

**“AÑO DEL BICENTENARIO DEL PERÚ: 200 AÑOS
DE INDEPENDENCIA”.**



**UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE
AREQUIPA**

**CIENCIA DE LA COMPUTACIÓN
BIOINFORMÁTICA**

Práctica 06

Estudiantes:

Miguel Alexander, Herrera Cooper
Milagros Celia, Cruz Mamani
Yara Jeanette, Quispe Quispe
Ingrid Sally, Espinel Quispe

Docente:

Mg. Vicente Enrique, Machaca Arceda

May 27, 2021



Contents

1	Actividad 1	2
1.1	Código	2
1.2	Pruebas	5
1.2.1	Input	5
1.2.2	Output	5
2	Actividad 2	6
2.1	Código	6
2.2	Pruebas	7
2.2.1	Input	7
2.2.2	Output	8
3	Actividad 3	10
3.1	Código	10
3.2	Resultados	10
4	Conclusiones	13
5	Repositorio	13

1 Actividad 1

Implemente el algoritmo de alineamiento de secuencias utilizando programación dinámica (Needleman–Wunsch). Evalúe sus resultados con las secuencias:

- S_1 : AAG
- S_2 : AGC

Utilice gapOpen = gapEXT EN D = -5 y la siguiente matriz de sustitución:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Table 1: Matriz de Similitud

La salida debe incluir el score matrix y todos los alineamientos posibles.

Resolución

1.1 Código

A continuación, mostramos el algoritmo implementado en Python.

```

1 import sys
2 import numpy as np
3
4 def Matriz_Sustitucion(file):
5     next(file)
6     r_c = {}
7     s = []
8     i = 0
9     for line in file:
10         if len(line.strip())!=0:
11             line=line.rstrip('\n')
12
13             r_c[line.split('\t')[0:1][0]] = i
14             i = i+1
15             s.append(list(map(int,line.split('\t')[1:])))
16     file.close()
17     return r_c, s
18
19 def Obtener_Secuencias(F, i, j, secuencia_alineada_1 = "",
20     secuencia_alineada_2 = ""):
21     if F[i][j][1]==0:
22         print (secuencia_alineada_1)
23         print (secuencia_alineada_2)
24         Obtener_Puntaje(secuencia_alineada_1, secuencia_alineada_2)
25         print ()

```

```

25     return
26
27     if len(F[i][j][1])>1:
28         direcciones = F[i][j][1]
29         for n in range(len(direcciones)):
30             F[i][j][1] = direcciones[n]
31             Obtener_Secuencias(F, i, j, secuencia_alineada_1,
32                                 secuencia_alineada_2)
33
34     else:
35         if F[i][j][1] == 'D':
36             secuencia_alineada_1 = secuencia_1[j-1] + secuencia_alineada_1
37             secuencia_alineada_2 = secuencia_2[i-1] + secuencia_alineada_2
38             i = i-1
39             j = j-1
40
41         elif F[i][j][1]=='U':
42             secuencia_alineada_1 = "-" + secuencia_alineada_2
43             secuencia_alineada_2 = secuencia_2[i-1] + secuencia_alineada_2
44             i = i-1
45
46         else:
47             secuencia_alineada_2 = "-" + secuencia_alineada_2
48             secuencia_alineada_1 = secuencia_1[j-1] + secuencia_alineada_1
49             j = j-1
50
51         Obtener_Secuencias(F, i, j, secuencia_alineada_1, secuencia_alineada_2)
52
53 def Obtener_Puntaje(secuencia_1, secuencia_2):
54     puntaje = 0
55     for i in range(len(secuencia_1)):
56         if secuencia_1[i] == secuencia_2[i]:
57             print (s[r_c[secuencia_2[i]]][r_c[secuencia_1[i]]], end='', sep='')
58             puntaje = puntaje + s[r_c[secuencia_2[i]]][r_c[secuencia_1[i]]]
59         else:
60             if secuencia_1[i]=='-' or secuencia_2[i]=='-':
61                 print ('(' ,d,')', end='', sep='')
62                 puntaje = puntaje + d
63             else:
64                 print ('(' ,s[r_c[secuencia_2[i]]][r_c[secuencia_1[i]]] ,')', end='',
65                     , sep='')
66                 puntaje = puntaje + s[r_c[secuencia_2[i]]][r_c[secuencia_1[i]]]
67
68     print ('+', end='', sep='')
69
70     print ("\b",end="")
71     print ('=', puntaje)
72
73 def Alineamiento_Global(F, i, j):
74     direcciones = ''
75
76     diag = F[i-1][j-1][0] + s[r_c[secuencia_2[i-1]]][r_c[secuencia_1[j-1]]]
77     up = F[i-1][j][0] + d
78     left = F[i][j-1][0] + d

```

```

79 F[i][j][0] = max(diag, up, left)
80
81 if F[i][j][0]==left:
82     direcciones = direcciones + 'L'
83
84 if F[i][j][0]==diag:
85     direcciones = direcciones + 'D'
86
87 if F[i][j][0]==up:
88     direcciones = direcciones + 'U'
89
90 F[i][j][1] = direcciones
91
92 if i==fila-1 and j==columna-1:
93     Guardar_Resultado(F)
94     Obtener_Secuencias(F, i, j)
95     return
96
97 elif j<columna-1:
98     Alineamiento_Global(F, i ,j+1)
99 else:
100     Alineamiento_Global(F, i+1 ,1)
101
102
103 def Guardar_Resultado(F):
104     f = open(archivo_final, "w")
105
106     f.write(' ')
107     for i in range(columna-1):
108         f.write(sequencia_1[i]+' ')
109
110     f.write('\n')
111
112     sequencia_2_t = ' ' + sequencia_2
113
114     for r in range(fila):
115         f.write(sequencia_2_t[r])
116         for c in range(columna):
117             f.write(' ( '+'{}'.format(F[r][c][0])+' '+'{}'.format(F[r][c][1])+' ')
118
119         f.write('\n')
120     f.close()
121
122     f = open(archivo_final, "r")
123     print(f.read())
124
125
126 if __name__ == "__main__":
127     filename = sys.argv[1]
128     d = int(sys.argv[2])
129     archivo_final = "resultado.txt"
130
131     file = open(filename, "r")
132
133     r_c, s = Matriz_Sustitucion(file)
134

