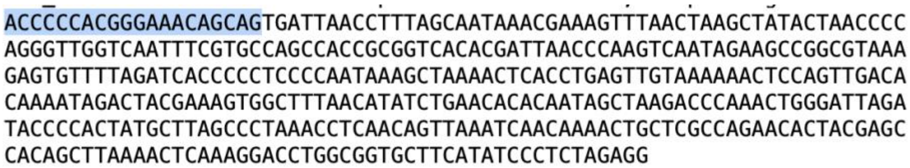
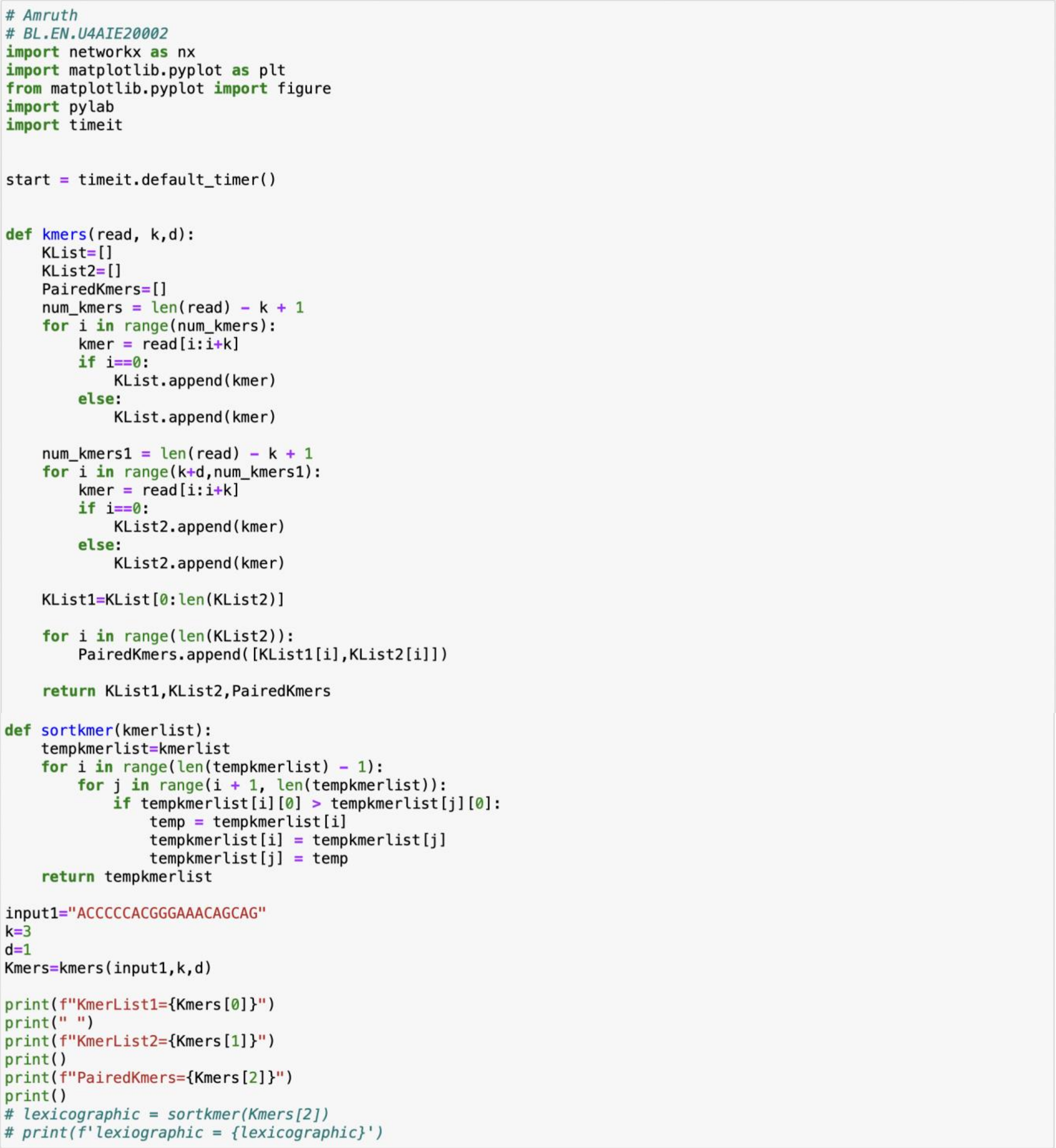
19BIO201 Intelligence of Biological systems
Assignment5- Paired Debruijn

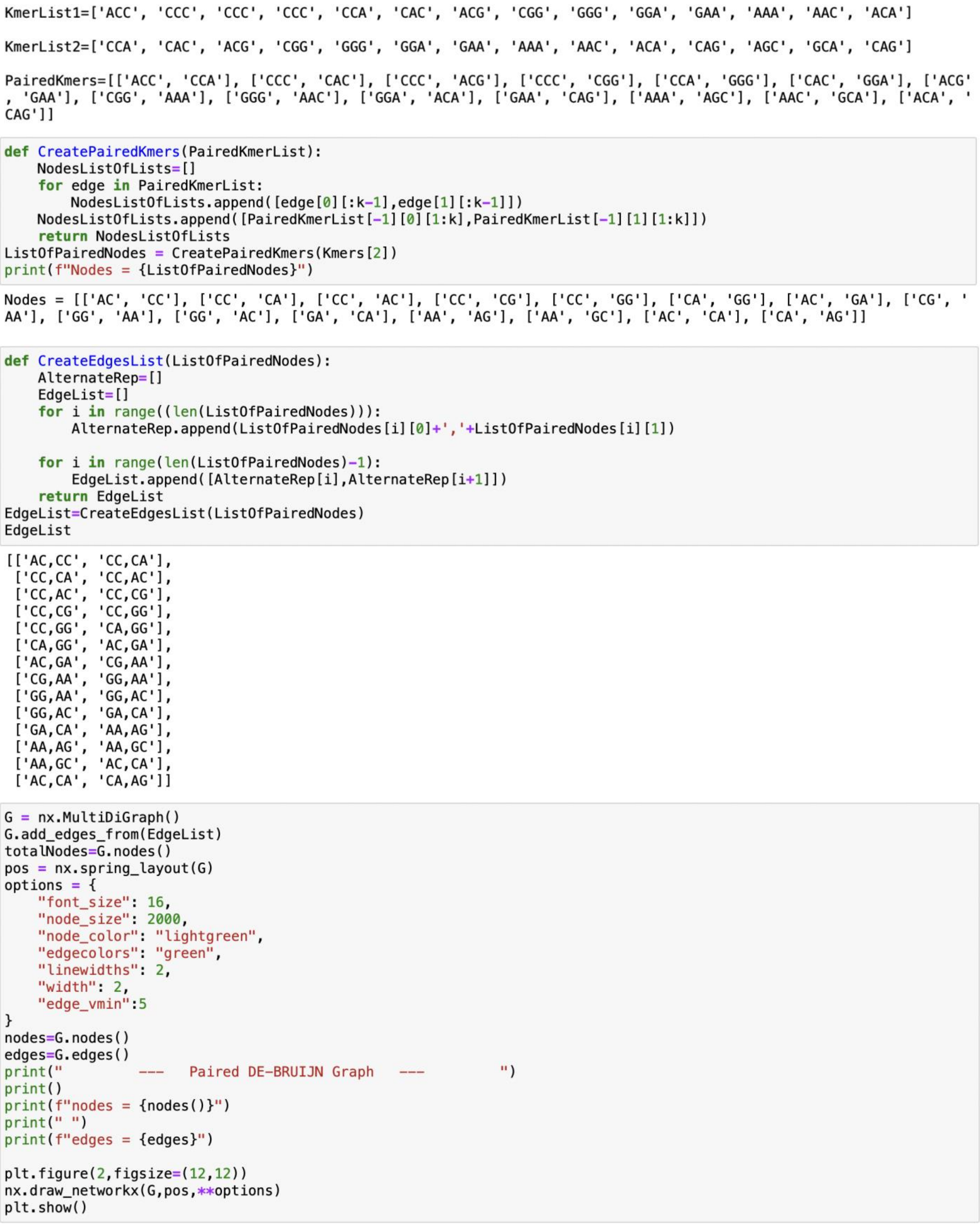
1. 

```
ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCC
AGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAA
GAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTTGACA
CAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGA
TACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCAACAAAACTGCTCGCCAGAACACTACGAGC
CACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG
```

K=3



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


start = timeit.default_timer()


def kmers(read, k,d):
    KList=[]
    KList2=[]
    PairedKmers=[]
    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)

    num_kmers1 = len(read) - k + 1
    for i in range(k+d,num_kmers1):
        kmer = read[i:i+k]
        if i==0:
            KList2.append(kmer)
        else:
            KList2.append(kmer)

    KList1=KList[0:len(KList2)]

    for i in range(len(KList2)):
        PairedKmers.append([KList1[i],KList2[i]])

    return KList1,KList2,PairedKmers

def sortkmer(kmerlist):
    tempkmerlist=kmerlist
    for i in range(len(tempkmerlist) - 1):
        for j in range(i + 1, len(tempkmerlist)):
            if tempkmerlist[i][0] > tempkmerlist[j][0]:
                temp = tempkmerlist[i]
                tempkmerlist[i] = tempkmerlist[j]
                tempkmerlist[j] = temp
    return tempkmerlist

input1="ACCCCCACGGGAAACAGCAG"
k=3
d=1
Kmers=kmers(input1,k,d)

print(f"KmerList1={Kmers[0]}")
print(" ")
print(f"KmerList2={Kmers[1]}")
print()
print(f"PairedKmers={Kmers[2]}")
print()
# lexicographic = sortkmer(Kmers[2])
# print(f'lexiographic = {lexicographic}')
```



```
KmerList1=['ACC', 'CCC', 'CCC', 'CCC', 'CCA', 'CAC', 'ACG', 'CGG', 'GGG', 'GGA', 'GAA', 'AAA', 'AAC', 'ACA']

KmerList2=['CCA', 'CAC', 'ACG', 'CGG', 'GGG', 'GGA', 'GAA', 'AAA', 'AAC', 'ACA', 'CAG', 'AGC', 'GCA', 'CAG']

PairedKmers=[['ACC', 'CCA'], ['CCC', 'CAC'], ['CCC', 'ACG'], ['CCC', 'CGG'], ['CCA', 'GGG'], ['CAC', 'GGA'], ['ACG'
, 'GAA'], ['CGG', 'AAA'], ['GGG', 'AAC'], ['GGA', 'ACA'], ['GAA', 'CAG'], ['AAA', 'AGC'], ['AAC', 'GCA'], ['ACA', '
CAG']]
```

```
def CreatePairedKmers(PairedKmerList):
    NodesListOfLists=[]
    for edge in PairedKmerList:
        NodesListOfLists.append([edge[0][:k-1],edge[1][:k-1]])
    NodesListOfLists.append([PairedKmerList[-1][0][1:k],PairedKmerList[-1][1][1:k]])
    return NodesListOfLists
ListOfPairedNodes = CreatePairedKmers(Kmers[2])
print(f"Nodes = {ListOfPairedNodes}")
```

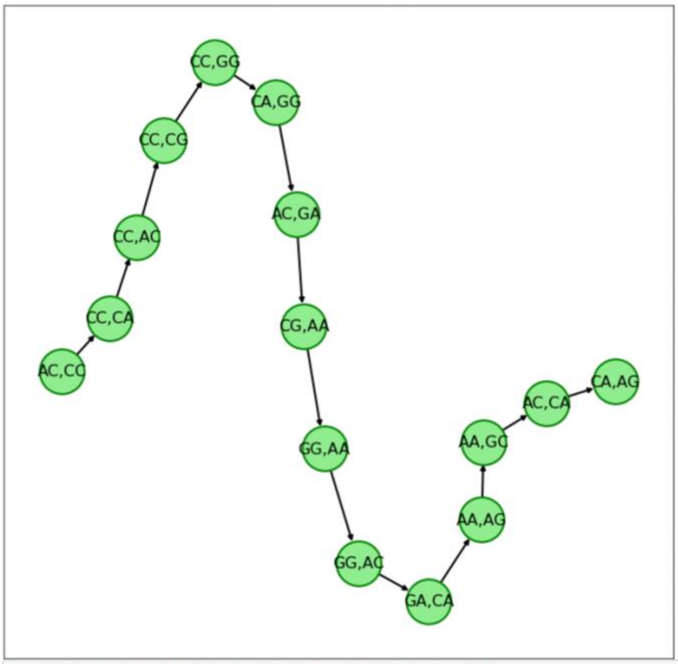
```
Nodes = [['AC', 'CC'], ['CC', 'CA'], ['CC', 'AC'], ['CC', 'CG'], ['CC', 'GG'], ['CA', 'GG'], ['AC', 'GA'], ['CG', '
AA'], ['GG', 'AA'], ['GG', 'AC'], ['GA', 'CA'], ['AA', 'AG'], ['AA', 'GC'], ['AC', 'CA'], ['CA', 'AG']]
```

