Big data , Biology Module 1 – End Sem

1) RollNo. : CS17B005

Name : ARABHI SUBHASH

District : VIZIANAGARAM

Sequence I:

CS17B005ARABHISUBHASHVIZIANAGARAM

Sequence of omics:

GATGCCCTTATCCCATCCTACCCACTGTATATA

3-mers :

{ GAT, ATG, TGC, GCC, CCC, CCT, CTT, TTA, TAT, ATC
TCC, CCC, CCA, CAT, ATC, TCC, CCT, CTA, TAC
ACC, CCC, CCA, CAC, ACT, CTG, TGT, GTA, TAT,
ATA, TAT, ATA }

3-mers, sorted:

{ ACC, ACT, ATA, ATA, ATC, ATC, ATG, CAC, CAT, CCA
CCA, CCC, CCC, CCC, CCT, CCT, CTA, CTG, CTT, GAT
GCC, GTA, TAC, TAT, TAT, TAT, TCC, TCC, TGC, TGT
TTA }

4-mers:

{ GATC, ATGC, TGCC, GCCC, CCCT, CCTT, CTTA, TTAT
TATC, ATCC, TCCC, CCCA, CCAT, CATC, ATCC, TCCT
CCTA, CTAC, TACC, ACCC, CCCA, CCAC, CACT
ACTG, CTGT, TGTA, GTAT, TATA, ATAT, TATA }

4-mers, Sorted:

{ ACCC, ACTG, ATAT, ATCC, ATCC, ATGC, CACT,

CATC, CCAC, CCAT, CCCA, CCCA, CCCT, CCTA, CCTT
CTAC, CTGT, CTTA, GATG, GCCC, GTAT, TACC
TATA, TATA, TATC, TCCC, TCCT, TGCC, TGTA
TTAT }

5-mers:

{ GATGC, ATGCC, TGCCC, GCCCT, CCCTT, CCTTA, CTTAT,
TTATC, TATCC, ATCCC, TCCCA, CCCAT, CCATC, CATCC
ATCCT, TCCTA, CCTAC, CTACC, TACCC, ACCCA,
CCCAC, CCACT, CACTG, ACTGT, CTGTA, TGTAT
GTATA, TATAT, ATATA }

5-mers, sorted:

{ ACCCA, ACTGT, ATATA, ATCCC, ATCCT, ATGCC
CACTG, CATCC, CCACT, CCATC, CCCAC, CCCAT,
CCCTT, CCTAC, CCTTA, CTACC, CTGTA, CTTAT.
GATGC, GCCCT, GTATA, TACCC, TATAT, TATCC
TCCCA, TCCTA, TGCCC, TGTAT, TTATC }

2) RollNo. : CS17B005 DOB : 18/1/2000

Set NO. : 5+8 = 13

a) Method: i) For a k-mer prefix is first k-1 letters and suffix is last k-1 letters.

→ Consider a pair A, B from set of given k-mers, there is an edge in the overlapping graph if suffix(A) = prefix(B)

ii) Join all such A, B pairs (directed edge from A→B) to get the overlapping graph

Step-1: (S → Suffix, P → Prefix)