```

```

135 secuencia_1 = "AAG"
136 secuencia_2 = "AGC"
137
138 # columnaa y fila de 0's al inicio de la matriz
139 columna = len(secuencia_1)+1
140 fila = len(secuencia_2)+1
141
142 F = np.zeros([fila, columna], dtype='i,O')
143
144 # Agregando 0's
145 for i in range(1,columna):
146     F[0][i][0] = i*d
147     F[0][i][1] = 'L'
148
149 for i in range(1,fila):
150     F[i][0][0] = i*d
151     F[i][0][1] = 'U'
152
153 Alineamiento_Global(F, 1, 1)

```

1.2 Pruebas

Ahora veremos la prueba realizada según lo solicitado en el ejercicio

1.2.1 Input

Evaluamos los resultados con las secuencias

```

1 secuencia_1 = "AAG"
2 secuencia_2 = "AGC"

```

1.2.2 Output

Seguidamente pasamos a ver el resultado generado :

```

cooper@cooper-Legion-Y545:~/Escritorio/5to/9no_SEMESTRE/BIOINFORMATICS/Semana_07
$ python3 ejercicio1.py substitution_matrix.txt -5
      A      A      G
( 0 0) (-5 L) (-10 L) (-15 L)
A (-5 U) ( 2 D) (-3 LD) (-8 L)
G (-10 U) (-3 U) (-3 D) (-1 D)
C (-15 U) (-8 U) (-8 U) (-6 U)

AAG-
A-GC
2+(-5)+2+(-5)= -6

AAG-
-AGC
(-5)+2+2+(-5)= -6

```

Figure 1: Matriz Final - Alineaciones - Puntajes

2 Actividad 2

Utilice el algoritmo implementado anteriormente, pero esta vez con secuencias reales. Se recomienda utilizar las secuencias de proteínas de la Práctica 3.

Utilice $gapOpen = gapEXTEND = -5$, $identicalMatch = 2$ y $mismatch = -2$.

Resolución

2.1 Código

A continuación, mostramos el algoritmo implementado en Python.

```
1 import sys
2 import numpy as np
3
4 def Alineamiento_Global(sequencia_1, sequencia_2, penalidad):
5     columna = len(sequencia_1)+1
6     fila = len(sequencia_2)+1
7     F = np.zeros([fila, columna], dtype=int)
8     # Agregamos 0's
9     for i in range(1, columna):
10         F[0][i] = i*penalidad
11
12     for i in range(1, fila):
13         F[i][0] = i*penalidad
14
15     for i in range(1, fila):
16         for j in range(1, columna):
17             valor = identicalMatch
18             if sequencia_2[i-1] != sequencia_1[j-1]:
19                 valor = mismatch
20             F[i][j] = max(F[i-1][j-1] + valor, F[i-1][j] + penalidad, F[i][j-1] +
                penalidad)
21     print("Matriz:\n")
22     print (F)
23
24     i = fila-1
25     j = columna-1
26
27     sequencia_alineada_1 = ""
28     sequencia_alineada_2 = ""
29
30     while (i > 0 or j > 0): # alineamiento
31         valor = identicalMatch
32         if sequencia_2[i-1] != sequencia_1[j-1]:
33             valor = mismatch
34         if (i>0 and j>0 and F[i][j] == F[i-1][j-1] + valor):
35             sequencia_alineada_1 = sequencia_1[j-1] + sequencia_alineada_1
36             sequencia_alineada_2 = sequencia_2[i-1] + sequencia_alineada_2
37             i = i-1
38             j = j-1
39         elif (i>0 and F[i][j]==F[i-1][j]+penalidad):
40             sequencia_alineada_1 = "-" + sequencia_alineada_1
41             sequencia_alineada_2 = sequencia_2[i-1] + sequencia_alineada_2
```

```

42     i = i-1
43     else:
44         secuencia_alineada_2 = "-" + secuencia_alineada_2
45         secuencia_alineada_1 = secuencia_1[j-1] + secuencia_alineada_1
46         j = j-1
47     print ()
48     print("Secuencia Alineada 1 :\n")
49     print (secuencia_alineada_1)
50     print ()
51     print("Secuencia Alineada 2:\n")
52     print (secuencia_alineada_2)
53     print("\n")
54     print()
55
56 def Obtener_Secuencia(file):
57     next(file)
58     secuencia = ""
59     for linea in file:
60         if len(linea.strip())!=0:
61             linea=linea.rstrip('\n')
62             secuencia = secuencia + linea
63     file.close()
64     return secuencia
65
66 if __name__ == "__main__":
67     file1 = sys.argv[1]
68     file2 = sys.argv[2]
69     penalidad = int(sys.argv[3])
70     identicalMatch = 2
71     mismatch = -2
72     f1 = open(file1, "r")
73     f2 = open(file2, "r")
74     secuencia_1 = Obtener_Secuencia(f1)
75     secuencia_2 = Obtener_Secuencia(f2)
76     Alineamiento_Global(secuencia_1, secuencia_2,penalidad)

```

2.2 Pruebas

2.2.1 Input

De acuerdo a lo solicitado, realizaremos las pruebas con los archivos .fasta, trabajados en la Práctica 03.