```
def CreateEdgesList(ListOfPairedNodes):
    AlternateRep=[]
    EdgeList=[]
    for i in range((len(ListOfPairedNodes))):
        AlternateRep.append(ListOfPairedNodes[i][0]+','+ListOfPairedNodes[i][1])

    for i in range(len(ListOfPairedNodes)-1):
        EdgeList.append([AlternateRep[i],AlternateRep[i+1]])
    return EdgeList
EdgeList=CreateEdgesList(ListOfPairedNodes)
EdgeList
```

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

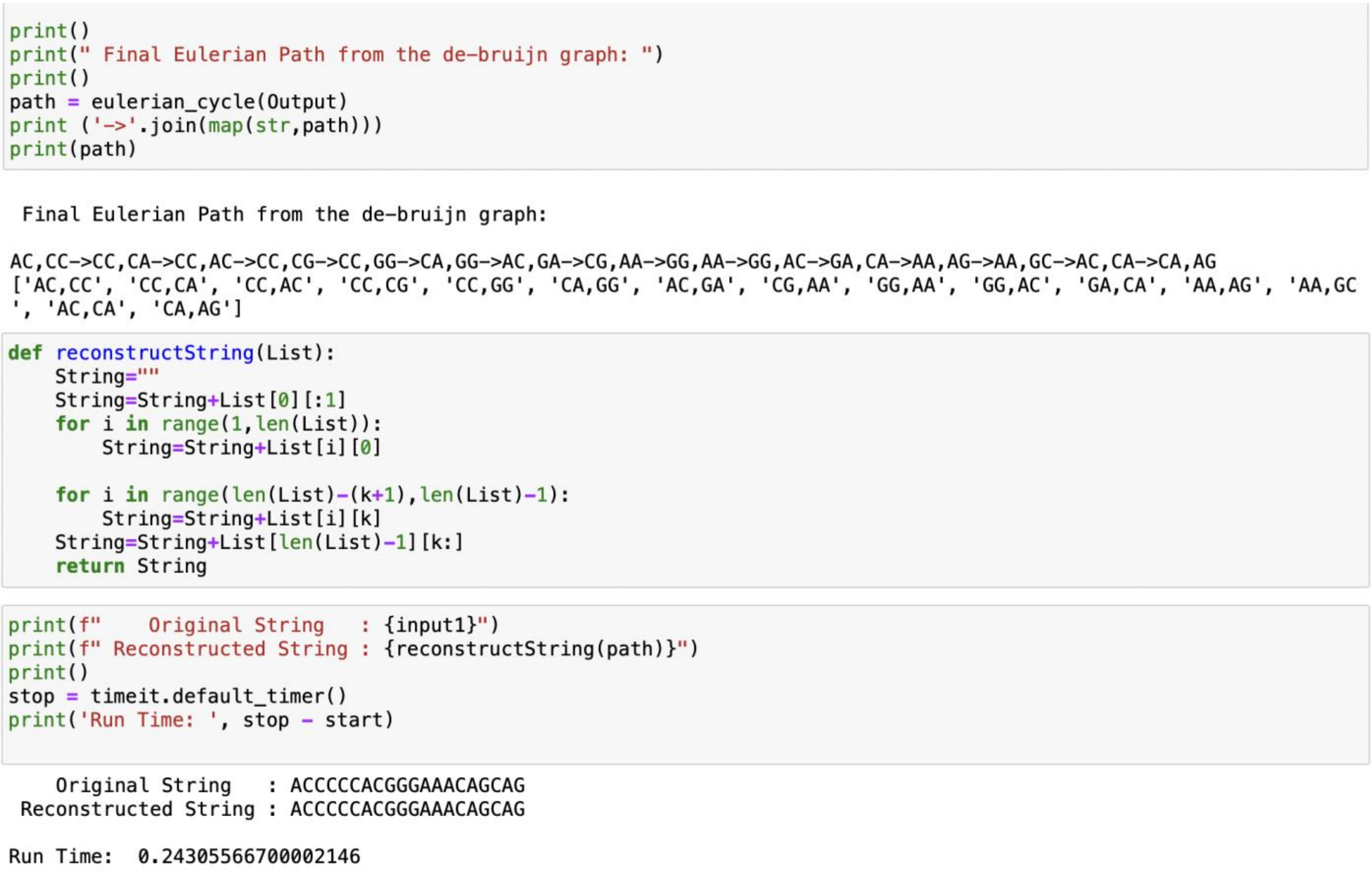
```
G = nx.MultiDiGraph()
G.add_edges_from(EdgeList)
totalNodes=G.nodes()
pos = nx.spring_layout(G)
options = {
    "font_size": 16,
    "node_size": 2000,
    "node_color": "lightgreen",
    "edgecolors": "green",
    "linewidths": 2,
    "width": 2,
    "edge_vmin":5
}
nodes=G.nodes()
edges=G.edges()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

plt.figure(2,figsize=(12,12))
nx.draw_networkx(G,pos,**options)
plt.show()
```

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

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

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



```python
# 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
```



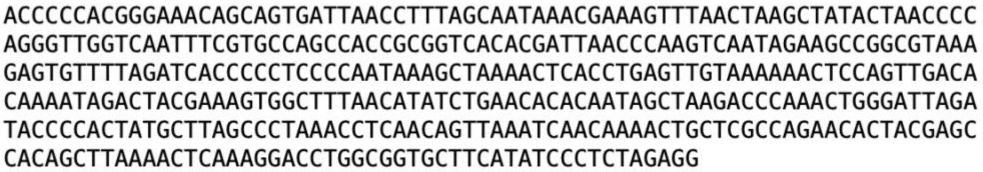
```
print()
print(" Final Eulerian Path from the de-bruijn graph: ")
print()
path = eulerian_cycle(Output)
print ('->'.join(map(str,path)))
print(path)
```

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

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

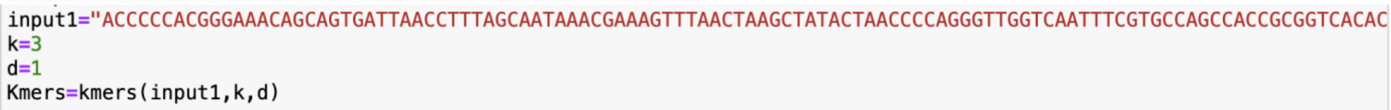
```
def reconstructString(List):
    String=""
    String=String+List[0][:1]
    for i in range(1,len(List)):
        String=String+List[i][0]

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

```
print(f"    Original String    : {input1}")
print(f" Reconstructed String : {reconstructString(path)}")
print()
stop = timeit.default_timer()
print('Run Time: ', stop - start)
```

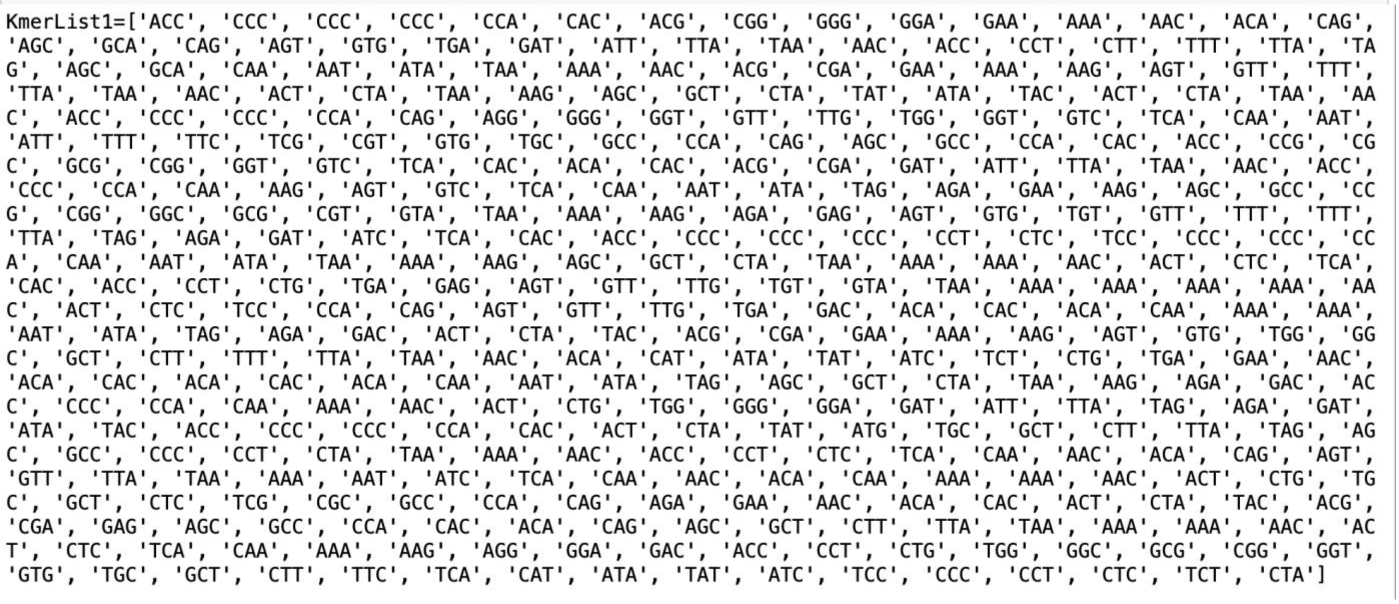
```
    Original String   : ACCCCCACGGGAAACAGCAG
 Reconstructed String : ACCCCCACGGGAAACAGCAG

Run Time:  0.24305566700002146
```



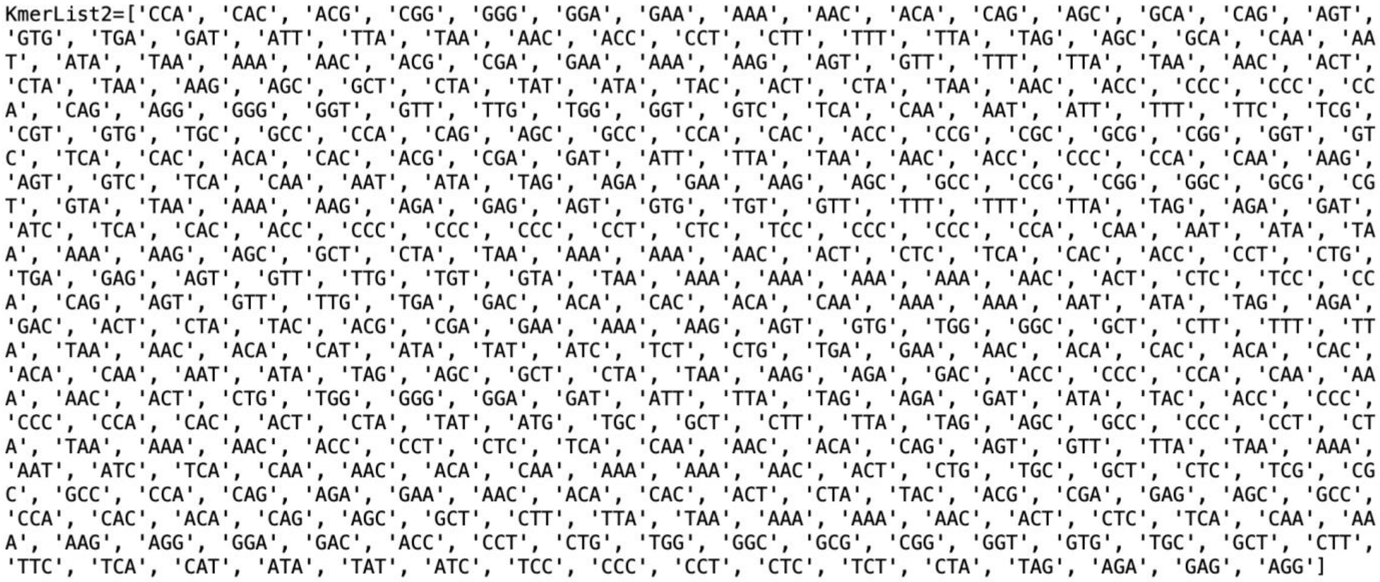
```
ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCC
AGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAA
GAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTTGACA
CAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGA
TACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCAACAAAACTGCTCGCCAGAACACTACGAGC
CACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG
```



