BNFO 591: High Performance Computing Assignment IV

Programs in Fortran and Python:

1. *Adjacency matrix of Titin protein*
2. Finding an exact match
3. Partial exact match
4.Appendix

Authors:
Hasan Alkhairo
Skyler Kuhn
Alexandrea Stylianou

PARTS 1, 2 and 3

```
program readFile
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

implicit none

character(len=1),dimension(0:34349):: TitinSeqlist
character(len=34350) :: Titin_string
character(len=60) :: a
character(len=60) :: aa
character(len=60),dimension(0:10000000):: filelist
character(len=60),dimension(1:573):: sequencelist
character (len=20) :: aaUni
character (len=1) :: tempchar
character(len=2) :: tempkmer
character(len=2), DIMENSION(0:399) :: uni_kmer
character(len=300):: b ! index length of the sequence array
character(len=1),dimension(0:273) :: probe_seq
character(len=274) :: string_seq, match_window, match_window2
character(len=300),dimension(0:1) :: WPh_list ! inialtizing an array to put the sequences aa i
character(len=300),dimension(1:1) :: WPh_seq_list
integer :: n=0, i, j, k, counter1=0, w,x, counter2,y,
t=0,m,match_count,z,exactAlignCount,partialAlignCount,u,threshcounter, user
integer :: t1,t2,count_rate
real :: wall_clock_time
integer, DIMENSION(0:19) :: AA_count
integer, DIMENSION(0:399) :: kmer_count
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! we are using these logical statements to read in the file
logical inFile,nextFile
inFile =.TRUE.
!logical nextFile
nextFile = .TRUE.

! we are using this for the header of the matrix
aaUni = "ACDEFGHIKLMNPQRSTVWY"
!This string will also be used to count unique frequencies as well as 2mer frequencies
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! read in titin file

open(unit=99,file="Titin.txt", status="old") !first file

open(unit=44,file="WPh_Probe.txt",status="old") !second file

!!!!!!!!!!!!!!!!!! user input
print*,"Please enter a threshold percentage in decimal form"
read(*,*)user
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! read in the first file
do while(inFile)
        read(unit=99,FMT=*) a
        if (a=="B") then  !checking for non sequence lines. End of file
                inFile= .false.
        endif
        !print *, a
        filelist(n) = a
        n = n + 1 #keep track of index for filelist
end do


do 666 i=1,573,1        ! need this loop to remove the header
        !print *, filelist(i)
        sequencelist(i) = filelist(i) !adding just the sequences to this new list
666 continue


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! read in probe file
do while(nextFile) !same procedure as above, but with the other file
        read(unit=44,FMT=*) b
        if (b=="B") then  !used same logic as before
                nextFile=.false.
        endif
        !print*,b
        WPh_list(t) = b
        t = t + 1
end do
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! turn probe into a string, from an array
counter1=0
do 777 j=1,1,1 #simple loop to remove header. May not have been necessary
        WPh_seq_list(j) = WPh_list(j) !same logic to remove the header
        do 444 k=1,274,1 #In this loop we convert the list of sub sequences into one string
```

```
                    probe_seq(counter1) = WPh_seq_list(j)(k:k) !move probe sequence into another
array
                    string_seq(k:k) = WPh_seq_list(j)(k:k) !move probe sequence into a string
                    counter1 = counter1 + 1
          444 continue
777 continue

!print*,string_seq !print string, this works
!do 309 j=0,274
!         print*,probe_seq(j) !print array, this works
!309 continue
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!Making sure everything is in the new sequence array
counter1 = 0
do 667 j=1, 573,1
          do 668 k=1,60,1

                    TitinSeqlist(counter1) = sequencelist(j)(k:k)
                    !print *, sequencelist(j)
                    counter1 = counter1 + 1
          668 continue
667 continue
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! turn titin into a string. Same procedure as probe seq
do 9000 j=0,34349
          z = j + 1 !first index of array is zero, first index of a string is 1, therefore we update j
          Titin_string(z:z) = TitinSeqlist(j)
          !print*,TitinSeqlist(j)
9000 continue
print*,Titin_string
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! exact match
! Exact Alignment Count
exactAlignCount=0
do 1313 j=1, 34350 #do the length of titin. J will serve as the first index when we slice the
titin_string
          match_window = Titin_string(j:j+273) ! we do 273 bc slice is inclusive in fortran
          !match_window in the first iteration of the loop will be the first 274 AA in titin.
          if (match_window == string_seq) then !string_seq is the entire probe seq
                    exactAlignCount = exactAlignCount +1 !if perfect match add 1 to counter
                    !print*,"Its a match"
          Endif
1313 continue
```

```
!print*, exactAlignCount
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! partial match
! Partial Alignment Count
user = 274 * user !multiply the sequence length by the  user input variable
partialAlignCount=0

call system_clock(t1,count_rate) !start timing for partial alignment
do 1314 j=1, 34350 !This approach requires a nested loop. First loop will grab the
match_window from titin and then we loop over that in another nested loop
        match_window2 = Titin_string(j:j+273)
        threshcounter = 0
                do 1315 u=1, 274 !In this loop we count how many AA matches are in the titin
match window to the probe seq
                if (match_window2(u:u) == string_seq(u:u)) then
                threshcounter = threshcounter + 1 !we keep counting an AA matches
                if (threshcounter > user) then   !Once we meet the threshold for the given
percentage then we exit the loop and add one to the partialAlignCount
                        partialAlignCount = partialAlignCount + 1
                        EXIT
                endif
                endif
        1315 continue
1314 continue

call system_clock(t2) !end timing

print*, partialAlignCount

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!AA frequencies in titin
do 899 x=1,20
        tempchar = aaUni(x:x) !One AA at a time. Already sorted
        counter2 = 0
        do 669 w=0, 34349 iterate the length of titin. Notice that tempchar will be the same AA
for this nested loop. This is how we count all A and C and so one.
                !print *, TitinSeqlist(w)
                if (TitinSeqlist(w) == tempchar) then !Compare current AA to tempchar
                counter2 = counter2 + 1
                endif
        669 continue
        AA_count(x) = counter2 append that count. We know order of counts because of aaUni
899 continue
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Create unique mers using a sorted string of AA
counter2 = 0
!These loops will loop through aaUni = "ACDEFGHIKLMNPQRSTVWY" twice and create 2mers.
!We needed a list of unique 2mers before we could count them in titin.
do 788 x=1,20 !Index for first letter in string
        do 799 y=1,20 !index for second letter in string
                uni_kmer(counter2) = aaUni(x:x) // aaUni(y:y) !add unique 2mer into list
                !This will look like “AA”, “AC”,”AD”...........”CA”,”CC”,”CD”....
                counter2 = counter2 + 1
        799 continue
788 continue

do 678 x=0,399 !Count 2mers in titin using unique 2mer array created above
        counter2 = 0
        do 567 y=0, 34349
                tempkmer = TitinSeqlist(y) // TitinSeqlist(y+1)
                if (uni_kmer(x) == tempkmer) then
                counter2 = counter2 + 1
                endif
        567 continue
        kmer_count(x) = counter2
        !print *, uni_kmer(x), counter2
678 continue

!print *, "A             C               D               E               F               G
H       I       K       L       M       N       P       Q       R       S               T       V
W               Y"

counter2 = 1
do 9654 x=0, 399, 20 !Used to print and write a formatted table
        print *, aaUni(counter2:counter2), kmer_count(x:x+19) !this is working
        counter2 = counter2 + 1
9654 continue


wall_clock_time = real(t2-t1)/real(count_rate)

print *, wall_clock_time

end program readFile
```