1. P21333.fasta - Filamina-A (Humano)
2. Q8BTM8.fasta - Filamina-A (Ratón)

2.2.2 Output

Seguidamente pasamos a ver el resultado generado :

```
cooper@cooper-Legion-Y545:~/Escritorio/5to/9no_SEMESTRE/BIOINFORMATICS/Semana_07
$ python3 ejercicio2.py P21333.fasta.txt Q8BTM8.fasta.txt -5
Matriz:

[[ 0 -5 -10 ... -13225 -13230 -13235]
 [ -5 2 -3 ... -13218 -13223 -13228]
 [ -10 -3 4 ... -13211 -13216 -13221]
 ...
 [-13225 -13218 -13211 ... 5006 5001 4996]
 [-13230 -13223 -13216 ... 5005 5008 5003]
 [-13235 -13228 -13221 ... 5000 5003 5010]]
```

Figure 2: Matriz Final - gapOpen = gapEXTEND = -5, identicalMatch = 2 y mismatch = -2

Alineamiento para P21333.fasta - Filamina-A (Humano)

```
Secuencia Alineada 1 :

MSSSHSRAGQSAAGAAPGGVDTRDAEMPATEKDLAEDAPWKKIQNTFTWCNEHLKCVSKRIANLQTDLSGLRLIALLEVLSSQKKMHRKHNRPTFRMQ
LENVSVALEFLDRESIKLVSDSKAIVDGNLKLILGLIWLTLIHYSISMPPMDEEEDEEAKKQTPKQRLLGWIQNKLPQLPITNFSRDWQSGRALGALVDS
PGLCPDWSDWASKPVTNAREAMQADDWLGIQVITPEEIVDPNVDEHSMVTLSSQFPKAKLPGAPLRPKLNPKKARAYGPGIEPTGNMVKKRAEFTVETR
SAGQGEVLVYVEDPAGHQEEAKVTANNDKNRTFSVWVPEVTGTHKVTVLFAQHIKSPFEVYVDKSSQGDASKVTAQGPGLPSGNIANKTITYFEIFTAGAG
TGEVEVVIQDPMGQKGTVEPQLEARGDSTYRCSYQPTMEGVHTVHVTFAGVPIPRSPYTVTVGQACNPSACRAVGRGLQPKGVVRKETADFKVYTKGAGSGEL
KVTVKGPKEERVQKDLGDGVYGEYPMVPGTYIVTITWGGQNIGRSPFEVKGTECGNQKVRWAGPGLGGVVGKSADVFVEAIGDDVGTGLGFSVEGPSQ
AKIECDDKGDGSCDVRYPQEAAGEYAVHVLNSEDIRLSPFMADIRDAQDFHPDRVKARGPGLKTVAVNKPAAFTVDAKHGGKAPLRVQVDNEGCPVEA
LVKDNNGTYSYVPRPKVHTAMVSWGGVSIPNSPFRVNVGAGSHPNKVKVYGPVAKTGLKAHEPTYFTVDCAEAGQGDVSIKICAPGVVGAEDIDF
DIIRNDTFTVKYTPRGAGSYTIMVLFADQATPTSPIRVKVESHDAKSKVAEGPGLSRTGVELGKPTHFTVNAKAAGKGLDVQFSLTKGDAVRDVIDI
HHDNTYTVKYTPVQGPVGVNVTYGGDPIKSPFSVAVSPSLDLKIKVSGLGEKVDVGKDQFTVKSAGGQGVASKIVGPSGAAPVCKVEPGLGADNSV
VRFLPREEGPYEVEVTDGVPVPGSPFPLEAVAPTSPKSKVAFGPGGLGGSAGSPARFTIDTKGAGTGGLLTVGEPCEAQLCELDNGDGTCSVSYVPTPGD
YNINILFADTHIPGSPFKAHVPCFDASKVKCSGGLERATAGEVQFQVDCSSAGSAELTIEICSEAGLPAEVYIQDHGDGHTHTITYIPLCPGAYTVTIKYG
GQVPVPNFPKLVQVEPAVDTSQVQCYGPGIEGQGVFREATTESVDARALTQTGGPHVKARVANPSGNLTETVYQDRDGMKYVEYTPYEEGLHSVDVTDGSP
VPSSPFQVPVTEGCDPSRVRVHGPQIGSGTTNKNKFTVETRGAGTGGLLAVEGPSEAKMSCMDNKGSCSVEYIPYEAGTYSLNVTYGGHQPVGSPFKVPV
HDVTDASKVKCSGGLSPGMVRANLPQSFQVDTSKAGVAPLQVKVQGPGLVEPVDVVDNADGTQTVNVVPSREGPYISVLYGDEEVPSPFKVKVLPHTDA
SKVKASGPGNLTTGVPASLPVEFTIDAKDAGEGLLAVQITDPEGKPKKTHIQDNHGTYTVAVVPDVTGRYILIKYGGDEIPFSPYRVRAVPTGDASKCTVT
VSIHGHLGAGIGPTIQIGEEVITVDTKAAGKGVKTCTVCTPDGSEVDVDVVENEDGTDFIFYTAPQPGKYVICVRFGGHVPNSPFQVTLAGDQPSVQPP
LRSQQLAPQYTYAQGGQQTWAPERPLVGVNGLDVTSLRPFDLVIPFTIKKGEITGEVRMPGSKVAQPTITDNKDGTVTVRYAPSEAGLHMDIRYDNMHPGS
PLQFYVDYVNCGHVTAYGPGTLHGTVNKPATFTVNTKDAGEGLSLAIEGPSKAEISCTDNQDGTCSVSYLPVLPDYSILVKYNEQHVPGSPFTARVTGDDS
MRMSHLKVGSAADIPINISSETDLSTLTATVPPSGREEPCLLRLRNHGVGISFVPKETGEHLVHVKNQGHVASSPIPVVISQSEIGDASRVVSGQLHEG
HTFEPAEFIIDTRDAGYGLLSLIEGPSKVDINTEDLEDGTCRVTYCPTPEGNYIINIKFADQHVPGSPFSVKVTGEGRVKESITRRRRAPSVANVSHCDLS
LKIPEISIQDMTAQVTSFGKTHEAEIVEGENHTYCIREFVPAEMGHTVSVKYKGQHVPGSPFQFTVGPLGEGGAHKVRAGGPGLERAEAGVPAEFSIWTREA
GAGGLAIAVEGPSKAEISFEDRKDGSCGVAVVQEPGDYEVSVKFNEEHIPOSPFVVPVSPSGDARRLTVSSLQESGLKVNQPAFVSLNAGAKAIDAKVH
SPSGALEECYVTEIDQDKYAVRFIPRENGVYLIDVKFNTHIPGSPFKIRVGEPEHGGDPGLVSAYGAGLEGGVTGNPAEFVVNTSNAGAGALSVTIDGPSKV
KMDCQCEPEGYRVYTPMAPGSYLISIKYGGPYHIGGSPFKAKVTGPRLVSNHSLHETSSVFVDSLTKATCAPQHAGPGPADASKVAKGLGLSKAYVGQK
SSFTVDCSKAGNNMLLVGHGPRTPCEEILVKHVGSRLYSVSYLLKDKGEYTLVVKWGEHIGSPYRVVVP
```