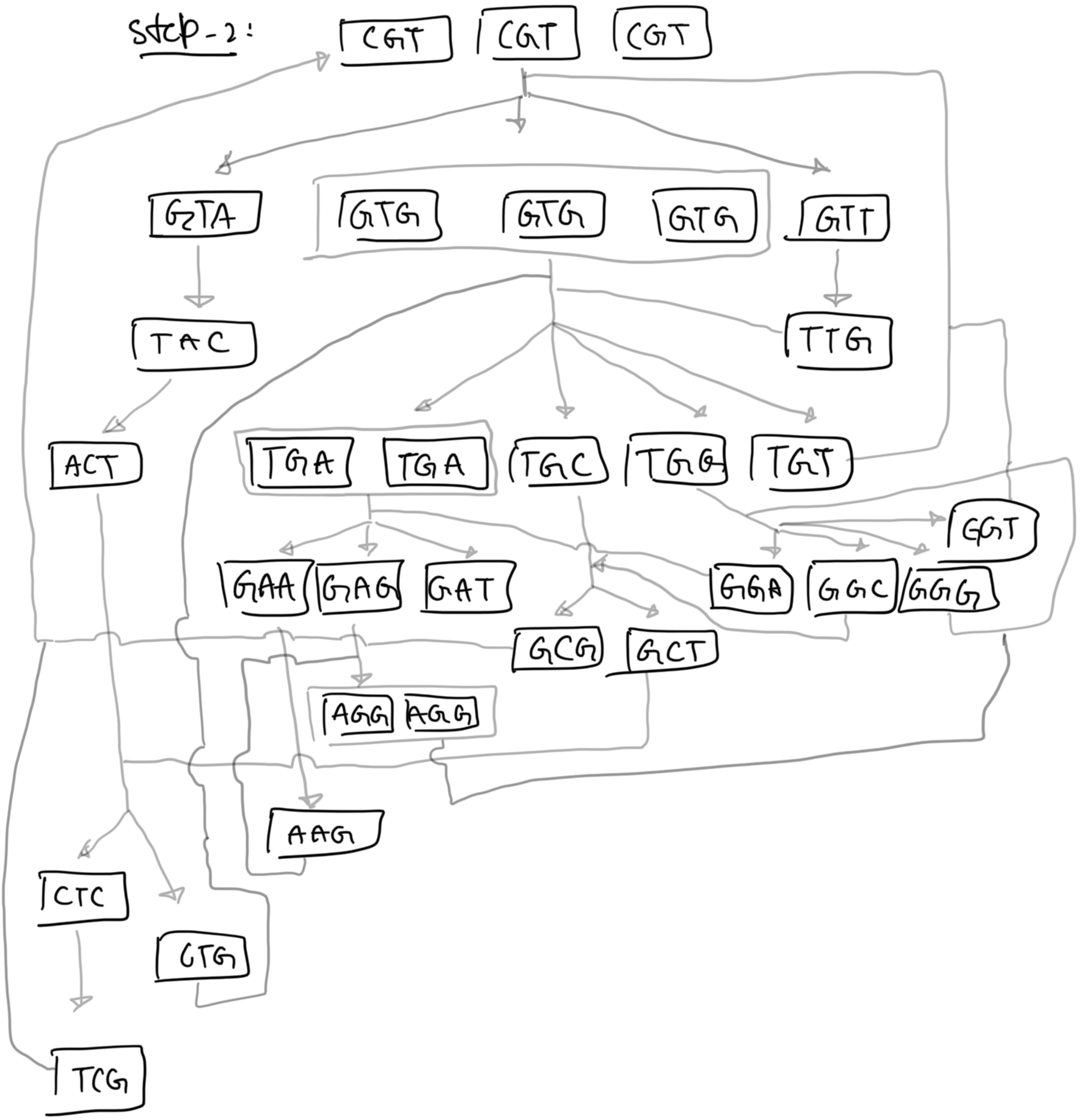
S(AAG) = AG	P(AAG) = AA
s(ACT) = CT	P(ACT) = AC
s(AGG) = GG	P(AGG) = AG
s(AGG) = GG	P(AGG) = AG
s(CGT) = GT	P(CGT) = CG
s(CGT) = GT	P(CGT) = CG
s(CGT) = GT	P(CGT) = CG
s(CTC) = TC	P(CTC) = CT
s(CTG) = TG	P(CTG) = CT
s(GAA) = AA	P(GAA) = GA
s(GAG) = AG	P(GAG) = GA
s(GAT) = AT	P(GAT) = GA
s(GCG) = CG	P(GCG) = GC
s(GCT) = CT	P(GCT) = GC
s(GGA) = GA	P(GGA) = GG
s(GGC) = GC	P(GGC) = GG
s(GGG) = GG	P(GGG) = GG
s(GGT) = GT	P(GGT) = GG
s(GTA) = TA	P(GTA) = GT
s(GTG) = TG	P(GTG) = GT
s(GTG) = TG	P(GTG) = GT
s(GTG) = TG	P(GTG) = GT
s(GTT) = TT	P(GTT) = GT
s(TAC) = AC	P(TAC) = TA
s(TCG) = CG	P(TCG) = TC
s(TGA) = GA	P(TGA) = TG
s(TGA) = GA	P(TGA) = TG
s(TGC) = GC	P(TGC) = TG