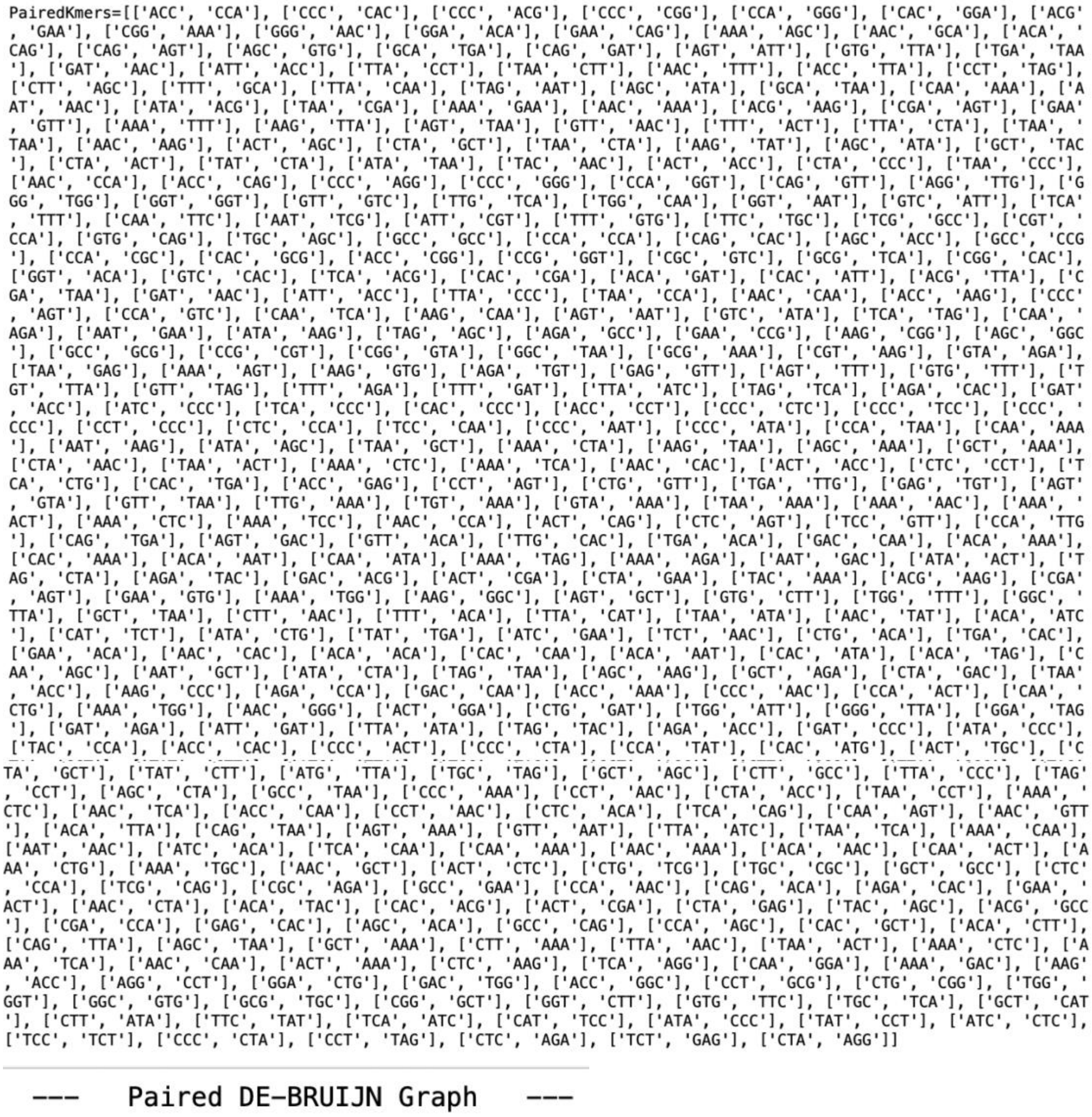
```
input1="ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACAC
k=3
d=1
Kmers=kmers(input1,k,d)
```



```
KmerList1=['ACC', 'CCC', 'CCC', 'CCC', 'CCA', 'CAC', 'ACG', 'CGG', 'GGG', 'GGA', 'GAA', 'AAA', 'AAC', 'ACA', 'CAG',
'AGC', 'GCA', 'CAG', 'AGT', 'GTG', 'TGA', 'GAT', 'ATT', 'TTA', 'TAA', 'AAC', 'ACC', 'CCT', 'CTT', 'TTT', 'TTA', 'TA
G', 'AGC', 'GCA', 'CAA', 'AAT', 'ATA', 'TAA', 'AAA', 'AAC', 'ACG', 'CGA', 'GAA', 'AAA', 'AAG', 'AGT', 'GTT', 'TTT',
'TTA', 'TAA', 'AAC', 'ACT', 'CTA', 'TAA', 'AAG', 'AGC', 'GCT', 'CTA', 'TAT', 'ATA', 'TAC', 'ACT', 'CTA', 'TAA', 'AA
C', 'ACC', 'CCC', 'CCC', 'CCA', 'CAG', 'AGG', 'GGG', 'GGT', 'GTT', 'TTG', 'TGG', 'GGT', 'GTC', 'TCA', 'CAA', 'AAT',
'ATT', 'TTT', 'TTC', 'TCG', 'CGT', 'GTG', 'TGC', 'GCC', 'CCA', 'CAG', 'AGC', 'GCC', 'CCA', 'CAC', 'ACC', 'CCG', 'CG
C', 'GCG', 'CGG', 'GGT', 'GTC', 'TCA', 'CAC', 'ACA', 'CAC', 'ACG', 'CGA', 'GAT', 'ATT', 'TTA', 'TAA', 'AAC', 'ACC',
'CCC', 'CCA', 'CAA', 'AAG', 'AGT', 'GTC', 'TCA', 'CAA', 'AAT', 'ATA', 'TAG', 'AGA', 'GAA', 'AAG', 'AGC', 'GCC', 'CC
G', 'CGG', 'GGC', 'GCG', 'CGT', 'GTA', 'TAA', 'AAA', 'AAG', 'AGA', 'GAG', 'AGT', 'GTG', 'TGT', 'GTT', 'TTT', 'TTT',
'TTA', 'TAG', 'AGA', 'GAT', 'ATC', 'TCA', 'CAC', 'ACC', 'CCC', 'CCC', 'CCC', 'CCT', 'CTC', 'TCC', 'CCC', 'CCC', 'CC
A', 'CAA', 'AAT', 'ATA', 'TAA', 'AAA', 'AAG', 'AGC', 'GCT', 'CTA', 'TAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTC', 'TCA',
'CAC', 'ACC', 'CCT', 'CTG', 'TGA', 'GAG', 'AGT', 'GTT', 'TTG', 'TGT', 'GTA', 'TAA', 'AAA', 'AAA', 'AAA', 'AAA', 'AA
C', 'ACT', 'CTC', 'TCC', 'CCA', 'CAG', 'AGT', 'GTT', 'TTG', 'TGA', 'GAC', 'ACA', 'CAC', 'ACA', 'CAA', 'AAA', 'AAA',
'AAT', 'ATA', 'TAG', 'AGA', 'GAC', 'ACT', 'CTA', 'TAC', 'ACG', 'CGA', 'GAA', 'AAA', 'AAG', 'AGT', 'GTG', 'TGG', 'GG
C', 'GCT', 'CTT', 'TTT', 'TTA', 'TAA', 'AAC', 'ACA', 'CAT', 'ATA', 'TAT', 'ATC', 'TCT', 'CTG', 'TGA', 'GAA', 'AAC',
'ACA', 'CAC', 'ACA', 'CAC', 'ACA', 'CAA', 'AAT', 'ATA', 'TAG', 'AGC', 'GCT', 'CTA', 'TAA', 'AAG', 'AGA', 'GAC', 'AC
C', 'CCC', 'CCA', 'CAA', 'AAA', 'AAC', 'ACT', 'CTG', 'TGG', 'GGG', 'GGA', 'GAT', 'ATT', 'TTA', 'TAG', 'AGA', 'GAT',
'ATA', 'TAC', 'ACC', 'CCC', 'CCC', 'CCA', 'CAC', 'ACT', 'CTA', 'TAT', 'ATG', 'TGC', 'GCT', 'CTT', 'TTA', 'TAG', 'AG
C', 'GCC', 'CCC', 'CCT', 'CTA', 'TAA', 'AAA', 'AAC', 'ACC', 'CCT', 'CTC', 'TCA', 'CAA', 'AAC', 'ACA', 'CAG', 'AGT',
'GTT', 'TTA', 'TAA', 'AAA', 'AAT', 'ATC', 'TCA', 'CAA', 'AAC', 'ACA', 'CAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTG', 'TG
C', 'GCT', 'CTC', 'TCG', 'CGC', 'GCC', 'CCA', 'CAG', 'AGA', 'GAA', 'AAC', 'ACA', 'CAC', 'ACT', 'CTA', 'TAC', 'ACG',
'CGA', 'GAG', 'AGC', 'GCC', 'CCA', 'CAC', 'ACA', 'CAG', 'AGC', 'GCT', 'CTT', 'TTA', 'TAA', 'AAA', 'AAA', 'AAC', 'AC
T', 'CTC', 'TCA', 'CAA', 'AAA', 'AAG', 'AGG', 'GGA', 'GAC', 'ACC', 'CCT', 'CTG', 'TGG', 'GGC', 'GCG', 'CGG', 'GGT',
'GTG', 'TGC', 'GCT', 'CTT', 'TTC', 'TCA', 'CAT', 'ATA', 'TAT', 'ATC', 'TCC', 'CCC', 'CCT', 'CTC', 'TCT', 'CTA']
```