Figure 3: Alineamiento P21333 - gapOpen = gapEXTEND = -5, identicalMatch = 2 y mismatch = -2

Alineamiento para Q8BTM8.fasta - Filamina-A (Ratón)

Secuencia Alineada 2:

```

MSSSHSRGQSAVASPGGSDSRDAEMPATEKDLAEDAPWKKIQNTFTTRWCNEHLKCVSKRIANLQTDLS DGLRLIALLEVL SQQKMRKHNRPTFRMQ
LENVSALEFLDRESIKLVSDSKAIVDGNLKLILGLIWLILHYSISMPPMDEEEDEEAKKQTPKQRLLGWIQNKLPQLPITNFSRDWQSGRALGALVDS
PGLCPDWDSDASKPVNNAREAMQADDWLGIQVITPEEIVDPNVDEHSMVTL SQFPKAKLKPGLRPLNPKKARAYGPGIEPTGNMVKKRAEFTVETR
SAGQGEVLVYVEDPAGHQEEAKVTANNDKNRTFSVWYVPEVTGTHKVTVL FAGQHIKSPFEVYVDSQGDASKVTAQGPGLPSGNIANKTTYFEIFTAGAG
MGEVEVVIQDPTGQKGTVEPQLEARGDSTYRCSYQPTMEGVHTVHVTFAGVPIPRSPYTVTVGQACNPAACRAIGRGLQPKGVVKETADFKVYTKGAGSGEL
KVTVKGPKEERVKQKDLGDGVYGFYYPITPGTYTITWGGQNIGRSPFEVKVGTGECNQKVRWAGPGLGGIVGKSADVFVEAIGDDVGTGLGFSVEGPSQ
AKIECDKGDGSCDVRYPQEGEYAVHVLNSEDIRLSPFMADIREAPQDFHPDRVKARGPGLKTVAVNKAPEFTVDAKHAGKAPLRVQVQDNEGCSVEA
TVKDNGNGTYSYVPRPKVHTAMVSWGGVSI PNSPFRVNVGAGSHPNKVKVYGPVAKTGLKAHEPTYFTVDCTEAGQGDVSI GIKCAPGVVGPTEADIDF
DIIRNDNTFTVKYTPCGAGSYTIMVLFADQATPTSPIRVKVEPSHDASKVKAEGPGLNRTGVELGKPTHFTVNAKTAGKGLDVQFSGLAKGDAVRDVID
HHDNTYTKYIPVQGPVGVNVTYGGDHIPKSPFVGVSPSLDL SKIKVSGLDGKVDVGKQDEFTVSKGAGGQGVASKIVSPSGAAVPCVPEPGLGADNSV
VRFVPREEGPYEVEVTVYDGVVPGSPFPLEAVAPTKPSKVAFGPGGLQGGNAGSPARFTIDTKGAGTGGLLTVEGPCEAQLECLDNGDGTCSVSYVPTPGD
YNINILFADTHIPGSPFKAHVAPCFDASKVKCSGGLERATAGEVGGFQVDCSSAGSAELTIEICSEAGLPAEVYIQDHGDGHTHTITYIPLCPGAYVTIKYG
GQPVNPFPSKLQVEPAVDTSGVQCYGPGIEGQGVFREATTETFSVDARALTQTGGPHVKARVANPSGNLTDTVVQDCGDGTYKVEYTPYEEGVHSDVTDYDGP
VPSPFPQVPVTEGCDPSRVRVHVGPIQSGTGNKPNKFTVETRAGTGGLLAVEGSPSEAKMSCMDNKGSCSVEYIPYAGTYSLVNVTYGGHQPVGSPFVKVPV
HDVTDASKVKCSGGLSPGMVRANLPQS FQVDTSKAGVAPLQVKVQGPGLVEPVDVVDNADGTQTVNVPVPSREGSYSISVL YGEEVPRSPFVKVLP THDA
SKVKASGPGNLTTGVPASLPVEFTIDAKDAGEGLLAVQITDPEGKPKKTHIQDNHGTYTVA YVPDVPGRYIILIKYGGDEIPFSPYRVRVPTGDASKCTVT
VSIGGHGLGAGIGPTIQIGEEVTITVDTKAAGKGVCTCTPDGSEVDVDVVENEDGTDFI FYTAPQPGKYVICVRFGEHVNPSPFQV TALAGDQPTVQTP
LRSQQLAPQYNYPQGSQQTWIPERPMVGVNGLDVTSLRPFDLVIFTIKKGEITGEVRMPSGKVAQPSITDNKDGTVTVRYSPEAGLHEMDIRYDNMHIPGS
PLQFYVDYVNCGHITAYGPGLTHGVVNKPATFTVNTKDAGEGLSLAIEGPSKAEISCTDNQDGTCSVSYLPVLP GDYSILVKYNDQHIPGSPFTARVTGDDS
MRMSHLKVGSAADIPINISSETDL SLLTATVPPSGREEPCLLRKLRNHGVGISFVPKETGEHLVHVKKNGQH VASSPIPVVISQSEIGDASRVRSVQGLHEG
HTFEPAEFIIDTRDAGYGGLSLSIEGPSKVDINTEDLEDGTCRVTYCPTPEGN YIINIKFADQHVPGSPFSVKVTGEGRVKESITRRRRAPS VANIGSHCDLS
LKIPESIQDMTAQVTS PGSKTHEAEIVEGENHTYCFIRFVPAEMGMHTVS VKYKQGHVPGSPFQFTVGPLGEGGAHKVRAGGPGLERA EVGVAEFGIWTREA
GAGGLAIAVEGPSKAEISFEDRKDGSCGVAYVQEPGDYEVSVKFNEEHIPDS PFVVPVSPSGDARRLTVSS LQESGLKVNQPSAFVSLNGAKGAIDAKVH
SPSGALEECYVTEIDQDKYAVRFIPRENGIYLIDVKFNGTHIPGSPFKIRVGEPEHGGDPGLVSAYGAGLEGGVTGSPA E FIVNTSNAGAGALSVTIDGPSKV
KMDQCQCEPEGRVTVTPMAPGSYLISIKYGGPYHIGGSPFAKVTGPRLVSNHSLHETSSVFVDSLTKVATVPQHATSGPGPADVSKVAKGLGLSKAYVGQK
SNFTVDCSKAGNNMLLVGHGPRTPCEEILVKHMGSRYSVSYLLKDKGEYTLVVKWGDHIEIPGSPYRIMVP