$s(TGG) = GG$ $P(TGG) = TG$

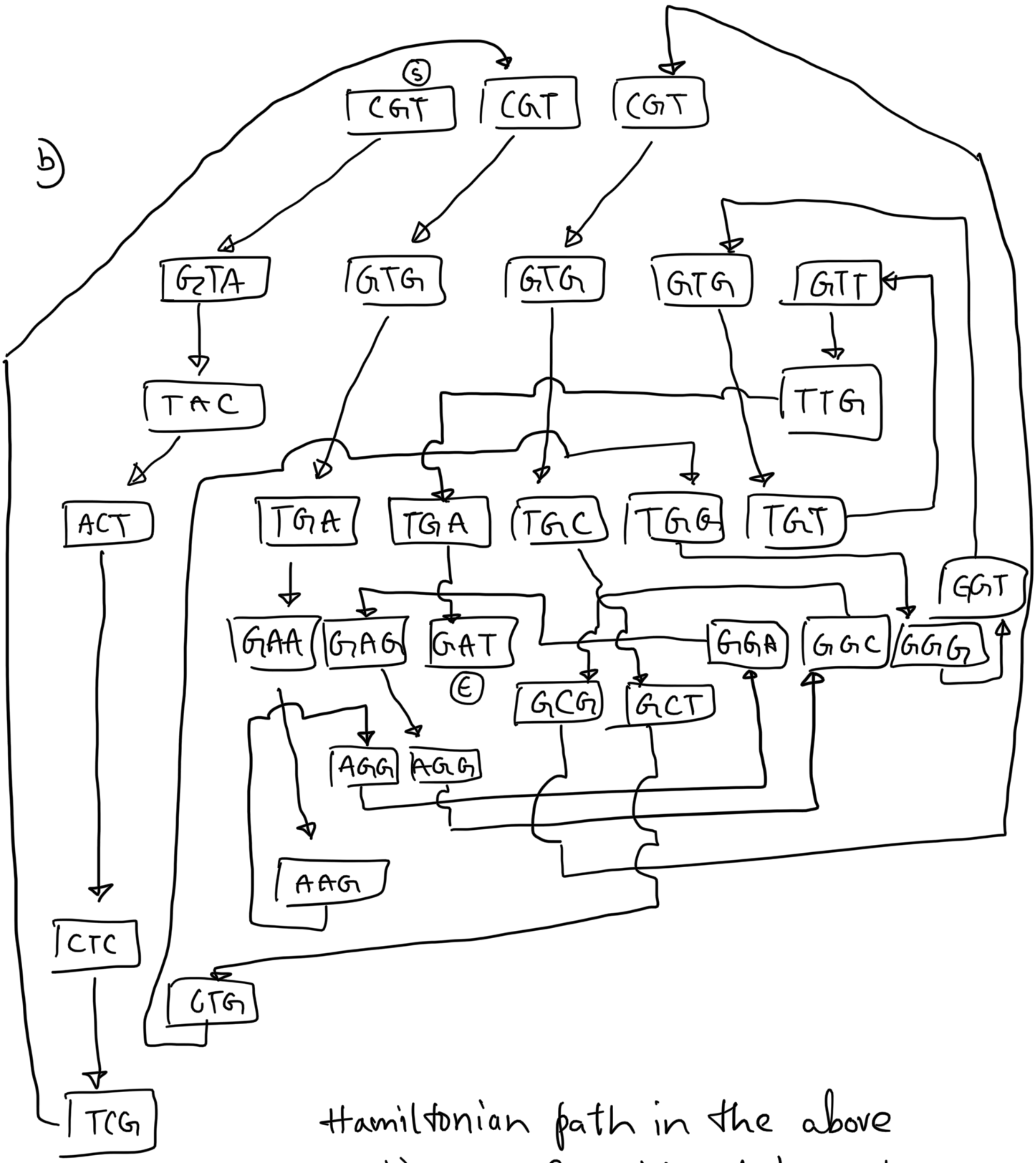
$s(TGT) = GT$ $P(TGT) = TG$

$s(TTG) = TG$ $P(TTG) = TT$

step-2:



b)



Hamiltonian path in the above overlapping graph with start & end marked

c) Reconstructing string : Follow hamiltonian path and merge last k-1 letters of this node to first k-1 letters of next node
the string is
CGTACTCGTGAAGGAGGCGTGCTGGGTGTTGAT

3) Set No. 13

S_1 = GGAGAAAAAACCACTGG

S_2 = CATGAAACACCACTGG

$m=10, S=7, d=3$

Best :

```
_GGA_GAAA_AAACCACTGG
C__ATGAAAC__ACCACTGG
```

score $= 13\times m - S\times 0 - d\times 7$

$= 130 - 21 = 109$

other-1 :

```
CAT___GAAAC_ACCACTGG
___GGAGAAAAAACCACTGG
```

Score $= 12\times m - S\times 1 - d\times 7$

$= 120 - 7 - 21$

$= 92$

other-2 :

```
CATGAAAC_ _________ACCACTGG
_________GGAGAAAAAACCACTGG
```

score $= 8\times m - S\times 0 - 17\times d$

$= 80 - 51$

$= 29$

other-3 :

```
CATGAAACACCACTGC_
GGAGAAAAAACCACTGG
```

score : $6\times m - 10\times S - 1\times d$

$= 60 - 70 - 3$

$= -13$

4) Set No. : 13

True\Pred	class 1	class 2
class 1	88 (f1)	8 (f2)
class 2	33 (f1)	12589 (f2)

a) Total Data points :

class 1 = 88 + 8 = 96

class 2 = 33 + 12589 = 12622

→ Data is highly imbalanced

→ Major class is class 2

Minor class is class 1

→ Using shannon entropy method to measure the balance of data

$$H = - \sum_{i=1}^{n} P(x_i) \log(P(x_i))$$

$$\text{Balance} = H/\log 2$$

Here $n=2$ $P(x_1) = \frac{C_1}{C_1+C_2}$, $P(x_2) = \frac{C_2}{C_1+C_2}$

$$\rightarrow H = -\frac{96}{96+12622} \log \frac{96}{12622} - \frac{12622}{96+12622} \log \frac{12622}{96+12622}$$

$$= 0.01602 + 0.003266$$

$$= 0.019285$$

→ Balance = $0.019285/\log 2 = 0.064062$

$\sim 6.4\%$.