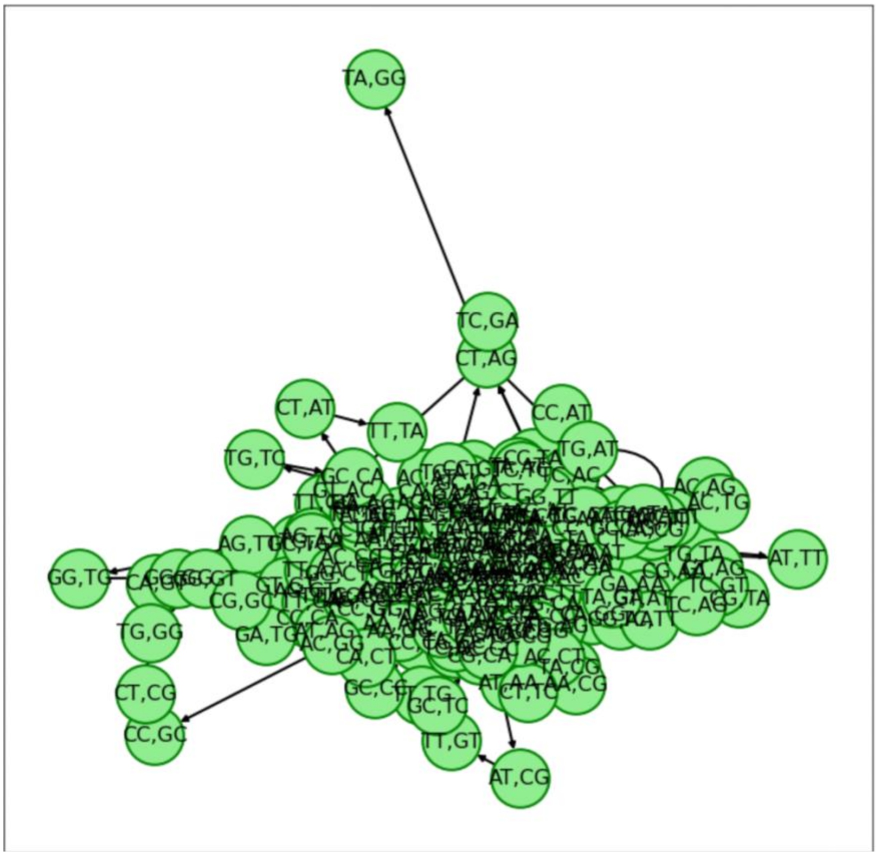


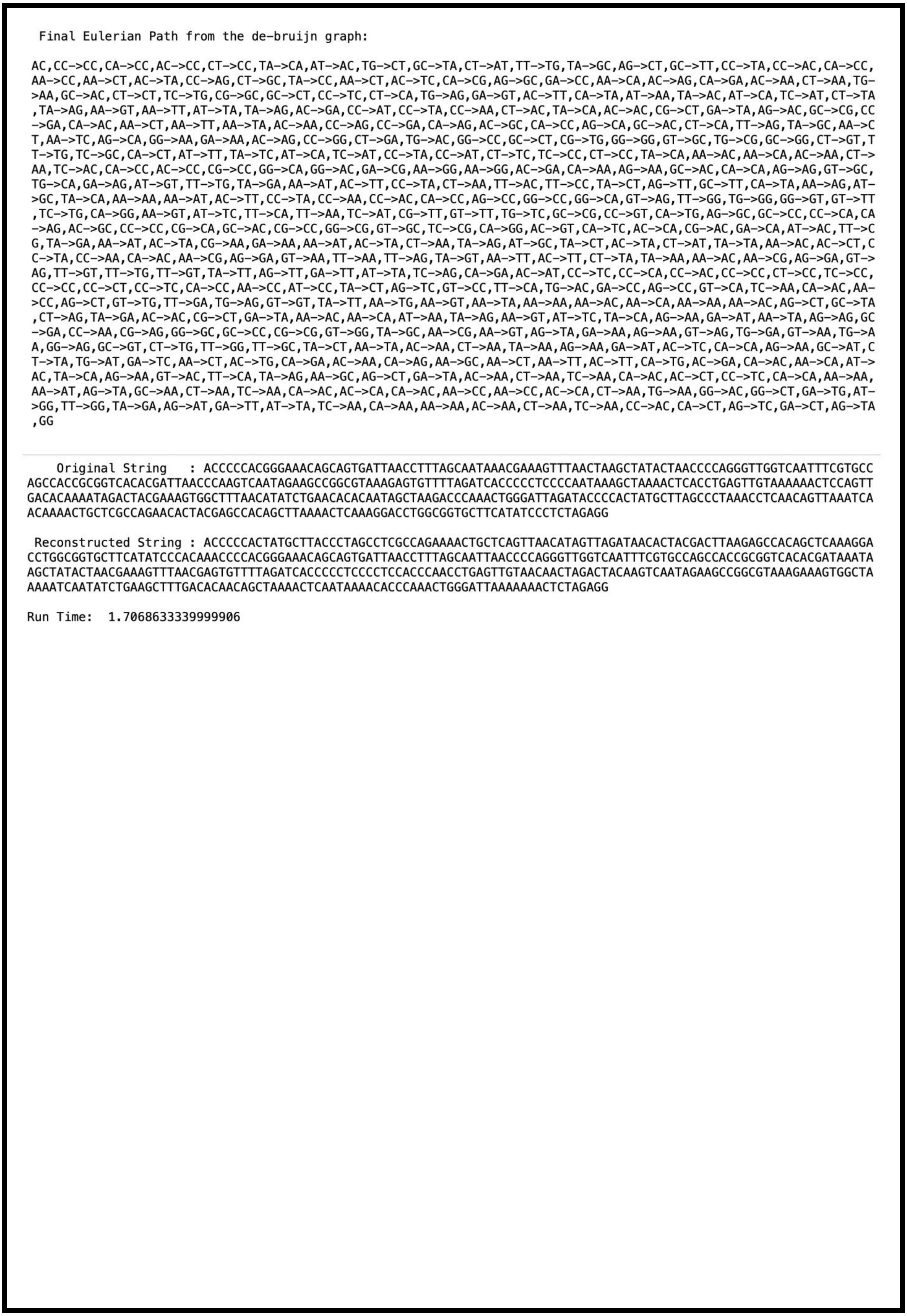
```
KmerList2=['CCA', 'CAC', 'ACG', 'CGG', 'GGG', 'GGA', 'GAA', 'AAA', 'AAC', 'ACA', 'CAG', 'AGC', 'GCA', 'CAG', 'AGT',
'GTG', 'TGA', 'GAT', 'ATT', 'TTA', 'TAA', 'AAC', 'ACC', 'CCT', 'CTT', 'TTT', 'TTA', 'TAG', 'AGC', 'GCA', 'CAA', 'AA
T', 'ATA', 'TAA', 'AAA', 'AAC', 'ACG', 'CGA', 'GAA', 'AAA', 'AAG', 'AGT', 'GTT', 'TTT', 'TTA', 'TAA', 'AAC', 'ACT',
'CTA', 'TAA', 'AAG', 'AGC', 'GCT', 'CTA', 'TAT', 'ATA', 'TAC', 'ACT', 'CTA', 'TAA', 'AAC', 'ACC', 'CCC', 'CCC', 'CC
A', 'CAG', 'AGG', 'GGG', 'GGT', 'GTT', 'TTG', 'TGG', 'GGT', 'GTC', 'TCA', 'CAA', 'AAT', 'ATT', 'TTT', 'TTC', 'TCG',
'CGT', 'GTG', 'TGC', 'GCC', 'CCA', 'CAG', 'AGC', 'GCC', 'CCA', 'CAC', 'ACC', 'CCG', 'CGC', 'GCG', 'CGG', 'GGT', 'GT
C', 'TCA', 'CAC', 'ACA', 'CAC', 'ACG', 'CGA', 'GAT', 'ATT', 'TTA', 'TAA', 'AAC', 'ACC', 'CCC', 'CCA', 'CAA', 'AAG',
'AGT', 'GTC', 'TCA', 'CAA', 'AAT', 'ATA', 'TAG', 'AGA', 'GAA', 'AAG', 'AGC', 'GCC', 'CCG', 'CGG', 'GGC', 'GCG', 'CG
T', 'GTA', 'TAA', 'AAA', 'AAG', 'AGA', 'GAG', 'AGT', 'GTG', 'TGT', 'GTT', 'TTT', 'TTT', 'TTA', 'TAG', 'AGA', 'GAT',
'ATC', 'TCA', 'CAC', 'ACC', 'CCC', 'CCC', 'CCC', 'CCT', 'CTC', 'TCC', 'CCC', 'CCC', 'CCA', 'CAA', 'AAT', 'ATA', 'TA
A', 'AAA', 'AAG', 'AGC', 'GCT', 'CTA', 'TAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTC', 'TCA', 'CAC', 'ACC', 'CCT', 'CTG',
'TGA', 'GAG', 'AGT', 'GTT', 'TTG', 'TGT', 'GTA', 'TAA', 'AAA', 'AAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTC', 'TCC', 'CC
A', 'CAG', 'AGT', 'GTT', 'TTG', 'TGA', 'GAC', 'ACA', 'CAC', 'ACA', 'CAA', 'AAA', 'AAA', 'AAT', 'ATA', 'TAG', 'AGA',
'GAC', 'ACT', 'CTA', 'TAC', 'ACG', 'CGA', 'GAA', 'AAA', 'AAG', 'AGT', 'GTG', 'TGG', 'GGC', 'GCT', 'CTT', 'TTT', 'TT
A', 'TAA', 'AAC', 'ACA', 'CAT', 'ATA', 'TAT', 'ATC', 'TCT', 'CTG', 'TGA', 'GAA', 'AAC', 'ACA', 'CAC', 'ACA', 'CAC',
'ACA', 'CAA', 'AAT', 'ATA', 'TAG', 'AGC', 'GCT', 'CTA', 'TAA', 'AAG', 'AGA', 'GAC', 'ACC', 'CCC', 'CCA', 'CAA', 'AA
A', 'AAC', 'ACT', 'CTG', 'TGG', 'GGG', 'GGA', 'GAT', 'ATT', 'TTA', 'TAG', 'AGA', 'GAT', 'ATA', 'TAC', 'ACC', 'CCC',
'CCC', 'CCA', 'CAC', 'ACT', 'CTA', 'TAT', 'ATG', 'TGC', 'GCT', 'CTT', 'TTA', 'TAG', 'AGC', 'GCC', 'CCC', 'CCT', 'CT
A', 'TAA', 'AAA', 'AAC', 'ACC', 'CCT', 'CTC', 'TCA', 'CAA', 'AAC', 'ACA', 'CAG', 'AGT', 'GTT', 'TTA', 'TAA', 'AAA',
'AAT', 'ATC', 'TCA', 'CAA', 'AAC', 'ACA', 'CAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTG', 'TGC', 'GCT', 'CTC', 'TCG', 'CG
C', 'GCC', 'CCA', 'CAG', 'AGA', 'GAA', 'AAC', 'ACA', 'CAC', 'ACT', 'CTA', 'TAC', 'ACG', 'CGA', 'GAG', 'AGC', 'GCC',
'CCA', 'CAC', 'ACA', 'CAG', 'AGC', 'GCT', 'CTT', 'TTA', 'TAA', 'AAA', 'AAA', 'AAC', 'ACT', 'CTC', 'TCA', 'CAA', 'AA
A', 'AAG', 'AGG', 'GGA', 'GAC', 'ACC', 'CCT', 'CTG', 'TGG', 'GGC', 'GCG', 'CGG', 'GGT', 'GTG', 'TGC', 'GCT', 'CTT',
'TTC', 'TCA', 'CAT', 'ATA', 'TAT', 'ATC', 'TCC', 'CCC', 'CCT', 'CTC', 'TCT', 'CTA', 'TAG', 'AGA', 'GAG', 'AGG']
```



```
PairedKmers=[['ACC', 'CCA'], ['CCC', 'CAC'], ['CCC', 'ACG'], ['CCC', 'CGG'], ['CCA', 'GGG'], ['CAC', 'GGA'], ['ACG'
, 'GAA'], ['CGG', 'AAA'], ['GGG', 'AAC'], ['GGA', 'ACA'], ['GAA', 'CAG'], ['AAA', 'AGC'], ['AAC', 'GCA'], ['ACA', '
CAG'], ['CAG', 'AGT'], ['AGC', 'GTG'], ['GCA', 'TGA'], ['CAG', 'GAT'], ['AGT', 'ATT'], ['GTG', 'TTA'], ['TGA', 'TAA
'], ['GAT', 'AAC'], ['ATT', 'ACC'], ['TTA', 'CCT'], ['TAA', 'CTT'], ['AAC', 'TTT'], ['ACC', 'TTA'], ['CCT', 'TAG'],
['CTT', 'AGC'], ['TTT', 'GCA'], ['TTA', 'CAA'], ['TAG', 'AAT'], ['AGC', 'ATA'], ['GCA', 'TAA'], ['CAA', 'AAA'], ['A
AT', 'AAC'], ['ATA', 'ACG'], ['TAA', 'CGA'], ['AAA', 'GAA'], ['AAC', 'AAA'], ['ACG', 'AAG'], ['CGA', 'AGT'], ['GAA'
, 'GTT'], ['AAA', 'TTT'], ['AAG', 'TTA'], ['AGT', 'TAA'], ['GTT', 'AAC'], ['TTT', 'ACT'], ['TTA', 'CTA'], ['TAA', '
TAA'], ['AAC', 'AAG'], ['ACT', 'AGC'], ['CTA', 'GCT'], ['TAA', 'CTA'], ['AAG', 'TAT'], ['AGC', 'ATA'], ['GCT', 'TAC
'], ['CTA', 'ACT'], ['TAT', 'CTA'], ['ATA', 'TAA'], ['TAC', 'AAC'], ['ACT', 'ACC'], ['CTA', 'CCC'], ['TAA', 'CCC'],
['AAC', 'CCA'], ['ACC', 'CAG'], ['CCC', 'AGG'], ['CCC', 'GGG'], ['CCA', 'GGT'], ['CAG', 'GTT'], ['AGG', 'TTG'], ['G
GG', 'TGG'], ['GGT', 'GGT'], ['GTT', 'GTC'], ['TTG', 'TCA'], ['TGG', 'CAA'], ['GGT', 'AAT'], ['GTC', 'ATT'], ['TCA'
, 'TTT'], ['CAA', 'TTC'], ['AAT', 'TCG'], ['ATT', 'CGT'], ['TTT', 'GTG'], ['TTC', 'TGC'], ['TCG', 'GCC'], ['CGT', '
CCA'], ['GTG', 'CAG'], ['TGC', 'AGC'], ['GCC', 'GCC'], ['CCA', 'CCA'], ['CAG', 'CAC'], ['AGC', 'ACC'], ['GCC', 'CCG
'], ['CCA', 'CGC'], ['CAC', 'GCG'], ['ACC', 'CGG'], ['CCG', 'GGT'], ['CGC', 'GTC'], ['GCG', 'TCA'], ['CGG', 'CAC'],
['GGT', 'ACA'], ['GTC', 'CAC'], ['TCA', 'ACG'], ['CAC', 'CGA'], ['ACA', 'GAT'], ['CAC', 'ATT'], ['ACG', 'TTA'], ['C