```

Figure 4: Alineamiento Q8BTM8 - gapOpen = gapEXTEND = -5, identicalMatch = 2 y mismatch = -2

- Se debe tomar en cuenta que la matriz mostrada en la Figura2 está comprimida debido a la volumen de datos en los archivos fasta.
- En consecuencia podemos visualizar el resultado del alineamiento de ambas secuencias de Filamina A, tanto para el Humano como para el ratón en la Figura3 y la Figura4 respectivamente.

3 Actividad 3

Evalúe la pregunta anterior con otros valores de *gapOpen*, *apEXTEND*, *identicalMatch* y *mismatch*. Verifique si obtiene el mismo alineamiento y comente sus resultados.

Resolución

3.1 Código

En el código cambiaremos los valores *gapOPEN*, *gapEXTEND*, *identicalMatch*, *mismatch*.

3.2 Resultados

- Primero visualizaremos la matriz generada

```
C:\Users\51931\Downloads>python ejercicio2.py P21333.fasta Q8BTM8.fasta -3
Matriz:
[[ 0 -3 -6 ... -7935 -7938 -7941]
 [ -3 4 1 ... -7928 -7931 -7934]
 [ -6 1 8 ... -7921 -7924 -7927]
 ...
 [-7935 -7928 -7921 ... 10098 10095 10092]
 [-7938 -7931 -7924 ... 10102 10102 10099]
 [-7941 -7934 -7927 ... 10099 10099 10106]]
```