poorly balanced data
(unbalanced)

b) Accuracy $= \dfrac{TP+FN}{TP+FP+TN+FN} = \dfrac{t_1+t_2}{t_1+f_1+t_2+f_2}$

$$= \frac{88+12589}{88+8+33+12589} = \frac{12677}{12718} = 0.99678$$

$\sim 99.68\%$.

precision, class-1 $= \dfrac{t_1}{t_1+f_1} = \dfrac{88}{88+33} = 0.72728$

precision, class 2 $= \dfrac{t_2}{t_2+f_2} = \dfrac{12589}{12589+8} = 0.99936$

Recall, class 1 $= \dfrac{t_1}{t_1+f_2} = \dfrac{88}{88+8} = 0.91667$

Recall, class 2 $= \dfrac{t_2}{t_2+f_1} = \dfrac{12589}{12589+33} = 0.99739$

F1-measure $= \dfrac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

class 1 $= \dfrac{2 \times 0.727 \times 0.9166}{0.727 + 0.9166} = 0.81106$

class 2 $= \dfrac{2 \times .99936 \times 0.99739}{.99936 + 0.99739} = 0.99837$

5) a) Small world Network :

→ A graph in which most nodes are not neighbours of one another but most nodes are reachable from every other node with small path length.

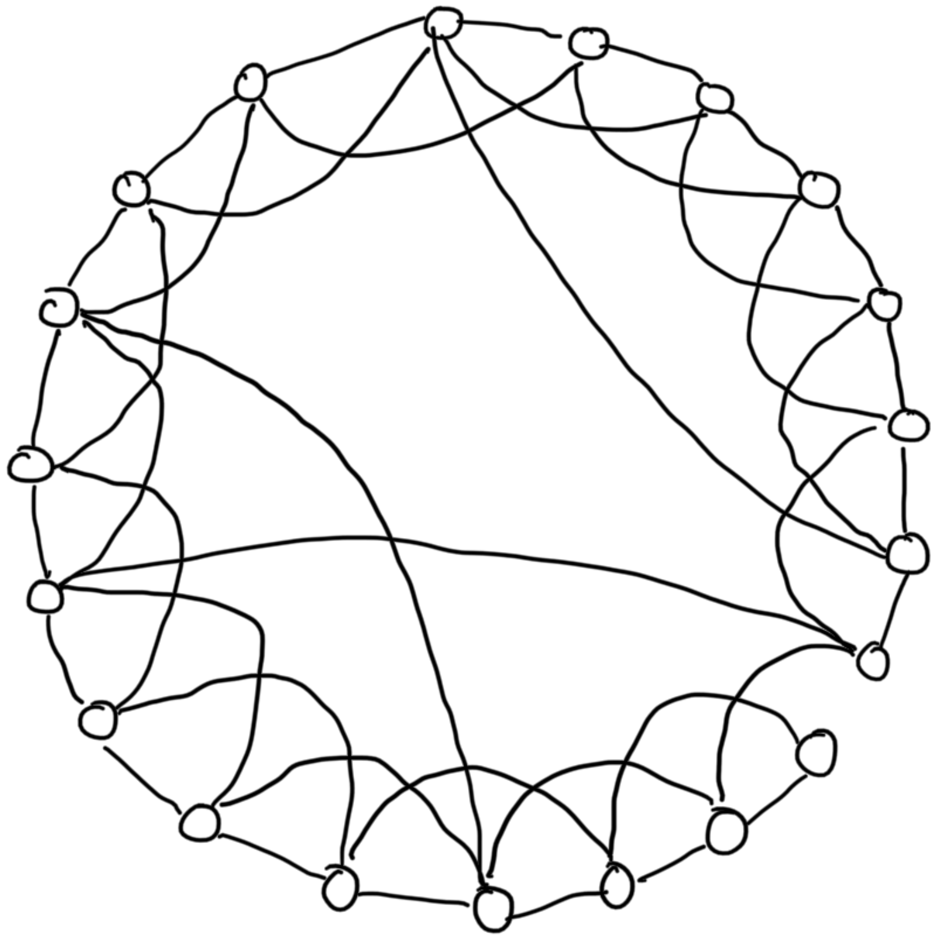
→ Neighbours are likely to be neighbors of each other

→ L - avg path length , C - clustering coefficient
n - no. of nodes in network
i) $L \propto \log(n)$

ii) L_r, C_r be L, C of equivalent random network with same avg. degree

* σ (small - coefficient) $= \dfrac{C/C_r}{L/L_r} > 1$

→



A small world Network with 20 nodes

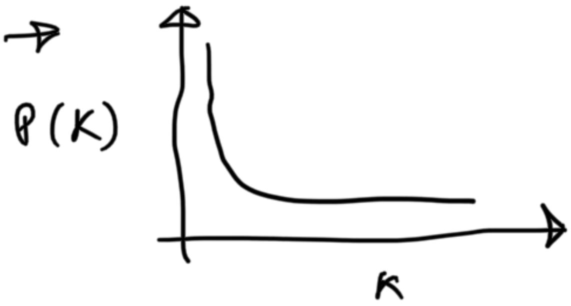
b) Scale-free network

→ A network whose degree distribution follows a power law i.e.