GA', 'TAA'], ['GAT', 'AAC'], ['ATT', 'ACC'], ['TTA', 'CCC'], ['TAA', 'CCA'], ['AAC', 'CAA'], ['ACC', 'AAG'], ['CCC'
, 'AGT'], ['CCA', 'GTC'], ['CAA', 'TCA'], ['AAG', 'CAA'], ['AGT', 'AAT'], ['GTC', 'ATA'], ['TCA', 'TAG'], ['CAA', '
AGA'], ['AAT', 'GAA'], ['ATA', 'AAG'], ['TAG', 'AGC'], ['AGA', 'GCC'], ['GAA', 'CCG'], ['AAG', 'CGG'], ['AGC', 'GGC
'], ['GCC', 'GCG'], ['CCG', 'CGT'], ['CGG', 'GTA'], ['GGC', 'TAA'], ['GCG', 'AAA'], ['CGT', 'AAG'], ['GTA', 'AGA'],
['TAA', 'GAG'], ['AAA', 'AGT'], ['AAG', 'GTG'], ['AGA', 'TGT'], ['GAG', 'GTT'], ['AGT', 'TTT'], ['GTG', 'TTT'], ['T
GT', 'TTA'], ['GTT', 'TAG'], ['TTT', 'AGA'], ['TTT', 'GAT'], ['TTA', 'ATC'], ['TAG', 'TCA'], ['AGA', 'CAC'], ['GAT'
, 'ACC'], ['ATC', 'CCC'], ['TCA', 'CCC'], ['CAC', 'CCC'], ['ACC', 'CCT'], ['CCC', 'CTC'], ['CCC', 'TCC'], ['CCC', '
CCC'], ['CCT', 'CCC'], ['CTC', 'CCA'], ['TCC', 'CAA'], ['CCC', 'AAT'], ['CCC', 'ATA'], ['CCA', 'TAA'], ['CAA', 'AAA
'], ['AAT', 'AAG'], ['ATA', 'AGC'], ['TAA', 'GCT'], ['AAA', 'CTA'], ['AAG', 'TAA'], ['AGC', 'AAA'], ['GCT', 'AAA'],
['CTA', 'AAC'], ['TAA', 'ACT'], ['AAA', 'CTC'], ['AAA', 'TCA'], ['AAC', 'CAC'], ['ACT', 'ACC'], ['CTC', 'CCT'], ['T
CA', 'CTG'], ['CAC', 'TGA'], ['ACC', 'GAG'], ['CCT', 'AGT'], ['CTG', 'GTT'], ['TGA', 'TTG'], ['GAG', 'TGT'], ['AGT'
, 'GTA'], ['GTT', 'TAA'], ['TTG', 'AAA'], ['TGT', 'AAA'], ['GTA', 'AAA'], ['TAA', 'AAA'], ['AAA', 'AAC'], ['AAA', '
ACT'], ['AAA', 'CTC'], ['AAA', 'TCC'], ['AAC', 'CCA'], ['ACT', 'CAG'], ['CTC', 'AGT'], ['TCC', 'GTT'], ['CCA', 'TTG
'], ['CAG', 'TGA'], ['AGT', 'GAC'], ['GTT', 'ACA'], ['TTG', 'CAC'], ['TGA', 'ACA'], ['GAC', 'CAA'], ['ACA', 'AAA'],
['CAC', 'AAA'], ['ACA', 'AAT'], ['CAA', 'ATA'], ['AAA', 'TAG'], ['AAA', 'AGA'], ['AAT', 'GAC'], ['ATA', 'ACT'], ['T
AG', 'CTA'], ['AGA', 'TAC'], ['GAC', 'ACG'], ['ACT', 'CGA'], ['CTA', 'GAA'], ['TAC', 'AAA'], ['ACG', 'AAG'], ['CGA'
, 'AGT'], ['GAA', 'GTG'], ['AAA', 'TGG'], ['AAG', 'GGC'], ['AGT', 'GCT'], ['GTG', 'CTT'], ['TGG', 'TTT'], ['GGC', '
TTA'], ['GCT', 'TAA'], ['CTT', 'AAC'], ['TTT', 'ACA'], ['TTA', 'CAT'], ['TAA', 'ATA'], ['AAC', 'TAT'], ['ACA', 'ATC
'], ['CAT', 'TCT'], ['ATA', 'CTG'], ['TAT', 'TGA'], ['ATC', 'GAA'], ['TCT', 'AAC'], ['CTG', 'ACA'], ['TGA', 'CAC'],
['GAA', 'ACA'], ['AAC', 'CAC'], ['ACA', 'ACA'], ['CAC', 'CAA'], ['ACA', 'AAT'], ['CAC', 'ATA'], ['ACA', 'TAG'], ['C
AA', 'AGC'], ['AAT', 'GCT'], ['ATA', 'CTA'], ['TAG', 'TAA'], ['AGC', 'AAG'], ['GCT', 'AGA'], ['CTA', 'GAC'], ['TAA'
, 'ACC'], ['AAG', 'CCC'], ['AGA', 'CCA'], ['GAC', 'CAA'], ['ACC', 'AAA'], ['CCC', 'AAC'], ['CCA', 'ACT'], ['CAA', '
CTG'], ['AAA', 'TGG'], ['AAC', 'GGG'], ['ACT', 'GGA'], ['CTG', 'GAT'], ['TGG', 'ATT'], ['GGG', 'TTA'], ['GGA', 'TAG
'], ['GAT', 'AGA'], ['ATT', 'GAT'], ['TTA', 'ATA'], ['TAG', 'TAC'], ['AGA', 'ACC'], ['GAT', 'CCC'], ['ATA', 'CCC'],
['TAC', 'CCA'], ['ACC', 'CAC'], ['CCC', 'ACT'], ['CCC', 'CTA'], ['CCA', 'TAT'], ['CAC', 'ATG'], ['ACT', 'TGC'], ['C
TA', 'GCT'], ['TAT', 'CTT'], ['ATG', 'TTA'], ['TGC', 'TAG'], ['GCT', 'AGC'], ['CTT', 'GCC'], ['TTA', 'CCC'], ['TAG'
, 'CCT'], ['AGC', 'CTA'], ['GCC', 'TAA'], ['CCC', 'AAA'], ['CCT', 'AAC'], ['CTA', 'ACC'], ['TAA', 'CCT'], ['AAA', '
CTC'], ['AAC', 'TCA'], ['ACC', 'CAA'], ['CCT', 'AAC'], ['CTC', 'ACA'], ['TCA', 'CAG'], ['CAA', 'AGT'], ['AAC', 'GTT
'], ['ACA', 'TTA'], ['CAG', 'TAA'], ['AGT', 'AAA'], ['GTT', 'AAT'], ['TTA', 'ATC'], ['TAA', 'TCA'], ['AAA', 'CAA'],
['AAT', 'AAC'], ['ATC', 'ACA'], ['TCA', 'CAA'], ['CAA', 'AAA'], ['AAC', 'AAA'], ['ACA', 'AAC'], ['CAA', 'ACT'], ['A
AA', 'CTG'], ['AAA', 'TGC'], ['AAC', 'GCT'], ['ACT', 'CTC'], ['CTG', 'TCG'], ['TGC', 'CGC'], ['GCT', 'GCC'], ['CTC'
, 'CCA'], ['TCG', 'CAG'], ['CGC', 'AGA'], ['GCC', 'GAA'], ['CCA', 'AAC'], ['CAG', 'ACA'], ['AGA', 'CAC'], ['GAA', '
ACT'], ['AAC', 'CTA'], ['ACA', 'TAC'], ['CAC', 'ACG'], ['ACT', 'CGA'], ['CTA', 'GAG'], ['TAC', 'AGC'], ['ACG', 'GCC
'], ['CGA', 'CCA'], ['GAG', 'CAC'], ['AGC', 'ACA'], ['GCC', 'CAG'], ['CCA', 'AGC'], ['CAC', 'GCT'], ['ACA', 'CTT'],
['CAG', 'TTA'], ['AGC', 'TAA'], ['GCT', 'AAA'], ['CTT', 'AAA'], ['TTA', 'AAC'], ['TAA', 'ACT'], ['AAA', 'CTC'], ['A
AA', 'TCA'], ['AAC', 'CAA'], ['ACT', 'AAA'], ['CTC', 'AAG'], ['TCA', 'AGG'], ['CAA', 'GGA'], ['AAA', 'GAC'], ['AAG'
, 'ACC'], ['AGG', 'CCT'], ['GGA', 'CTG'], ['GAC', 'TGG'], ['ACC', 'GGC'], ['CCT', 'GCG'], ['CTG', 'CGG'], ['TGG', '
GGT'], ['GGC', 'GTG'], ['GCG', 'TGC'], ['CGG', 'GCT'], ['GGT', 'CTT'], ['GTG', 'TTC'], ['TGC', 'TCA'], ['GCT', 'CAT
'], ['CTT', 'ATA'], ['TTC', 'TAT'], ['TCA', 'ATC'], ['CAT', 'TCC'], ['ATA', 'CCC'], ['TAT', 'CCT'], ['ATC', 'CTC'],
['TCC', 'TCT'], ['CCC', 'CTA'], ['CCT', 'TAG'], ['CTC', 'AGA'], ['TCT', 'GAG'], ['CTA', 'AGG']]
```

