Assignment 2

Bioinformatics (CIS 455)

Javier Arechalde

February 22, 2018

Problem 2-1 Jones & Pevzner, Problem 4.1

In this problem, we will be given a set X, and we will need to calculate the multiset ΔX,

Pseudocode

In our implementation, we will iterate over the list with two index, the first one to indicate the position we are calculating the distances from, and the second one to indicate the position we want to calculate the distance with. We suppose that the given list comes presorted, so we won't.

Algorithm 1 title

for i in range $1 \rightarrow length(X)$ **do**
  for j in range $j \rightarrow length(X)$ **do**
    $distance = X[j] - X[i]$
    We add the calculated distance to the ΔX array
  end for
end for

Implementation

Here is the implementation of the pseudocode listed above.

```
#First we declare the X
X =[1,2,4,6,9,14]

#We will store DX in this list
dx =[]

#Now we will iterate over the list to calculate all the distances
for i in range(0,len(X)):
 for j in range(i+1,len(X)):
  deltax = X[j]-X[i]
  dx.append(deltax)

#Print the results
print('X:')
print(X)
print('')
print('Delta X:')
print(dx)
```

After running the code over a sample dataset, we obtained the following results.

```
X:
[1, 2, 4, 6, 9, 14]

Delta X:
[1, 3, 5, 8, 13, 2, 4, 7, 12, 2, 5, 10, 3, 8, 5]
```

Problem 2-2 Jones & Pevzner, Problem 4.2

Following $ANOTHERBRUTEFORCEPDP(L, n)$

Here is the implementation code for $ANOTHERBRUTEFORCEPDP$, note that for finding all the possible sets of length $n-2$, we decided to use *itertools*, to make the implementation easier.

```
#Importing numpy module
import numpy as np
from itertools import permutations

#We are given the following partial digest
L = [1,1,1,2,2,3,3,3,4,4,5,5,6,6,6,9,9,10,11,12,15]

def anotherbruteforcepdp(L):
 M = max(L)
 Ll = len(L)

 #Calculating n
 #n^2-n-2*L = 0
 n = (1+np.sqrt(1+8*Ll))/2

 if (int(n) != n):
  print('Error')
  return

 n = int(n)

 print('Maximum: %i Length: %i ' %(M,n))

 #First we need to get the unique integers
 UL = []
 for number in L:
```

```
 if UL == []:
  UL.append(number)
 else:
  if number != UL[len(UL)-1] and number<M and number>0:
        UL.append(number)
  else:
        continue

#List of good sets
XList = []

#Valid X
Xret = []

#Now we will build all the different possible sets,
#using the itertools package
for p in permutations(UL, 5):
 valid = 1

 #We check if the set is valid
 for i in range(0,len(p)-1):
        if p[i]>=p[i+1]:
         valid = 0
 #If the set is valid, we create X
 if valid == 1:
        p = list(p)
        p.insert(0,0)
        p.append(M)
        XList.append(p) #Adding set to the list of sets

#For each set, we check if deltaX equals L
for X in XList:

 dx =[]

 for i in range(0,len(X)):
  for j in range(i+1,len(X)):
   deltax = X[j]-X[i]
   dx.append(deltax)

 #Sorting the Delta X list
 dx.sort()

 if dx == L:
        Xret.append(X)
```

```
 #If we finish iterating the for loop without finding any solution we
 #return that there is no solution

 if Xret == []:
  print('No solution')
  return

 else:
  return Xret

#We call the function and store the results
X = anotherbruteforcepdp(L)

print('')
print('X Found: ')
print(X)
```

After running the code over the given partial digest, we obtained the following results.

```
Maximum: 15 Length: 7

X Found:
[[0, 3, 4, 5, 6, 9, 15], [0, 6, 9, 10, 11, 12, 15]]
```

Problem 2-3 Jones & Pevzner, Problem 4.5

Problem 2-4 Jones & Pevzner, Problem 4.9

Pseudocode

Branch and Bound

Problem 2-5 Jones & Pevzner, Problem 4.12

Pseudocode

Time Complexity

Problem 2-6 Jones & Pevzner, Problem 4.16

Problem 2-7 Rosalind

This problem was solved entirely on **Rosalind**.

Username: *jarechalde*