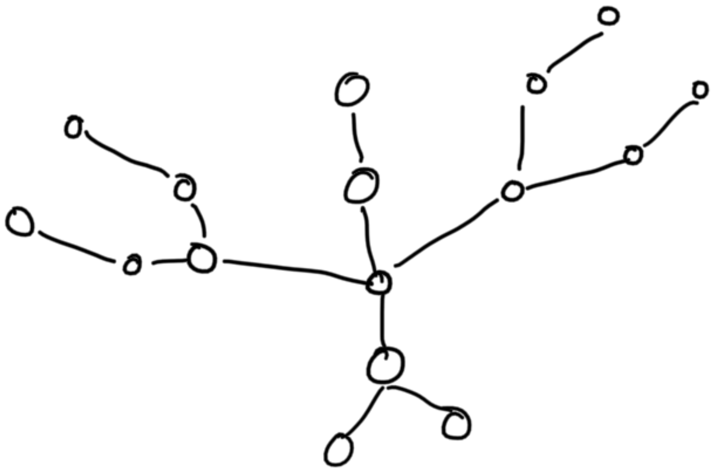
$$P(K) \sim K^{-b}$$

- $P(K)$ - prob. of nodes in network having K connections $= n_k/n$ = fraction of nodes with K-degree
- b - parameter, typically lies in $(2,3)$

→



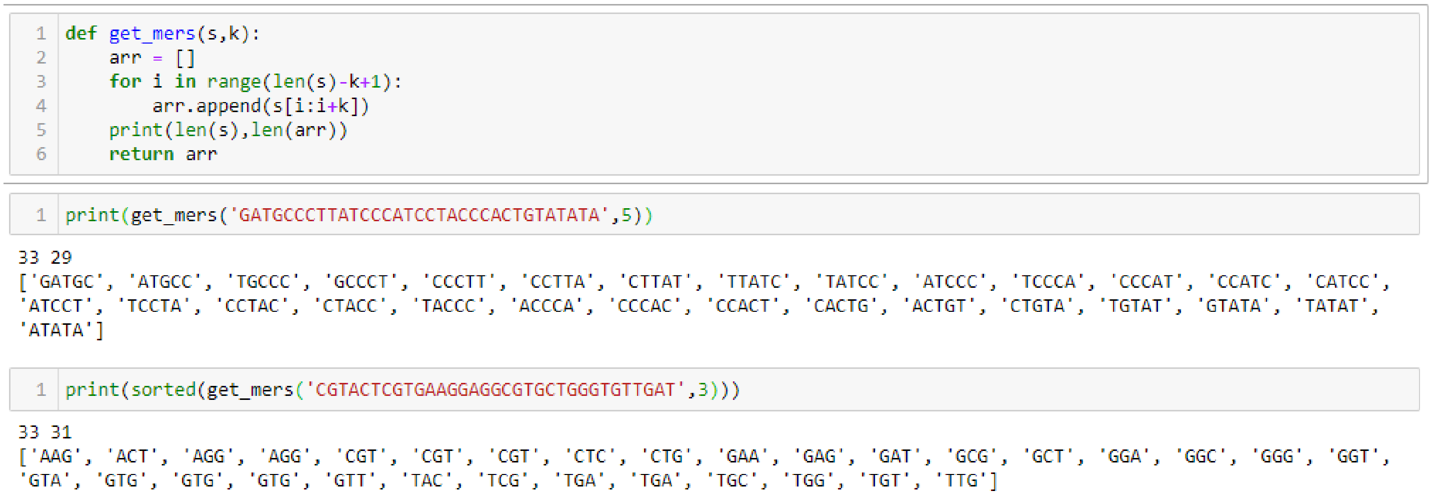
→



A scale free network with 16 nodes

Algorithms Coded in Python for different Questions

Q1: K-mers



```
def get_mers(s,k):
    arr = []
    for i in range(len(s)-k+1):
        arr.append(s[i:i+k])
    print(len(s),len(arr))
    return arr
```