Figure 5: Matriz con *gapOpen* = *gapEXTEND* = -3

- Ahora las dos secuencias alineadas

```
Secuencia Alineada 1 :
MSSSHSRAGQSAAGAAGPGGVDTRDAEMPATEKDLEADAPWKKIQNTFTWCNEHLKCVSKRIANLQTDLSGDLRLIALLEVLSQLKMHKHNQRPTRFMQMLENVSALEFLDRESIKLVS
IDSKAIVDGNLKLILGLIWLILHYISIMPMWDEEEDEEAKQTPKQRLGLWIONKLPQLPITNFSRDWQSGRALGALVDSAPGLCPDWDSDWASKPVTNAREAMQADDWLGIPOVITPEE
IVDPINVDSEHSMVTYLSQFPAKALKPGAPLRPKLNPKKARAYGPGIEPTGNMVKKRAEFTVETRAGQGEVLVYVEDPAGHQEEAKVTANNDKNRTFSVWYVPEVTGTHKVTVLFAQGHIAKSP
FEVYVDKSDGASKVTAQGGPLESPGNIANKTTYFEIFTAGAGTGEVEVVIQDPMGQKGTVEPQLFARGDSTYRCSYQPTMEGVHTVHTFAGVPIPRSPYTVTVGQACNPASACRAVGRGLQP
KGVVRKETADFKVYTKGAGSGELKVTGKPKGEERVKQKDLGDGVYGFYYPMVPGTYIVTIITWGGQNIGRSPFEVKVGTGECNQKVRWAGPGLGEGVVGKSAFVVEAIGDDVGTGLGFSVEG
PSQAKIECDDKGDGSCDVRYWQEGEYAVHVLNSENIDIRLSPFMADIRDAPODFHDPDRVKARGPGLKTKGAVNKPAAEFVDAKHGGKAPLRVQVQDNEGCPVEALVKDNGNGTYSYVPR
KPVKHTAMVSWGGVSIIPNSPFRVNVGAGSHPNKVKVYGPVAKTGLKAHEPTYFTVDCAEAGQGDVSIIGKCAPGVVGPAAEDIDFDIIRNDNDFTVKYTPRGAGSYTIMVLADQATPTSP
IRVKVEPSHDASKVKAEGPGLSRTGVELGKPTHTVNAKAAGKGLDVQFSGLTGKDAVRDVIDIDHNDNTYTKYTPVQGGPVGVNVYGGDPIPKSPFSAVSPSLDLKSKIKVSLGEKVD
VGKDQEFVTKSKGAGGQGVASKIVGSPGAAPVCKVEPLGADNSVVRFLPREEGPYEVEVTVYDGVVPVGPSPFLEAVAPTKPSKVKAFGPGLOGGSAGSPARFTIDTKGAGTGGLGLTVEGP
CEAQLECLDNGDGTCSVSYVPTPGDYNINILFADTHIPGSPFKAHVVPFCDASKVKCSGPGLERATAGEVQFQVDCSSAGSAELTIEICSEAGLPAEVYIQDHGDGHTITTYIPLCPGAYT
VTIKYGGQPVNPFPSKLQVEPAVDTSVGQCYGPGIEGQGVFEATTEFSVDARALTQTGGPHVKARVANPSGNLTETVYQDRGDGMKYVEYTPYEEGLHSVDVTDGSPVPSPFQVPTVEGC
DPSRVRVHGPQIQSGTTNKNPKFTVETRGAGTGGLGLAVEGPGSEAKMSCMDNKGDSVSEYIPYEAGTYSLVNVTYGGHQVPGSPFKVPVHDVTDASKVKCSGPGLSPGMVRANLQSFQVDT
KAGVAPLQVKVQGPGLVEPVDVVDNADGTQTVNIVPSREGPYISVLYGDEEVRSPFVKVLPDTHASKVKASGPGLNNTGVPASLPVEFTIDAKDAGEGLLAVQITDPEGKPKKTHIQDN
HDGTYTVAYVPDVTGRYTLIKYGGDEIPFSYRVRVAVPTGDASKCTVTVSIIGHGLGAGIGPTIQIGEEETVITVDTKAAGKGVCTCTVCTPDGSEVDVVDVNEEDGTFDIFYTAPQPGKYVI
CVRFGGEHVPNSPFQVLTALAGDQPSVQPLRSQQLAPQYTYAQQGQQTWAPERPLVGVNGLDVTSLRPFDLVIPFTIKKGEITGEVRMPSPGKVAOPTITDNKDGTVTVRYAPSEAGLHEMDIR
YDNMHPGSLQFVYDVYVNCGHVTAAGPGLTHGVNKPATFTVNTKDAGEGGLSLAIEGSKAEISCTDNQDGTCSVSYLPVLPDGYSLVKYNEQHVPGSPFTRVTDGDSMRMSHLKVGSA
ADIPINISETDL SLLTATVVPSPGREGPCLLKLRLNGHVGISFVPKTEGHELVHVKNGQHVASSPPIPVVISQSEIGDASRVVSGQGLHEGHTFEPAEFIIDTRDAGYGGLSLIEGSPKVD
INTEDLEDGTCRVYCPTEPGNVIINIKFADQHPGSPFSVKVTGEGRVKESITRRRRAPSVAHVSHCDLSLKIPESISIQDMTAQVTSPSGKTHEAEIVEGENHTYICIRFVPAEMGHTHTVSV
KYKGQHVPGSPFQFVTVGLGEGGAHKVRAGGPGLERAAGVPAEFSTWREAGAGGLATAVEGPKSAEISFEDRKDGSCGVAYVVOEPGDYEVSVKFNEEHPDPSFVVPVSPSGDARRLTV
SSLQESGLKVNQPAFAVSLNAGAKAIDAKVHSPSGALEECYVTEIDQKYAVRFPRENGVYLIDVKFNGTHIPGSPFKIRVGEHPHGGDPGLVSAYAGLEGGVTGNPAEFVNTSNAGAG
ALSVTIDGSPKVMDCQCEPGEYRVITYTPMAPGSYLSIKYGGPYHIGGSPFKAKVTGPRLVSNHSLHETSSVFVDSLTK-AT-VPQH-ATSGPGPADSVKVAAGLGLSKAYVQKSNFTVD
CSKAGNMMLLVGVHGPRTPEEILVKHMSRLYSVSYLLKDKGEYTLVKKWGEHIGSPYRIMVP
```

Figure 6: Alineamiento P21333 - *gapOpen* = *gapEXTEND* = -3, *identicalMatch* = 4 y *mismatch* = -3