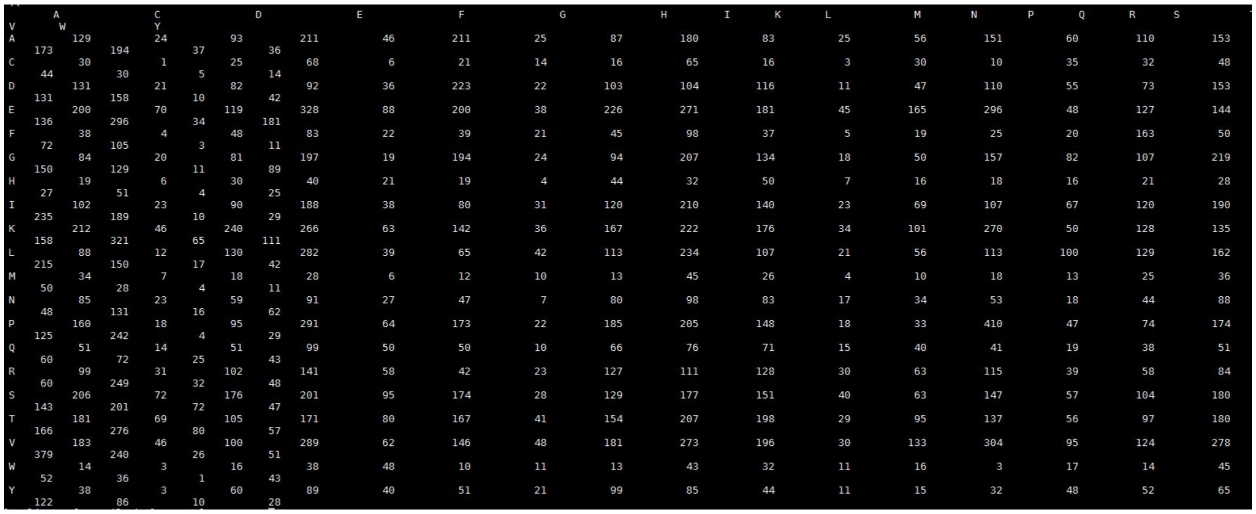
Appendix:

i. Python Code
ii . Output of Matrix To a File
iii.Histogram

i:Python for part 1

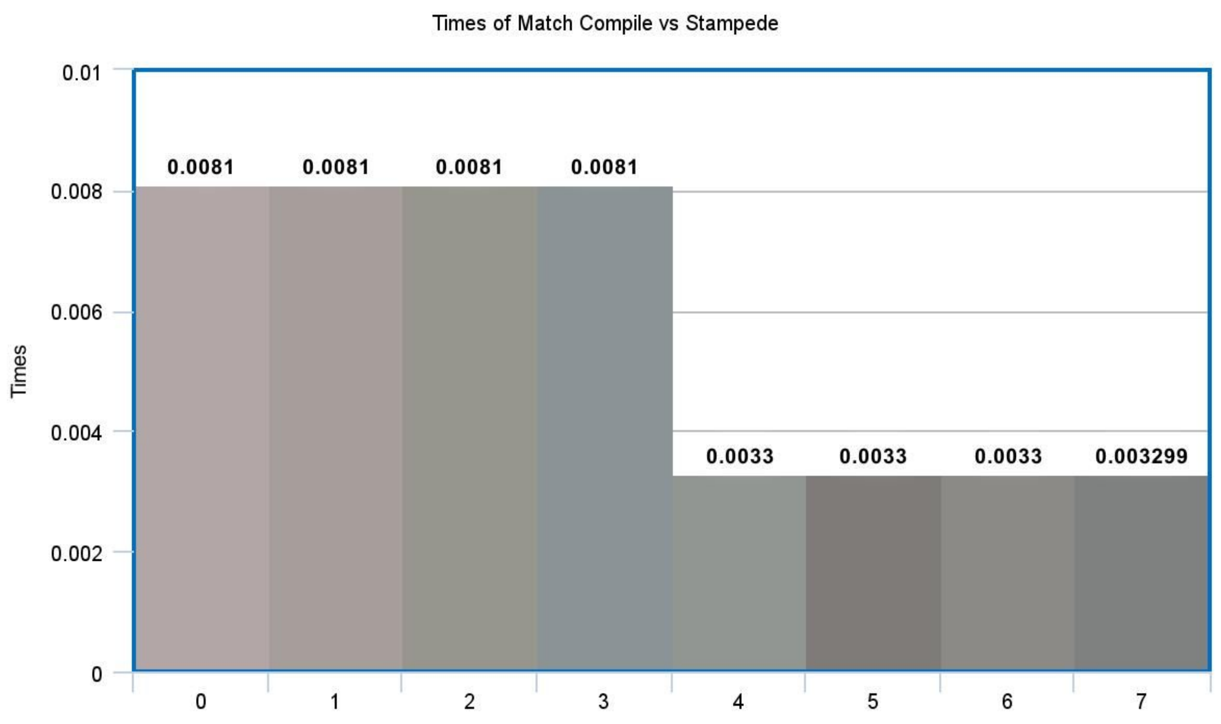
```
fh = open("titin",'r')

full_seq = ""
freq_dict = {}
kmer_dict = {}
for x in fh:
   if ">" not in x:
       x = x.strip()
       full_seq += x
for x in full_seq:
   if x in freq_dict: freq_dict[x] += 1
   else: freq_dict[x] = 1
AA_letters = []
for k, v in sorted(freq_dict.items()):
   AA_letters.append(k)

for x in range(len(full_seq)):
   if len(full_seq[x:x+2]) == 2:
       if full_seq[x:x+2] in kmer_dict:
           kmer_dict[full_seq[x:x+2]] += 1
       else:
           kmer_dict[full_seq[x:x+2]] = 1
new_dict = {}
kmer_tuple = []
for k, v in sorted(kmer_dict.items()):
   kmer_tuple.append((k,v))

for x in AA_letters: print x, "\t",
print "\n"
for x in AA_letters:
   print x,
   for y in kmer_tuple:
       if x == y[0][0]:
           print y[1],
   print "\n"
print kmer_tuple
print freq_dict
```

