->CAAT->AATA->ATAG->TAGA->AGAA->GAAG->AAGC->AGCT->GCTA->CTAA->TAAA->AAAG->AAGT->AGTC->GTCA->TCAC->CACA->ACAC->CACG->ACGA->CGAA->GAAA->AAAG->AAGA->AGAC->GACT->ACTA->CTAC->TACG->ACGA->CGAT->GATT->ATTA->TTAA->TAAC->AACT->ACTC->CTCA->TCAC->CACC->ACCC->CCCC->CCCA->CCAG->CAGG->AGGG->GGGT->GGTT->GTTG->TTGT->TGTA->GTAA->TAAA->AAAG->AAGC->AGCC->GCCA->CCAG->CAGC->AGCC->GCCA->CCAG->CAGT->AGTT->GTTG->TTGA->TGAC->GACA->ACAC->CACA->ACAC->CACA->ACAA->CAAA->AAAA->AAAC->AACT->ACTC->CTCA->TCAA->CAAC->AACA->ACAG->CAGC->AGCT->GCTT->CTTT->TTTA->TTAA->TAAC->AACA->ACAT->CATA->ATAT->TATC->ATCT->TCTG->CTGA->TGAA->GAAC->AACA->ACAC->CACA->ACAA->CAAA->AAAA->AAAC->AACT->ACTC->CTCC->TCCC->CCCC->CCCA->CCAA->CAAG->AAGT->AGTG->GTGG->TGGC->GGCT->GCTT->CTTA->TTAG->TAGA->AGAT->GATC->ATCA->TCAC->CACC->ACCT->CCTG->CTGG->TGGG->GGGA->GGAT->GATT->ATTA->TTAG->TAGA->AGAT->GATA->ATAC->TACC->ACCC->CCCC->CCCT->CCTC->CTCC->TCCA->CCAG->CAGA->AGAA->GAAC->AACA->ACAC->CACT->ACTA->CTAC->TACG->ACGA->CGAG->GAGC->AGCC->GCCG->CCGG->CGGC->GGCG->GCGT->CGTA->GTAA->TAAA->AAAA->AAAC->AACT->ACTG->CTGG->TGGC->GGCG->GCGG->CGGT->GGTG->GTGC->TGCC->GCCA->CCAC->CACT->ACTA->CTAT->TATG->ATGC->TGCT->GCTT->CTTA->TTAA->TAAA->AAAA->AAAC->AACT->ACTG->CTGC->TGCT->GCTT->CTTC->TTCA->TCAT->CATA->ATAT->TATC->ATCC->TCCC->CCCT->CCTC->CTCA->TCAA->CAAC->AACA->ACAA->CAAT->AATA->ATAG->TAGA->AGAC->GACC->ACCC->CCCA->CCAA->CAAT->AATT->ATTT->TTTC->TTCG->TCGT->CGTG->GTGC->TGCT->GCTC->CTCG->TCGC->CGCC->GCCA->CCAC->CACA->ACAG->CAGT->AGTT->GTTA->TTAA->TAAA->AAAA->AAAA->AAAA->AAAT->AATA->ATAG->TAGC->AGCC->GCCC->CCCT->CCTA->CTAA->TAAA->AAAT->AATC->ATCA->TCAA->CAAA->AAAC->AACC->ACCC->CCCA->CCAA->CAAA->AAAG->AAGG->AGGA->GGAC->GACC->ACCT->CCTC->CTCT->TCTA->CTAG->TAGA->AGAG->GAGG

```
    Original String   : ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCC
AGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTT
GACACAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCA
ACAAAACTGCTCGCCAGAACACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

 Reconstructed String : ACCCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAGCTATACTAAGCTAAGAGTGTTTTAACCTGAGTTGGTCA
ATAAACTAACCCCACCGCGGTCAATAGAAGCTAAAGTCACACGAAAGACTACGATTAACTCACCCCAGGGTTGTAAAGCCAGCCAGTTGACACACAAAACTCAACAGCTTTAACA
TATCTGAACACAAAACTCCCCAAGTGGCTTAGATCACCTGGGATTAGATACCCCTCCAGAACACTACGAGCCGGCGTAAAACTGGCGGTGCCACTATGCTTAAAACTGCTTCATA
TCCCTCAACAATAGACCCAATTTCGTGCTCGCCACAGTTAAAAAATAGCCCTAAATCAAACCCAAAGGACCTCTAGAGG
Run Time:  1.7081665420000718
```

End