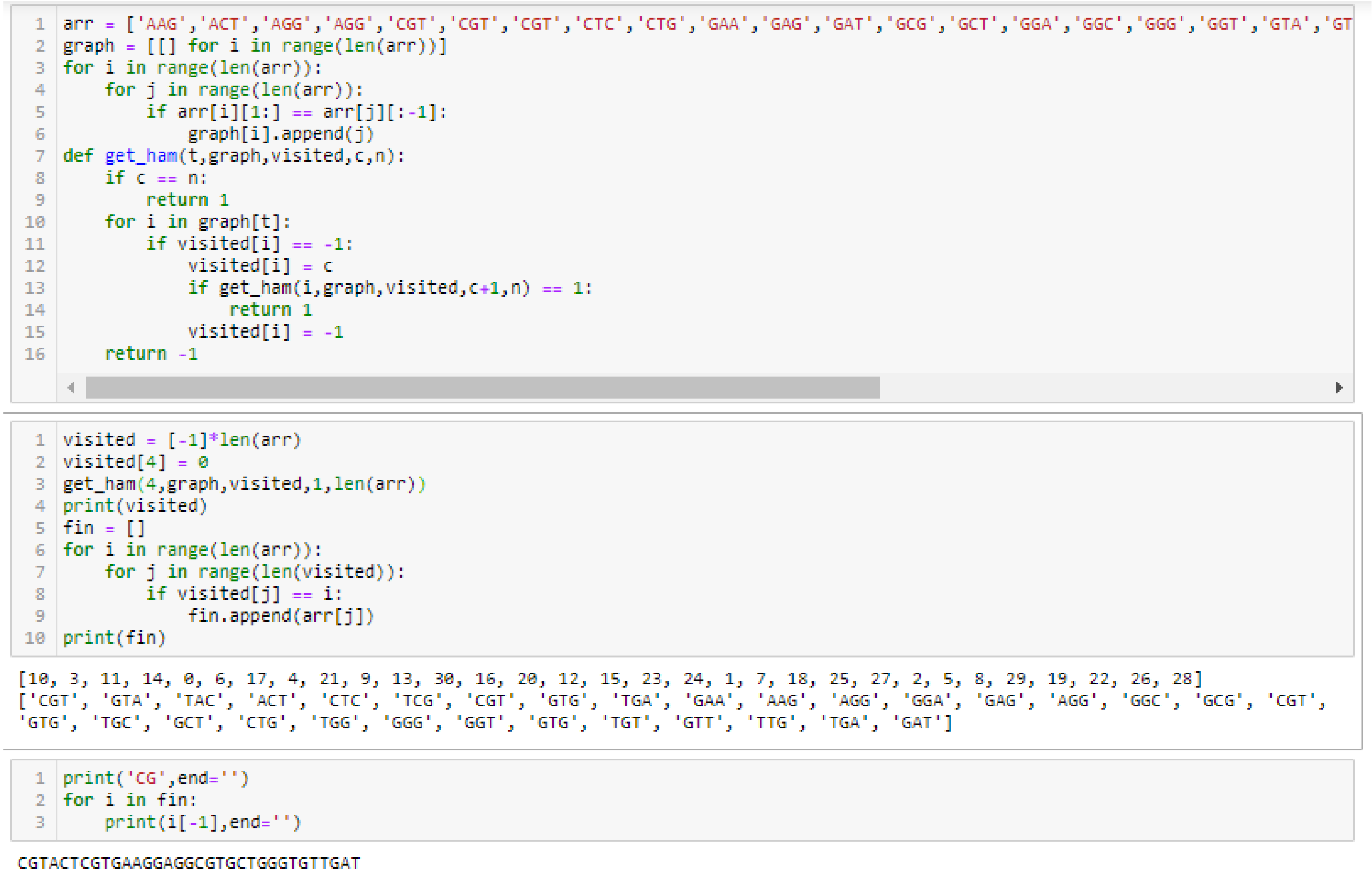
```
print(get_mers('GATGCCCTTATCCCATCCTACCCACTGTATATA',5))
```

```
33 29
['GATGC', 'ATGCC', 'TGCCC', 'GCCCT', 'CCCTT', 'CCTTA', 'CTTAT', 'TTATC', 'TATCC', 'ATCCC', 'TCCCA', 'CCCAT', 'CCATC', 'CATCC',
'ATCCT', 'TCCTA', 'CCTAC', 'CTACC', 'TACCC', 'ACCCA', 'CCCAC', 'CCACT', 'CACTG', 'ACTGT', 'CTGTA', 'TGTAT', 'GTATA', 'TATAT',
'ATATA']
```

```
print(sorted(get_mers('CGTACTCGTGAAGGAGGCGTGCTGGGTGTTGAT',3)))
```

```
33 31
['AAG', 'ACT', 'AGG', 'AGG', 'CGT', 'CGT', 'CGT', 'CTC', 'CTG', 'GAA', 'GAG', 'GAT', 'GCG', 'GCT', 'GGA', 'GGC', 'GGG', 'GGT',
'GTA', 'GTG', 'GTG', 'GTG', 'GTT', 'TAC', 'TCG', 'TGA', 'TGA', 'TGC', 'TGG', 'TGT', 'TTG']
```