ii.Matrix



```
       A               C               D               E               F               G               H         I       K       L             M        N        P       Q       R      S
V       W              Y
A         129          24          93         211          46         211          25          87         180          83          25          56         151          60         110         153
    173         194          37          36
C          30           1          25          68           6          21          14          16          65          16           3          30          10          35          32          48
     44          30           5          14
D         131          21          82          92          36         223          22         103         104         116          11          47         110          55          73         153
    131         158          10          42
E         200          70         119         328          88         200          38         226         271         181          45         165         296          48         127         144
    136         296          34         181
F          38           4          48          83          22          39          21          45          98          37           5          19          25          20         163          50
     72         105           3          11
G          84          20          81         197          19         194          24          94         207         134          18          50         157          82         107         219
    150         129          11          89
H          19           6          30          40          21          19           4          44          32          50           7          16          18          16          21          28
     27          51           4          25
I         102          23          90         188          38          80          31         120         210         140          23          69         107          67         120         190
    235         189          10          29
K         212          46         240         266          63         142          36         167         222         176          34         101         270          50         128         135
    158         321          65         111
L          88          12         130         282          39          65          42         113         234         107          21          56         113         100         129         162
    215         150          17          42
M          34           7          18          28           6          12          10          13          45          26           4          10          18          13          25          36
     50          28           4          11
N          85          23          59          91          27          47           7          80          98          83          17          34          53          18          44          88
     48         131          16          62
P         160          18          95         291          64         173          22         185         205         148          18          33         410          47          74         174
    125         242           4          29
Q          51          14          51          99          50          50          10          66          76          71          15          40          41          19          38          51
     60          72          25          43
R          99          31         102         141          58          42          23         127         111         128          30          63         115          39          58          84
     60         249          32          48
S         206          72         176         201          95         174          28         129         177         151          40          63         147          57         104         180
    143         201          72          47
T         181          69         105         171          80         167          41         154         207         198          29          95         137          56          97         180
    166         276          80          57
V         183          46         100         289          62         146          48         181         273         196          30         133         304          95         124         278
    379         240          26          51
W          14           3          16          38          48          10          11          13          43          32          11          16           3          17          14          45
     52          36           1          43
Y          38           3          60          89          40          51          21          99          85          44          11          15          32          48          52          65
    122          86          10          28
```

iii.Histogram



We notice that the times are constant, no matter what the match percentage is. This is because we have a nested do loop with counters that are evaluated against thresholds. Nested do loops function in quadratic time (n*m).