--- Paired DE-BRUIJN Graph ---

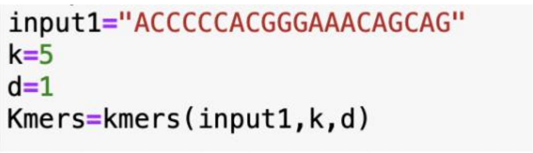




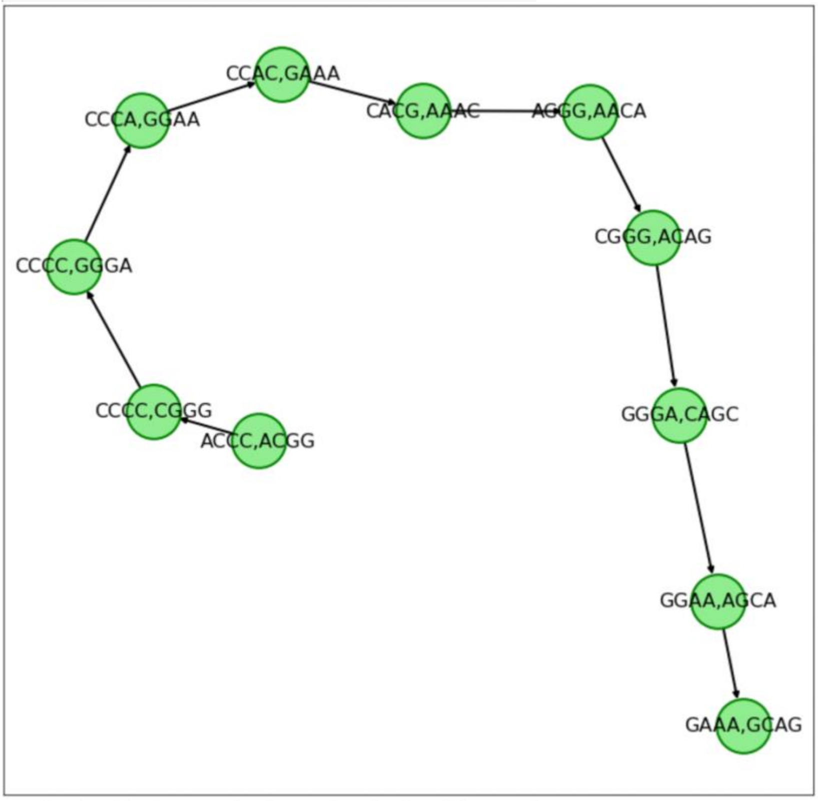
```
 Final Eulerian Path from the de-bruijn graph:

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

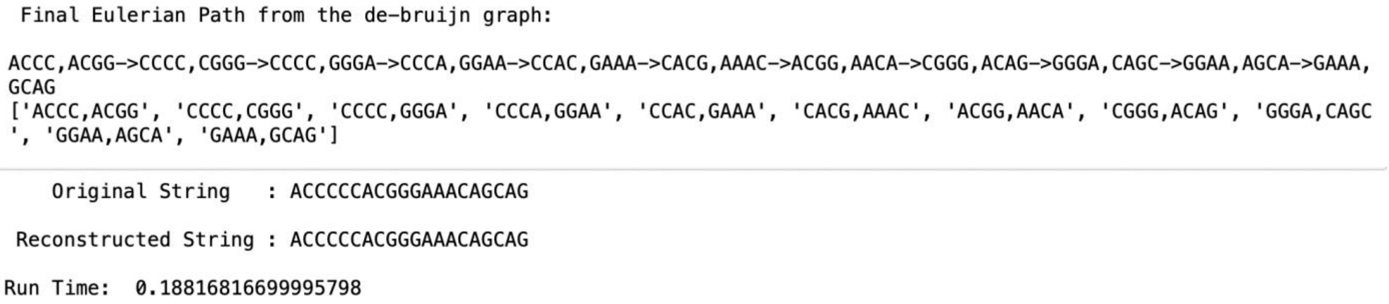
```
    Original String   : ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCC
AGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTT
GACACAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCA
ACAAAACTGCTCGCCAGAACACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

 Reconstructed String : ACCCCCACTATGCTTACCCTAGCCTCGCCAGAAAACTGCTCAGTTAACATAGTTAGATAACACTACGACTTAAGAGCCACAGCTCAAAGGA
CCTGGCGGTGCTTCATATCCCACAAACCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATTAACCCCAGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACACGATAAATA
AGCTATACTAACGAAAGTTTAACGAGTGTTTTAGATCACCCCCTCCCCTCCACCCAACCTGAGTTGTAACAACTAGACTACAAGTCAATAGAAGCCGGCGTAAAGAAAGTGGCTA
AAAATCAATATCTGAAGCTTTGACACAACAGCTAAAACTCAATAAAACACCCAAACTGGGATTAAAAAAACTCTAGAGG

Run Time:  1.7068633339999906
```

1. K=5



```
input1="ACCCCCACGGGAAACAGCAG"
k=5
d=1
Kmers=kmers(input1,k,d)
```

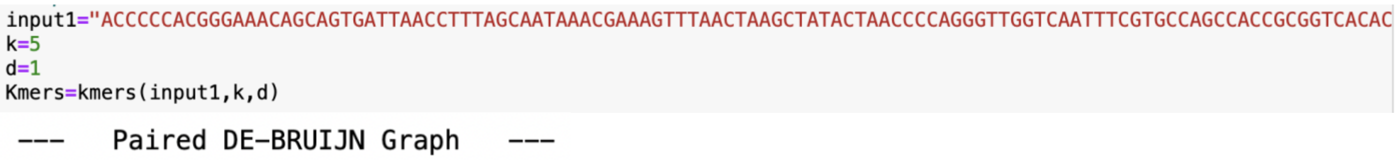




```
 Final Eulerian Path from the de-bruijn graph:

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

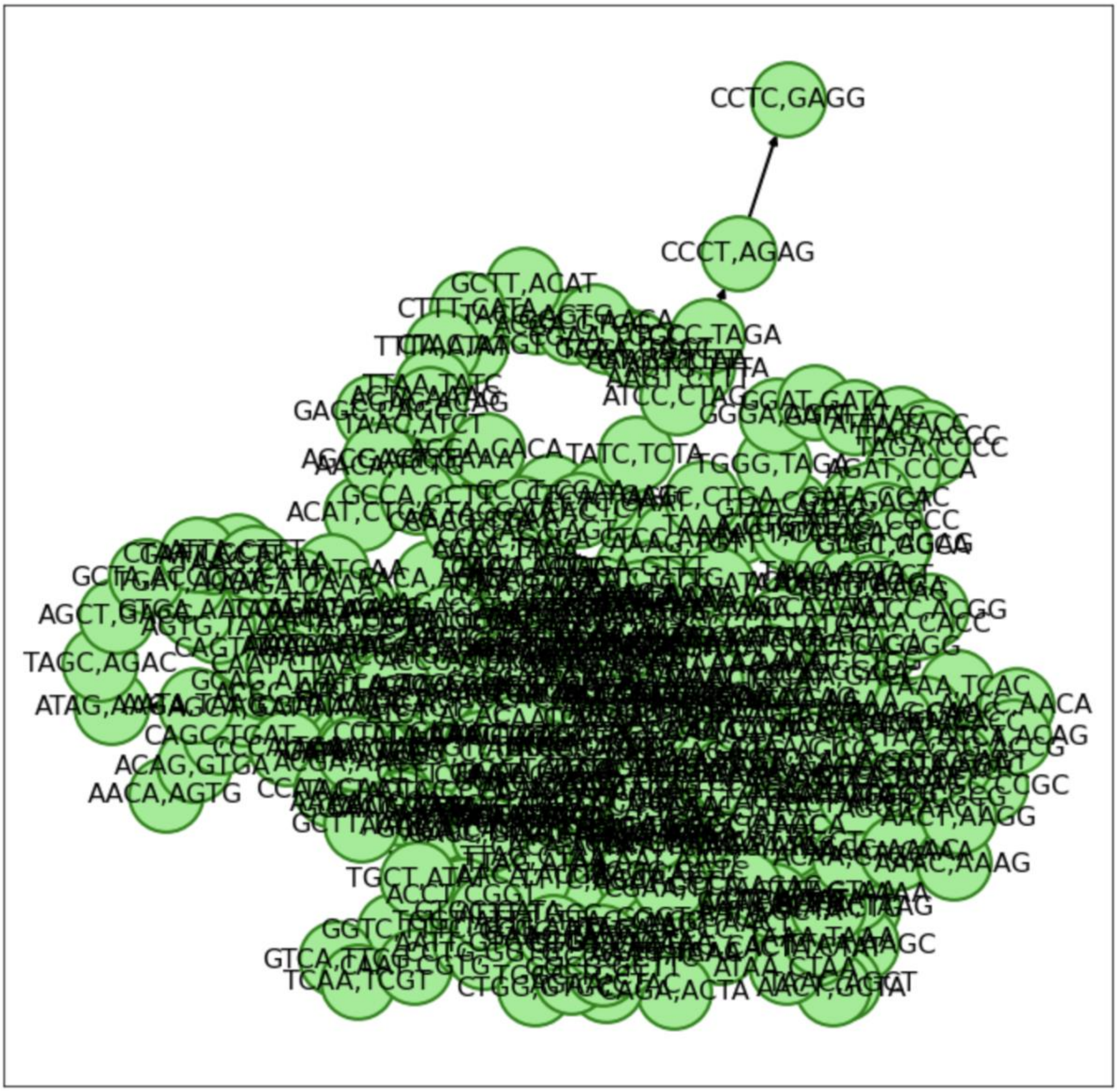
```
    Original String    : ACCCCCACGGGAAACAGCAG

 Reconstructed String : ACCCCCACGGGAAACAGCAG

Run Time:  0.18816816699995798
```

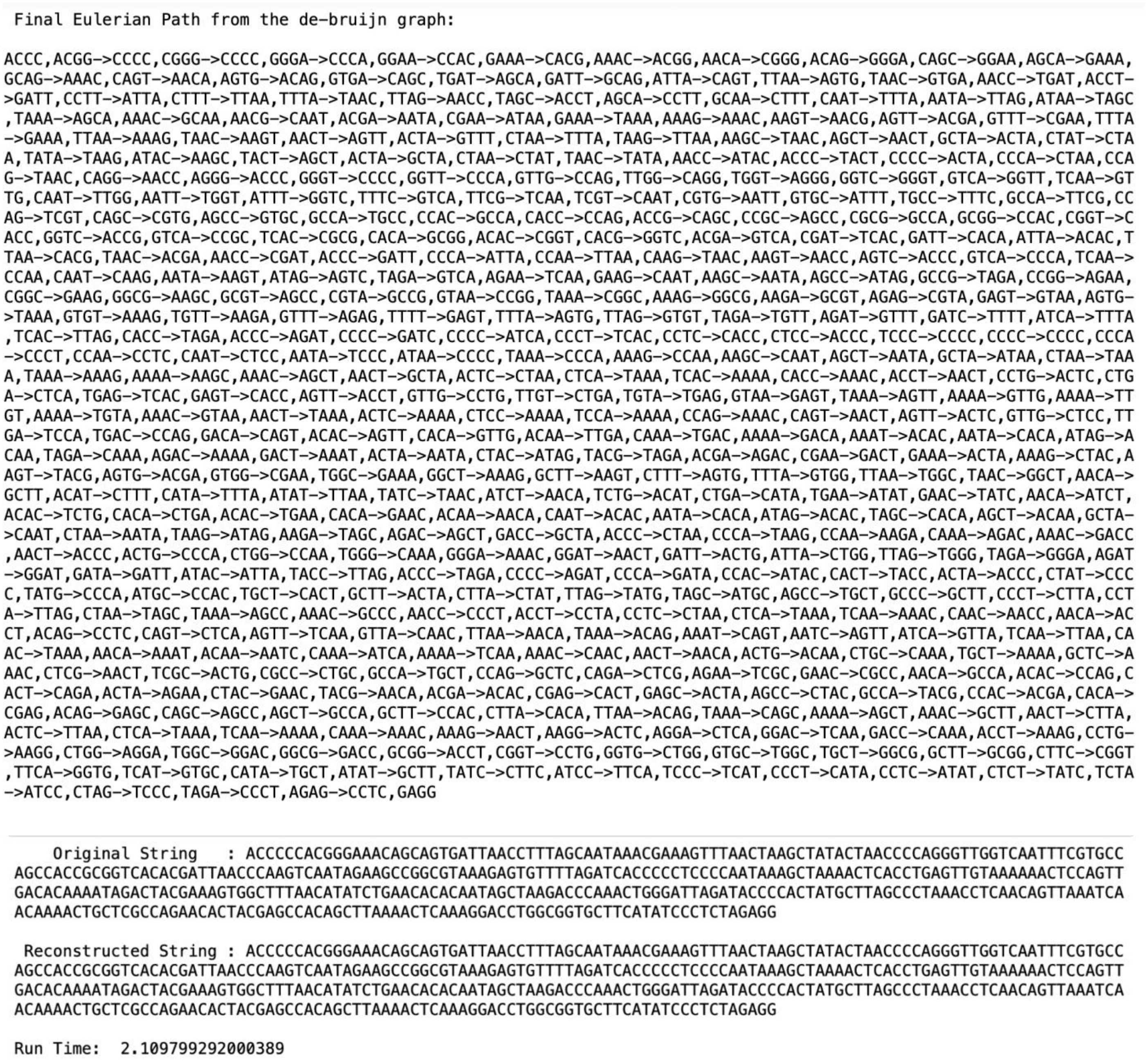
K=5



```
input1="ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCCAGCCACCGCGGTCACAC
k=5
d=1
Kmers=kmers(input1,k,d)
```