Secuencia Alineada 2:

```
MSSSHSRGQSAASVPGGSDSRDAEMPATEKDLAEDAPWKKIQNTFTTRWCNEHLKCVSKRIANLQTDLSGDLRLIALLEVL S QKKMHRKHNRPTFRQMLNENSVALEFLDRESIKLVS
IDSKAIVDGNLKLILGLIWLILHYSISMPMWDEEEDAEAKQTPKQRL LGWQNKLPQLPITNFSRDWQSGRALGALVDSAPGLCPDNDSDASKPVNNAREAMQADDWLGIQVITPEE
IVDPNVDEHSVMYLSQFPKAKLKPGLRPLNPKKARAYGPGIEPTGNMVKKRAEFTVETRAGQGEVLVYVEDPAGHQEEAKVTANNDKNRTFSWVYVPEVTGTHKVTFLFAGQHIKSP
FEVYVDKSGQDASKVTAQGPGLGPSNIAKNTTYFEIFTAGAGMGEVEVVIQDPTGQKGTVEPQL EARGDSTYRCSYQPTMEGVHTVHVT FAGVPIPRSPYTVTVGQACNPAACRAIGRGLQP
KGVVRKETADFVYTKGAGSGELKVTVKGPKEERVQKDLGDGVYGF EYPTIPGTYTITITWGGQNIQRSPFEVKVGTCEGNQKVRWANGPGL EGGIVGKSADFVVEAIGDDVGTGLGFSVEG
PSQAKIECDDKGDGSDVRVWPQEAAGEYAVHVLNSEDIRLSPFMADIREAPQDFHPRVKARGPGL EKTGAVNKPAAEFTVDAKHAGKAPLRVQVQDNEGCSVEATVKDNGNGTYSYVPR
KPVKHTAMVSWGGVSPNSPFRVNVGAGSHPNKVKVYGPVAKTGLKAHEPTYFTVDCTEAGQGDVSI GIKCAPGVVGPTEADIDFDIIRNDNDFTV KYTPCGAGSYTIMVLFADQATPTSP
IRVKVEPSHDASKVKAEGPGLNRTGVELGKPTHFTVNAKTAGKGLDVQFSLAGLAKDAVRDVIDIHDHNTYT VKYIPVQGGPVGVNNVTYGGDHIPKSPFSVGVSPLDL SKIKVSGLDGKVD
VGKDQEFVTKSGAGGQGVASKIVSPSGAAVPCKEPGLGADNSVVRFPREEGPYEVEVTVYDGVVPVGPSPFLEAVAPT KPSKVKAFGPGLQGGNAGSPARFTIDTKGAGTGGLGLTVEGP
CEAQLECLDNGDGTCSVSYVPTPEGDYINILFADTHIPGSPFAHVAPCFDASKVKCSGPGLERATAGEVQGFQVDCSSAGSAELTIEICSEAGLPAEVYIQDHGDGHTITITPLCPGAYT
VTIKYGGQPVNPFPSKLVQEPADVTSGVQCYGPGIEGQGVFREATT EFSVDARALTOTGGPHVKARVANPSGNLTDITYVQDCGDGT YKVEYTPYEEGVHSVDVTYDGSVPVSSPFQVPVTEGC
DPSRVRVHGPQISQSTNKNPKFTVETRGAGTGGLGLAVEGPSEAKMSCMDNKGSCSVEYIPYEAGTYS LNVTYGGHQVPGSPFKVPVHDVT DASKVKCSGGLSPGMVRANLPQS FQVDT S
KAGVAPLVKQVQGPGLVEPVDVNDAGTQTVNYVPSREGSYSISVL YGEEVPRSPFKVKVLP THDASKVKASGGLNTTGVPASLPVEFT IDAKDAGEGLLAVQITDPEGKPKKTHIQDN
HDGTYTVAYVPDVPGRYITILIKYGGDEIPFSPYRVRVPTGDASKCTVTSIGGHLGAGIGPTIQIGEEVTITVDTKAAGKGVCTCTVCTPDGSEVDVDVVENEDGTDFIYFAPQPGKYVI
CVRFGGEHVPNSPFQVTLAAGDQPTVQTPLSQQLAPQYNYPGQSQTTHIPERPMVGVNGLDVTSLRPFDLVIPFTIKKGEITGEVRMPSGKVAQPSITDNKDGTVTVRYSPEAGLHEMDIR
YDNMHPGSLQFVYDVNCGHITAYGPGLTHGVNKPATFTVNTKDAGEGGLSLAIEGSKAEISCTDNQDGTCSVSYLPVLP GDYSILVKYNDQHIPGSPFTARVTGDDSMRMSHLKVGSA
ADIPINISETDL SLLTATVPPSGREEPCLLRLRNGHVGISFVPKETEGLHVVHKNQGHVASSPIPVVISQSEIGDASRVVSGQGLHEGHTFEPAEF IIDTRDAGYGLSLIEGSPKVD
INTEDLEDGTCRVTCPTPEGNVIINIKFADQHVPSPFSVKVTGEGRVKESITRRRRAPS VANIGSHCDLSLK IPEISIQDMTAQVTPSPSGKTHEAEIVEGENHTYCI R FVPAEMGMHTSV
KYKGQHVPGSPFQFTVGLGEGGAHKVRAGGPGLERA EVGVAEFGIWTREAGAGGLAIAVEGSKAEISFEDRKDGSCGVAYVQEPGDYEVSVKFNEEHIPDSPFVVPVSPSGDARRLTV
SSLQESGLKVNQPAFAVSLNAGKAIDAKVHSPSGAL EECYVTEIDQDKYAVRFIPRENGIYLIDVKFNGTHIPGSPFKIRVGEPPHGGDPGLVSAYGAGLEGGVTGSPA E FIVNTSNAGAG
ALSVTIDGSPKVKMDCQCEPGRVYTPMAPGSYLISIKYGGPYHIGGSPFKAKVTGPRLVSNHSLHETSSVFDLSLTKVAT-VPQH-ATSGPGPADVSKVAKGLGLSKAYVGQKSNFTVD
CSKAGNNMLLVGHGPRTPCEEILVKHMGSRYSVSYLLKDKGEYTLVWKGDHPIGSPYRIMVP
```

Figure 7: Alineamiento Q8BTM8 - gapOpen = gapEXTEND = -3, identicalMatch = 4 y mismatch = -3