Q2: Overlapping Graph and Hamiltonian Path



```
arr = ['AAG','ACT','AGG','AGG','CGT','CGT','CGT','CTC','CTG','GAA','GAG','GAT','GCG','GCT','GGA','GGC','GGG','GGT','GTA','GT
graph = [[] for i in range(len(arr))]
for i in range(len(arr)):
    for j in range(len(arr)):
        if arr[i][1:] == arr[j][:-1]:
            graph[i].append(j)
def get_ham(t,graph,visited,c,n):
    if c == n:
        return 1
    for i in graph[t]:
        if visited[i] == -1:
            visited[i] = c
            if get_ham(i,graph,visited,c+1,n) == 1:
                return 1
            visited[i] = -1
    return -1
```

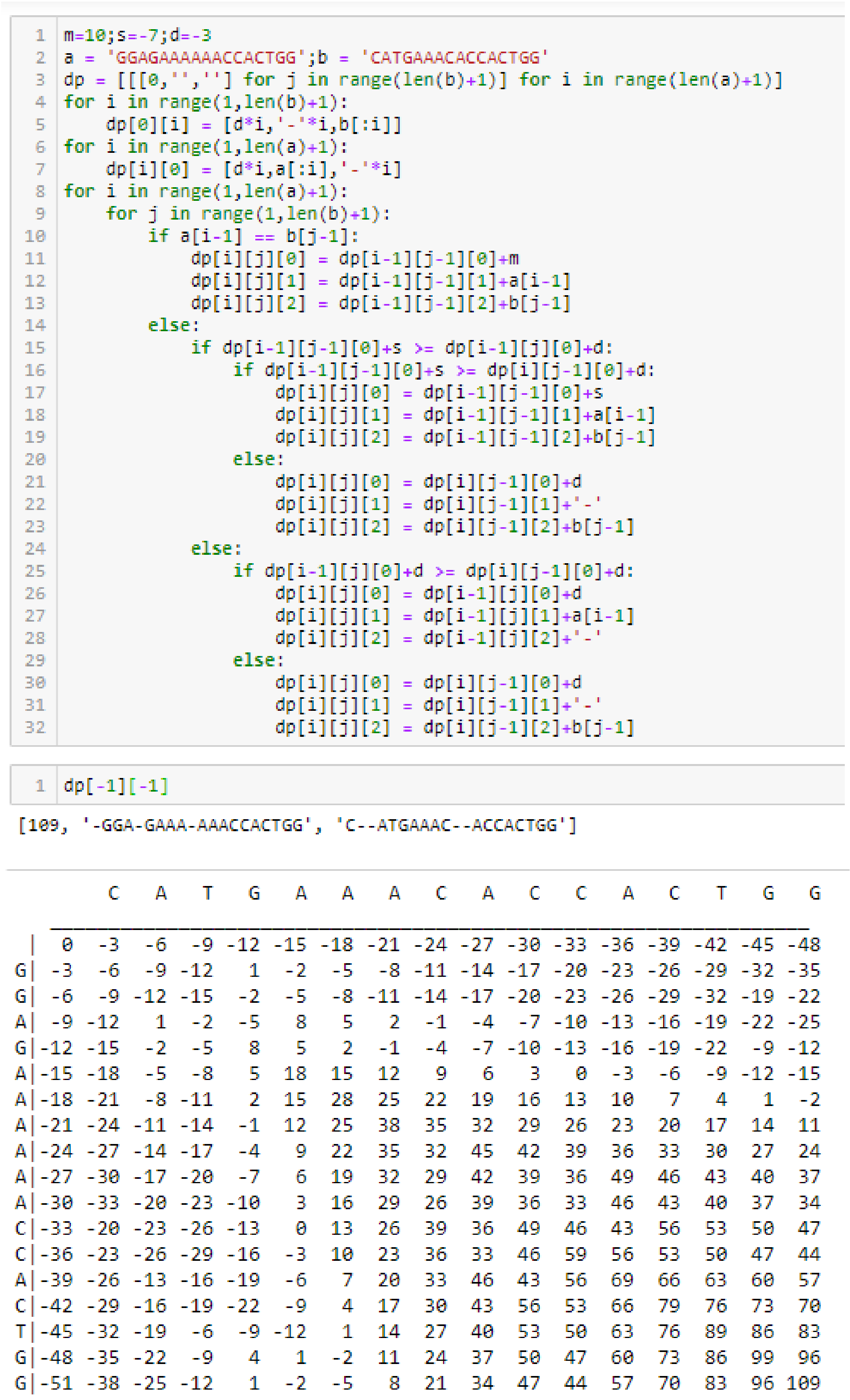
```
visited = [-1]*len(arr)
visited[4] = 0
get_ham(4,graph,visited,1,len(arr))
print(visited)
fin = []
for i in range(len(arr)):
    for j in range(len(visited)):
        if visited[j] == i:
            fin.append(arr[j])
print(fin)
```

```
[10, 3, 11, 14, 0, 6, 17, 4, 21, 9, 13, 30, 16, 20, 12, 15, 23, 24, 1, 7, 18, 25, 27, 2, 5, 8, 29, 19, 22, 26, 28]
['CGT', 'GTA', 'TAC', 'ACT', 'CTC', 'TCG', 'CGT', 'GTG', 'TGA', 'GAA', 'AAG', 'AGG', 'GGA', 'GAG', 'AGG', 'GGC', 'GCG', 'CGT',
'GTG', 'TGC', 'GCT', 'CTG', 'TGG', 'GGG', 'GGT', 'GTG', 'TGT', 'GTT', 'TTG', 'TGA', 'GAT']
```

```
print('CG',end='')
for i in fin:
    print(i[-1],end='')
```

```
CGTACTCGTGAAGGAGGCGTGCTGGGTGTTGAT
```