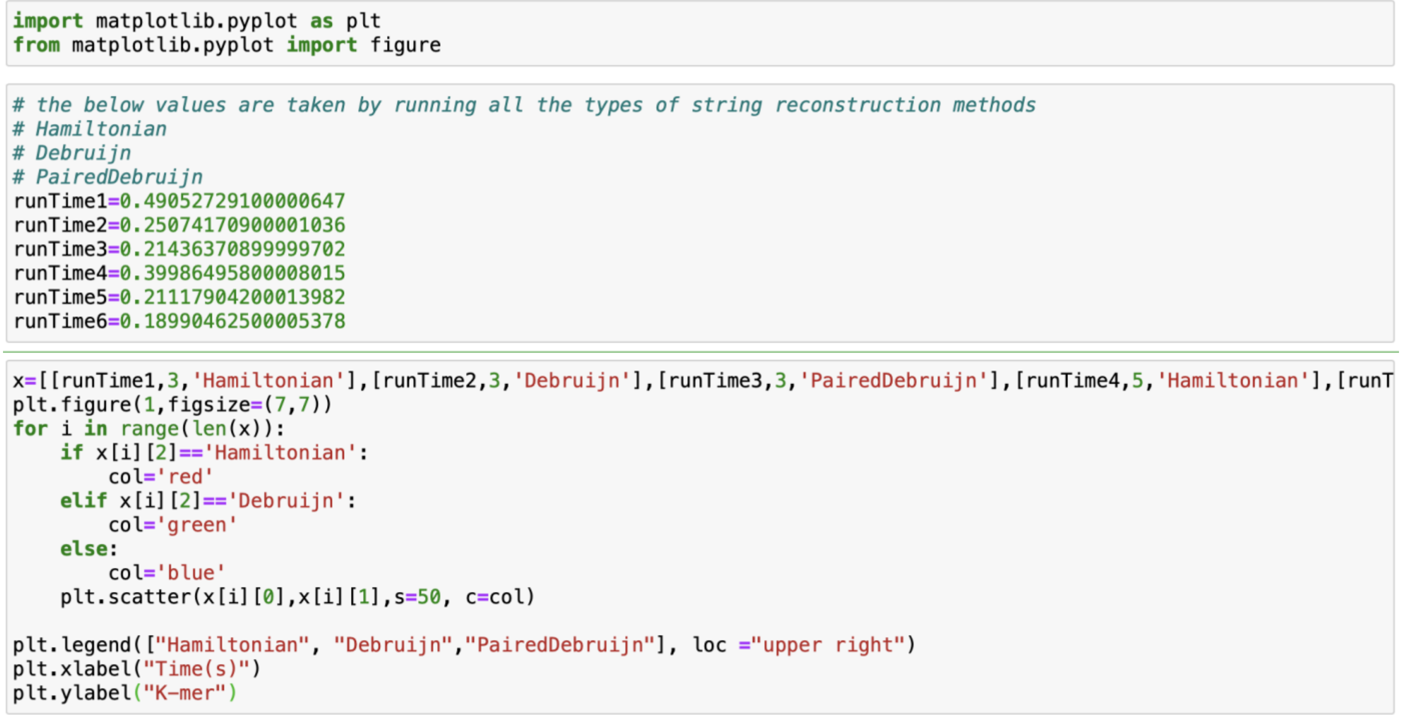
```
 ---   Paired DE-BRUIJN Graph   ---
```





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

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


     Original String   : ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCC
 AGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTT
 GACACAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCA
 ACAAAACTGCTCGCCAGAACACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

  Reconstructed String : ACCCCCACGGGAAACAGCAGTGATTAACCTTTAGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCC
 AGCCACCGCGGTCACACGATTAACCCAAGTCAATAGAAGCCGGCGTAAAGAGTGTTTTAGATCACCCCCTCCCCAATAAAGCTAAAACTCACCTGAGTTGTAAAAAACTCCAGTT
 GACACAAAATAGACTACGAAAGTGGCTTTAACATATCTGAACACACAATAGCTAAGACCCAAACTGGGATTAGATACCCCACTATGCTTAGCCCTAAACCTCAACAGTTAAATCA
 ACAAAACTGCTCGCCAGAACACTACGAGCCACAGCTTAAAACTCAAAGGACCTGGCGGTGCTTCATATCCCTCTAGAGG

 Run Time:  2.109799292000389
```

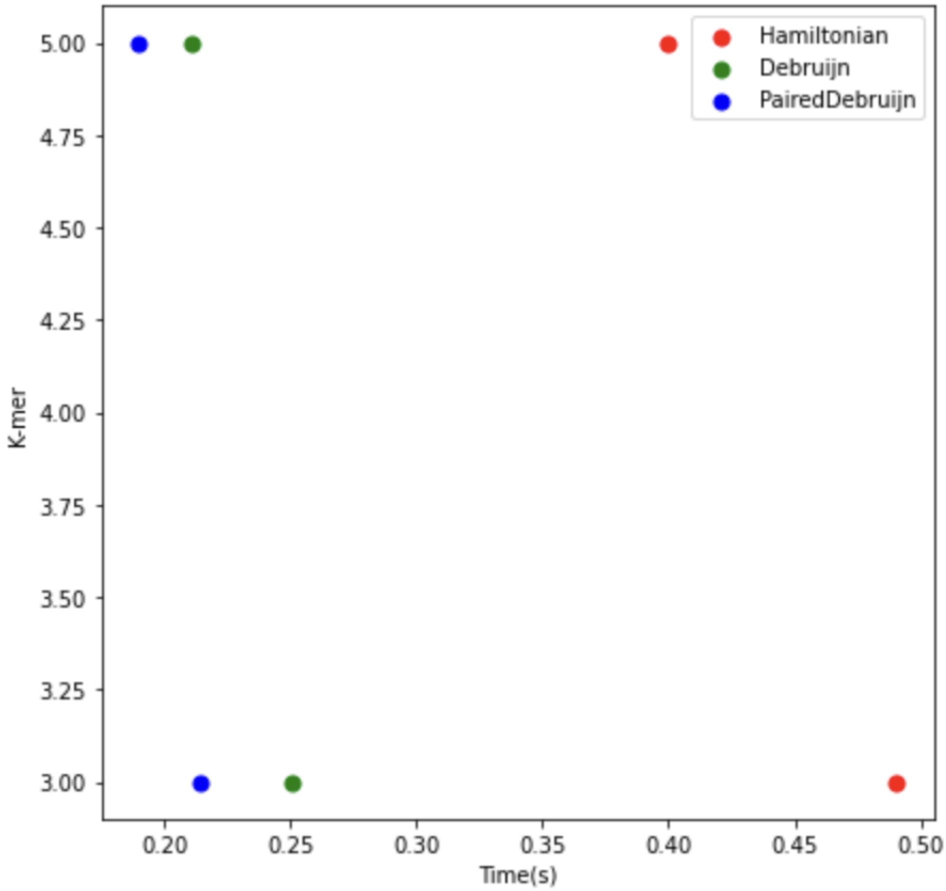
2. Write a program to plot the different run times (for 20bp) for each of the string reconstruction methods such as Hamiltonian, DeBruijn and Paired DeBruijn (for both k=3, & k=5). Tabulate the different runtimes in the report. Also explain in brief justifying which method is better. (Hint: Use **matplotlib** package for plotting)



```
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

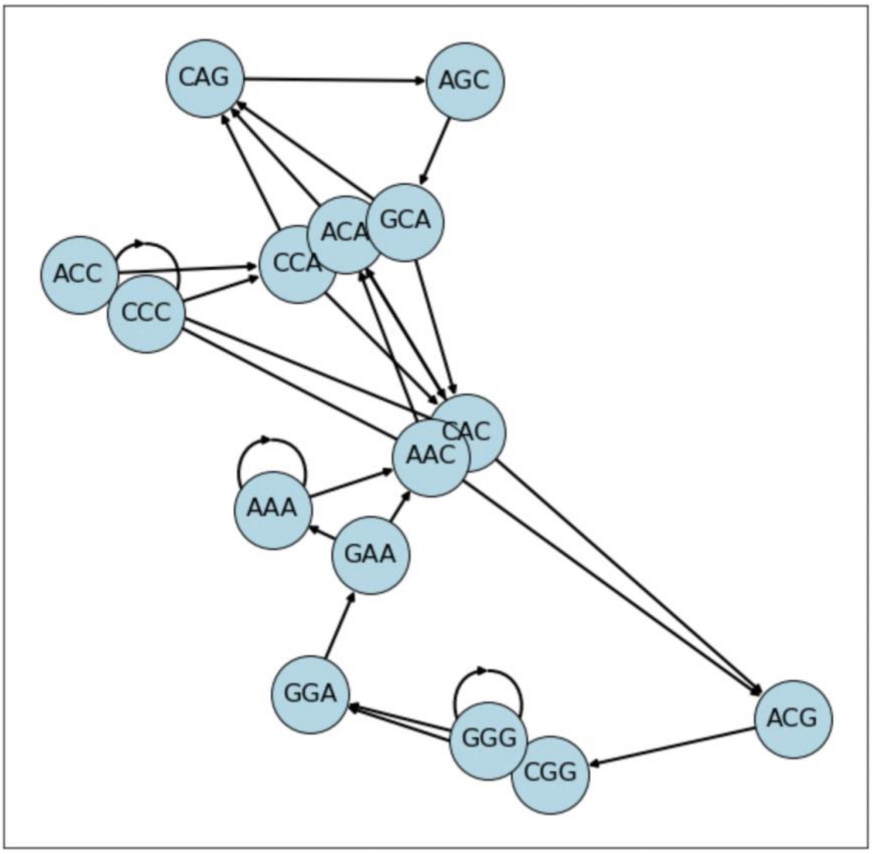
```
# the below values are taken by running all the types of string reconstruction methods
# Hamiltonian
# Debruijn
# PairedDebruijn
runTime1=0.49052729100000647
runTime2=0.25074170900001036
runTime3=0.21436370899999702
runTime4=0.39986495800008015
runTime5=0.21117904200013982
runTime6=0.18990462500005378
```

```
x=[[runTime1,3,'Hamiltonian'],[runTime2,3,'Debruijn'],[runTime3,3,'PairedDebruijn'],[runTime4,5,'Hamiltonian'],[runT
plt.figure(1,figsize=(7,7))
for i in range(len(x)):
    if x[i][2]=='Hamiltonian':
        col='red'
    elif x[i][2]=='Debruijn':
        col='green'
    else:
        col='blue'
    plt.scatter(x[i][0],x[i][1],s=50, c=col)

plt.legend(["Hamiltonian", "Debruijn","PairedDebruijn"], loc ="upper right")
plt.xlabel("Time(s)")
plt.ylabel("K-mer")
```