- Observamos el dotmather de cada uno

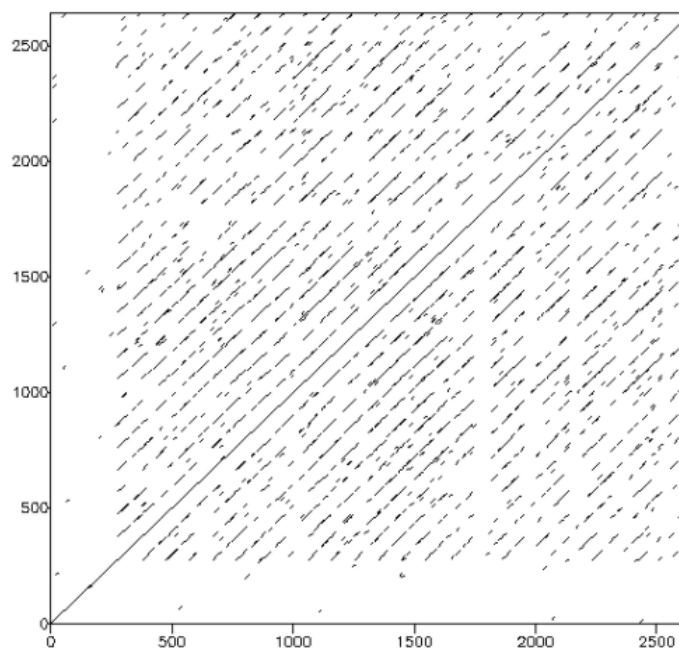


Figure 8: Alineamiento Global del Ejercicio 2

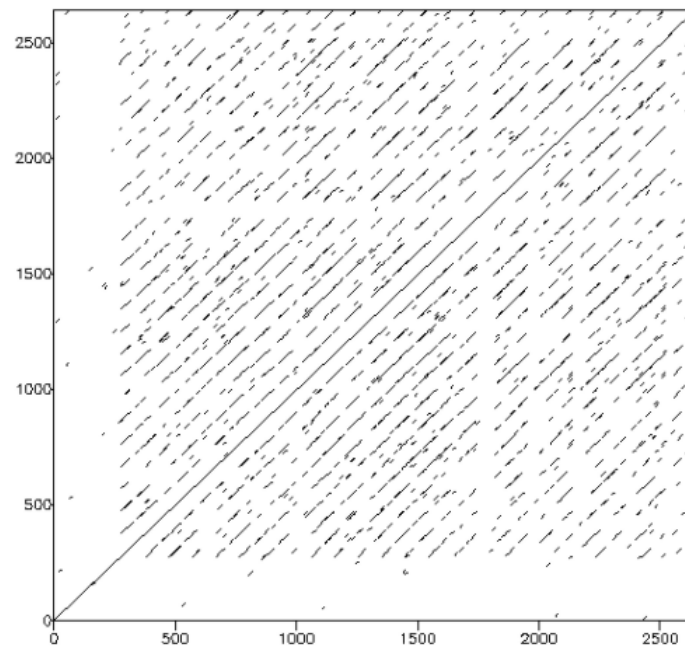


Figure 9: Alineamiento global del Ejercicio 3

- A simple vista parecen iguales, entonces los analizamos un poco más.

```
THIPGSPFKIRVGEPGHGGDPLVSAYGAGLEGGVTGNPAEFVNTSNAGAGALSVTIDGPSKVKMDCQCEPEGYRVITYTPMAPGS
YLISIKYGGPYHIGGSPFKAKVTGPRLVSNHSLHETSSVFVDSLTKATCAPQHAGPGGPADASKVVAKGLGLSKAYVGQKSFTV
DCSKAGNNMLLVGVHGPRTPCEEILVKHVGSRLYSVSYLLKDKGEYTLVVKWGDHIGSPYRVVVP
```

Figure 10: Secuencia 1 de Alineamiento del Ejercicio 2

```
YLISIKYGGPYHIGGSPFKAKVTGPRLVSNHSLHETSSVFVDSLTKATVPQH
ATSGPGPADVSKVVAKGLGLSKAYVGQKSFTVDCSKAGNNMLLVGVHGPRTPCEEILVKHVGSRLYSVSYLLKDK-
GEYTLVVKWGDHIGSPYRVVVP
```

Figure 11: Secuencia 1 de alineamiento del Ejercicio 3

- Como se muestra en el imagen hay diferencias pocas pero las hay, lo mismo ocurre con la Secuenciación 2 de cada ejercicio

4 Conclusiones

1. Programando el Algoritmo Needleman-Wunsch se obtuvo una manera más rápida y práctica de encontrar un alineamiento global, de esta manera se logra automatizar dicho proceso para n -casos de secuencias.
2. En términos de Programación Dinámica se usa un enfoque de abajo hacia arriba siendo de ayuda para los algoritmos de alineamiento de cadenas. Computacionalmente la complejidad del algoritmo (Needleman-Wunsch) para dos secuencias de longitud n y m es $O(mn)$.
3. Se probó el algoritmo con secuencias reales de proteínas y nos dio como resultado nuestra matriz y las dos secuencias de alineamiento global, lo cual serviría para futuros trabajos de procesamiento genómico, teniendo un correcto alineamiento a escala se puede usar la data para predicciones con LSTM.
4. Nos dimos cuenta que aunque podamos cambiar los valores ($gapOPEN$, $gapEXTEND$, $identicalMatch$, $mismatch$) las secuencias resultantes cambian, pero su cambio es mínimo (menos del 1% de la secuencia total), lo que nos da a decir que al alineamiento global se conserva en su mayoría.
5. Los valores $gapOPEN$, $gapEXTEND$, $identicalMatch$, $misMatch$, cada vez que sean variados, lo primero que cambiará será nuestra matriz y las secuencias resultantes seguirán sufriendo adulteraciones. El mínimo dependerá de los valores antes mencionados, y el cambio puede ser imperceptible.
6. Una desventaja que se puede tener con este algoritmo es que puede ocasionar muchos problemas si se busca es un alineamiento óptimo u constante.

5 Repositorio

La resolución de la práctica la encuentra en el siguiente enlace [Practica_06](#)