Q3: Dynamic Programming approach for Best Alignment and Matrix Weights



```python
m=10;s=-7;d=-3
a = 'GGAGAAAAAACCACTGG';b = 'CATGAAACACCACTGG'
dp = [[[0,'',''] for j in range(len(b)+1)] for i in range(len(a)+1)]
for i in range(1,len(b)+1):
    dp[0][i] = [d*i,'-'*i,b[:i]]
for i in range(1,len(a)+1):
    dp[i][0] = [d*i,a[:i],'-'*i]
for i in range(1,len(a)+1):
    for j in range(1,len(b)+1):
        if a[i-1] == b[j-1]:
            dp[i][j][0] = dp[i-1][j-1][0]+m
            dp[i][j][1] = dp[i-1][j-1][1]+a[i-1]
            dp[i][j][2] = dp[i-1][j-1][2]+b[j-1]
        else:
            if dp[i-1][j-1][0]+s >= dp[i-1][j][0]+d:
                if dp[i-1][j-1][0]+s >= dp[i][j-1][0]+d:
                    dp[i][j][0] = dp[i-1][j-1][0]+s
                    dp[i][j][1] = dp[i-1][j-1][1]+a[i-1]
                    dp[i][j][2] = dp[i-1][j-1][2]+b[j-1]
                else:
                    dp[i][j][0] = dp[i][j-1][0]+d
                    dp[i][j][1] = dp[i][j-1][1]+'-'
                    dp[i][j][2] = dp[i][j-1][2]+b[j-1]
            else:
                if dp[i-1][j][0]+d >= dp[i][j-1][0]+d:
                    dp[i][j][0] = dp[i-1][j][0]+d
                    dp[i][j][1] = dp[i-1][j][1]+a[i-1]
                    dp[i][j][2] = dp[i-1][j][2]+'-'
                else:
                    dp[i][j][0] = dp[i][j-1][0]+d
                    dp[i][j][1] = dp[i][j-1][1]+'-'
                    dp[i][j][2] = dp[i][j-1][2]+b[j-1]
```

```python
dp[-1][-1]
```

```
[109, '-GGA-GAAA-AAACCACTGG', 'C--ATGAAAC--ACCACTGG']
```

```
        C   A   T   G   A   A   A   C   A   C   C   A   C   T   G   G
  __________________________________________________________________
 |  0  -3  -6  -9 -12 -15 -18 -21 -24 -27 -30 -33 -36 -39 -42 -45 -48
G| -3  -6  -9 -12   1  -2  -5  -8 -11 -14 -17 -20 -23 -26 -29 -32 -35
G| -6  -9 -12 -15  -2  -5  -8 -11 -14 -17 -20 -23 -26 -29 -32 -19 -22
A| -9 -12   1  -2  -5   8   5   2  -1  -4  -7 -10 -13 -16 -19 -22 -25
G|-12 -15  -2  -5   8   5   2  -1  -4  -7 -10 -13 -16 -19 -22  -9 -12
A|-15 -18  -5  -8   5  18  15  12   9   6   3   0  -3  -6  -9 -12 -15
A|-18 -21  -8 -11   2  15  28  25  22  19  16  13  10   7   4   1  -2
A|-21 -24 -11 -14  -1  12  25  38  35  32  29  26  23  20  17  14  11
A|-24 -27 -14 -17  -4   9  22  35  32  45  42  39  36  33  30  27  24
A|-27 -30 -17 -20  -7   6  19  32  29  42  39  36  49  46  43  40  37
A|-30 -33 -20 -23 -10   3  16  29  26  39  36  33  46  43  40  37  34
C|-33 -20 -23 -26 -13   0  13  26  39  36  49  46  43  56  53  50  47
C|-36 -23 -26 -29 -16  -3  10  23  36  33  46  59  56  53  50  47  44
A|-39 -26 -13 -16 -19  -6   7  20  33  46  43  56  69  66  63  60  57
C|-42 -29 -16 -19 -22  -9   4  17  30  43  56  53  66  79  76  73  70
T|-45 -32 -19  -6  -9 -12   1  14  27  40  53  50  63  76  89  86  83
G|-48 -35 -22  -9   4   1  -2  11  24  37  50  47  60  73  86  99  96
G|-51 -38 -25 -12   1  -2  -5   8  21  34  47  44  57  70  83  96 109
```