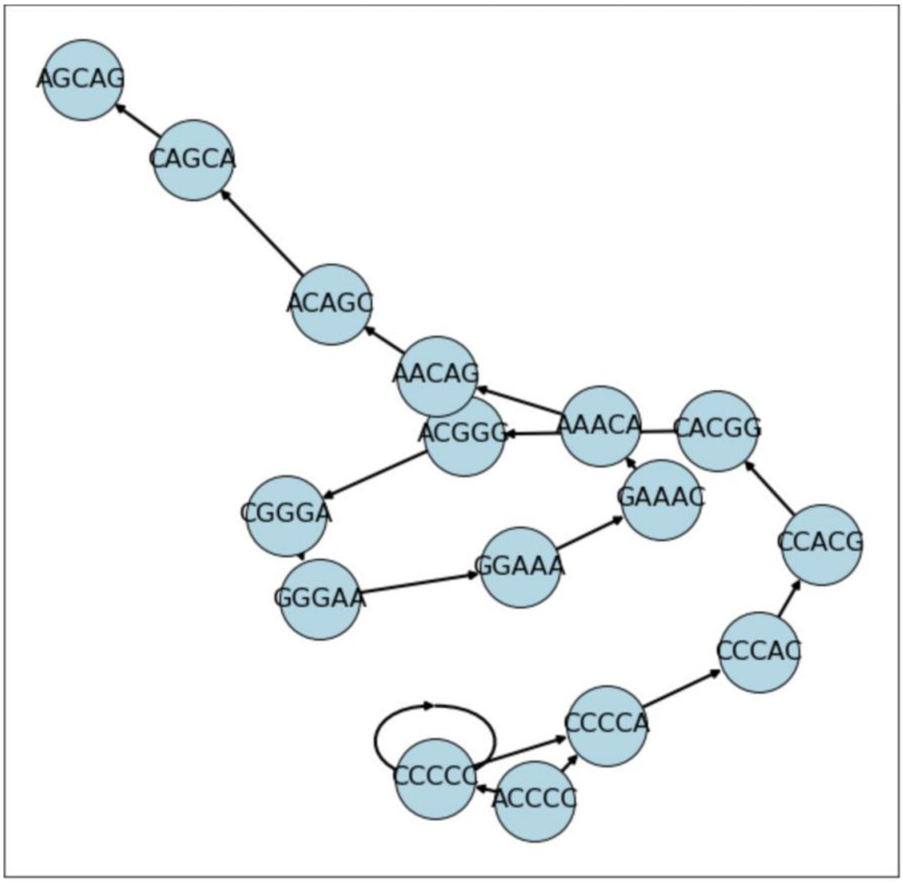


Hamiltonian:

K=3:

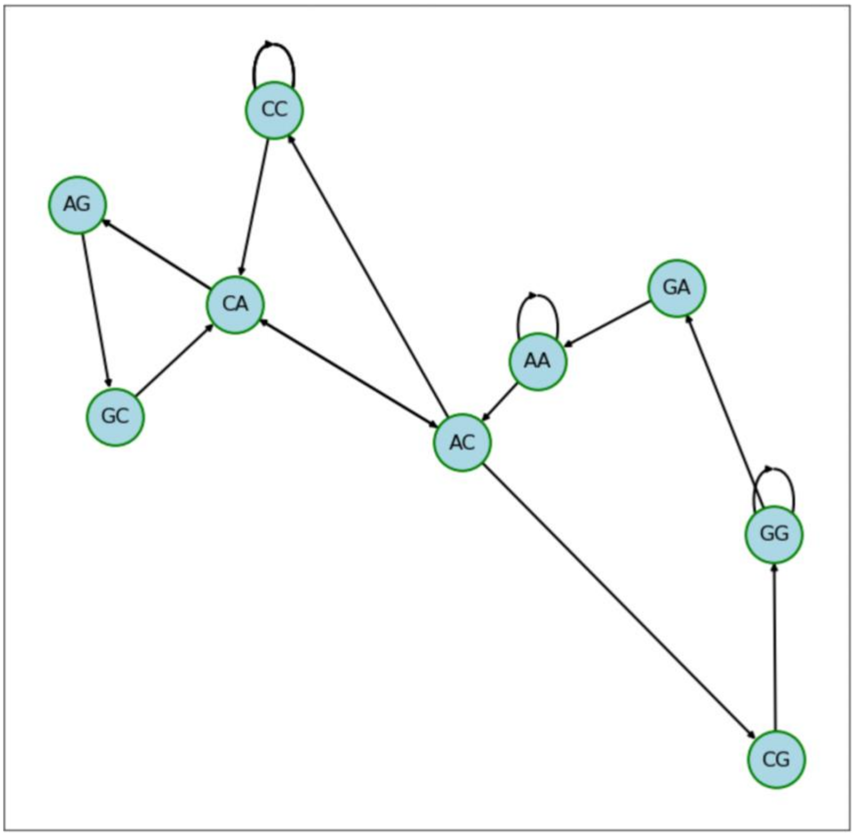


K=5:

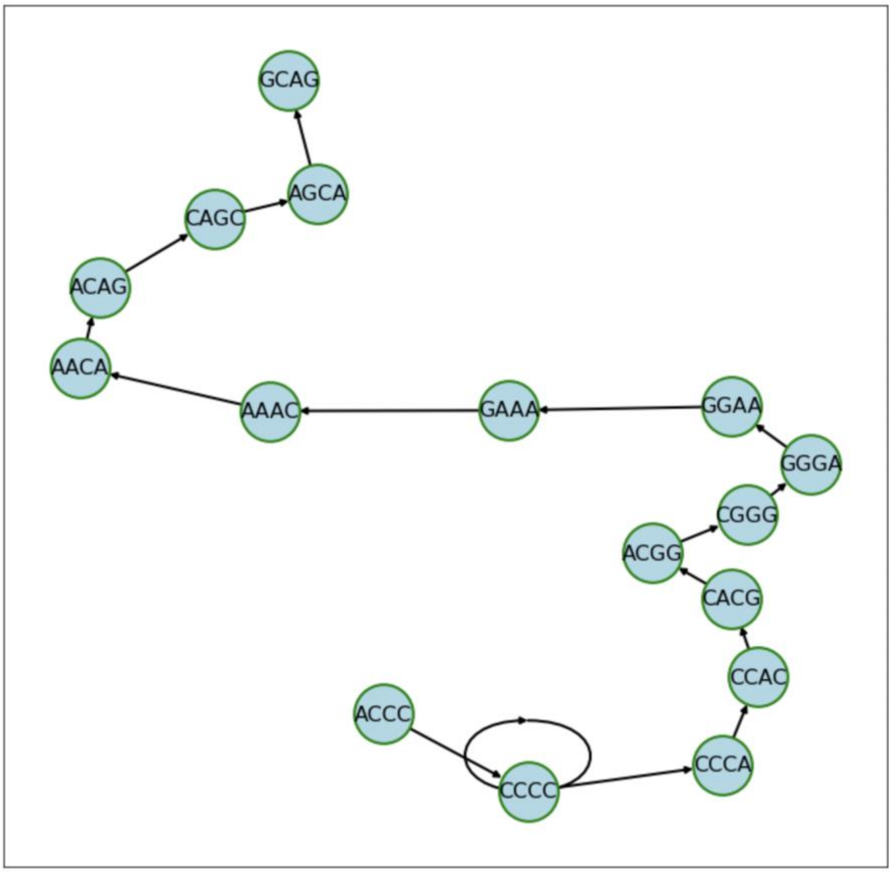


DEBRUIJN:

K=3:

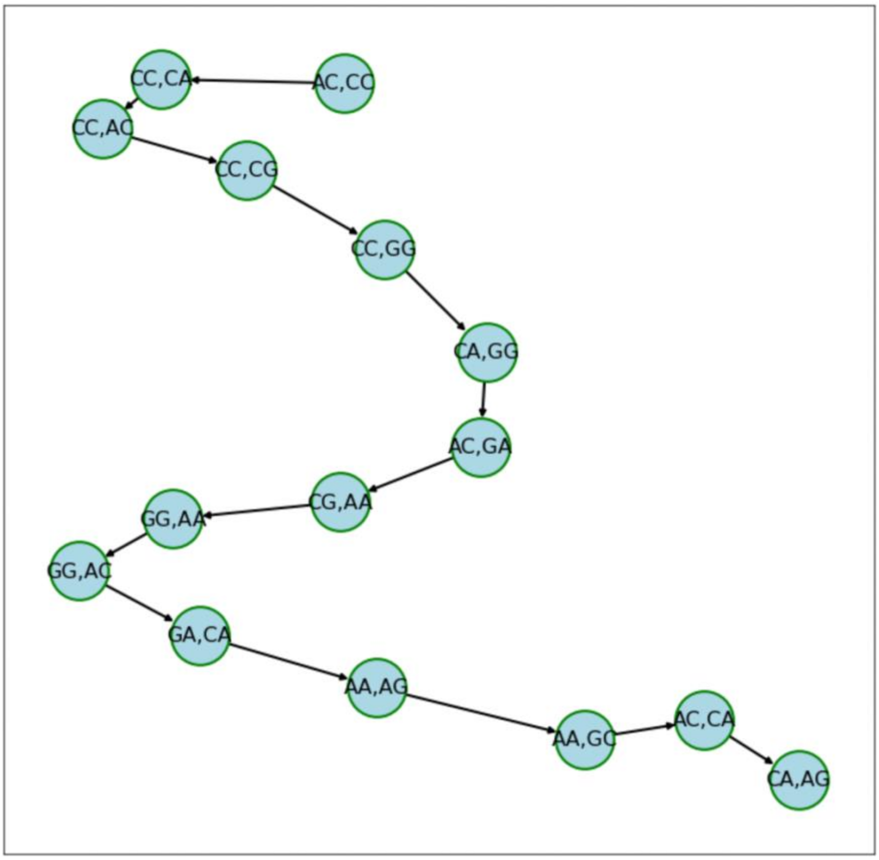


K=5:

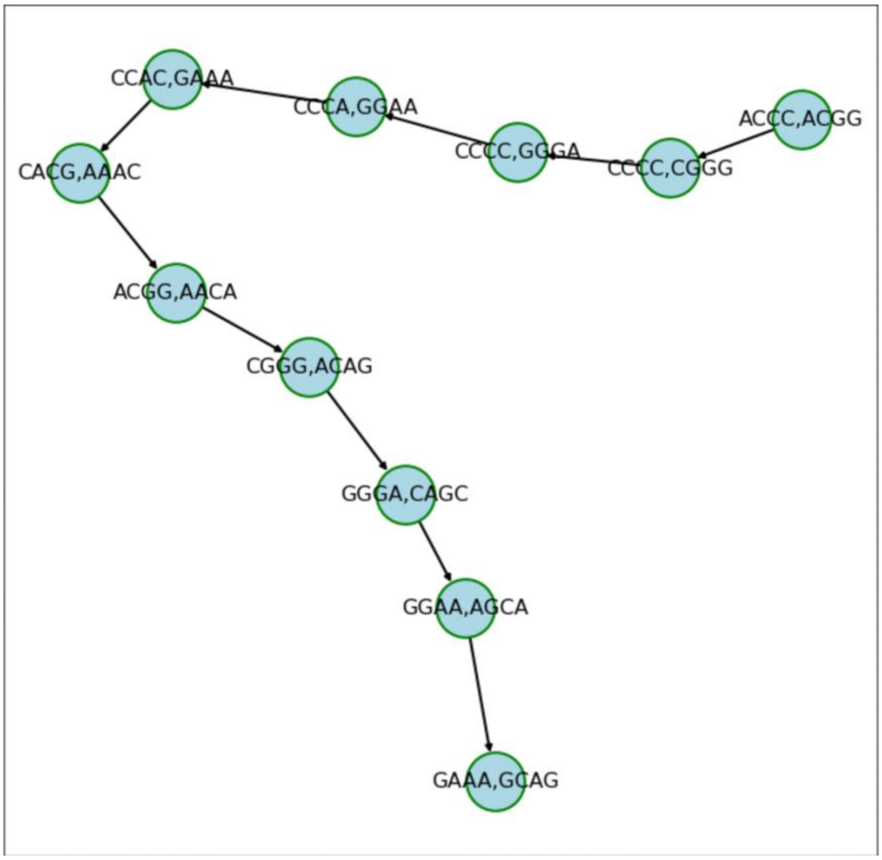


PAIRED_DEBRUIJN:

K=3:



K=5:



K	Hamiltonian	Debruijn	Paired_Debruijn
K=3 StringLength 20	0.5271	0.2849	0.20436
K=3 StringLength 400	Infinitely long	1.5525	1.76343
K=3 StringLength 51200	Infinitely long	183.43	179.105
K=5 StringLength 20	0.4836	0.3455	0.2043
K=5 StringLength 400	Infinitely long	2.1107	1.5973
K=5 StringLength 51200	Infinitely long	179.69	152.9707

NOTE:

By analyzing the above tabulation column, we can notice that the paired Debruijn graph gives us the reconstructed string in the shortest time, and one more important factor for declaring that **paired debruijn graph** is a better way is that it gives us a **unique** reconstructed string.

End