Name: Fadi Alahmad Alomar
ID: 120180049

Assignment 9: The Fast Fourier Transform and Polynomial Multiplication

Question 1

(a)

$$LCS(A,B,i,j) = \begin{cases} 0 & if\ i=0\ or\ j=0 \\ max(1+LCS(A,B,i-1,j-1), LCS(A,B,i-1,j), LCS(A,B,i,j-1), & \\ LCS(A,B,i-1,j-1)) & if\ A[i]=B[j] \\ max(LCS(A,B,i-1,j-1), LCS(A,B,i-1,j), LCS(A,B,i,j-1), & \\ LCS(A,B,i-1,j-1)) & if\ A[i]!=B[j] \end{cases}$$

(b)

$$SCS(A,B,i,j) = \begin{cases} 0 & if\ i=0\ and\ j=0 \\ i & if\ i!=0\ and\ j=0 \\ j & if\ i=0\ and\ j!=0 \\ 1+SCS(A,B,i-1,j-1) & if\ A[i]=B[j] \\ min(1+SCS(A,B,i-1,j-1), 1+SCS(A,B,i-1,j), 1+SCS(A,B,i,j-1)) & if\ A[i]!=B[j] \end{cases}$$

(c)

$$LIS(X,i,j) = \begin{cases} 0 & if\ i=0 \\ max(1+LIS(X,i-1,i), LIS(X,i-1,j)) & if\ X[j]>X[i] \end{cases}$$

$$RDS(X,i,j) = \begin{cases} 0 & if\ i=len(X) \\ max(1+RDS(X,i+1,i), RDS(X,i+1,j)) & if\ X[j]<X[i] \end{cases}$$

$$LBS(X,i) = \begin{cases} 0 & ifi=len(X)-2 \\ max(LBS(X,i+1), 1+LIS(X,i,i)+RDS(X,i,i)) & otherwise \end{cases}$$

(d)

$$LOSLOW(X,i,j) = \begin{cases} 0 & if\ j>|X| \\ LOSLOW(X,i,j+1) & if\ X[i]>x[j] \\ max(LOSLOW(i,j+1), 1+LOSHIGH(i,j+1)) & \end{cases}$$

$$LOSHIGH(X,i,j) = \begin{cases} 0 & if\ j>|X| \\ LOSLOW(X,i,j+1) & if\ X[i]<x[j] \\ max(LOSLOW(i,j+1), 1+LOSHIGH(i,j+1)) & \end{cases}$$

$$LOS(X) = max_i(LOSLOW(X,i,i+1)+1)$$

(e)

$$SOSHIGH(X,i) = \begin{cases} 0 & if\ i = len(X) - 1 \\ SOSHIGH(X, i+1) & if X[i] < X[i+1] \\ SOSHIGH(X, i+1) + 1 & otherwise \end{cases}$$

$$SOSLOW(X,i) = \begin{cases} 0 & if\ i = len(X) - 1 \\ SOSLOW(X, i+1) & if X[i] > X[i+1] \\ SOSLOW(X, i+1) + 1 & otherwise \end{cases}$$

$$SOS(X) = min(SOSLOW(X), 1 + SOSHIGH(X))$$

(f)

$$LXS(X,j,i,k) = \begin{cases} 0 & if\ k > |X|\ or\ i > k\ or\ j > i \\ max(1 + LXS(X, j+1, i+1, k+1), LXS(X, j, i+1, k), LXS(X, j+1, i, k), & \\ LXS(X, j, i, k+1)) & if\ 2X[i] < X[j] + x[k] \\ max(LXS(X, j+1, i+1, k+1), LXS(X, j+1, i+1, k), & \\ LXS(X, j, i+1, k+1), LXS(X, j+1, i, k+1), LXS(X, j, i+1, k), & \\ LXS(X, j+1, i, k), LXS(X, j, i, k+1)) & otherwise \end{cases}$$

Question 2

(a)

```
def checkSubsequence(x, y, i):
    if len(y) == 0:
        return False
    if i == len(x):
        return True
    if x[i] == y[0]:
        return checkSubsequence(x, y[1:], i + 1) or checkSubsequence(x, y[1:], i)
    return checkSubsequence(x, y[1:], i)
```

(b)

```
def smallestSymbolsToBeRemoved(x, y):
    if len(y) < len(x):
        return 0
    elif len(x) == 0:
        return len(y)
    if not checkSubsequence(x, y, 0):
        return 0
    if x[0] == y[0]:
        return min(1 + smallestSymbolsToBeRemoved(x, y[1:]), smallestSymbolsToBeRemoved(x[1:], y))
    return smallestSymbolsToBeRemoved(x, y[1:])
```

(c)

```
def checkTwiceOccurrence(x, y, i1=0, i2=0):
    if len(y) == 0:
        return False
    if i1 == len(x):
        return checkSubsequence(x, y, i2)
    if i2 == len(x):
        return checkSubsequence(x, y, i1)
    if x[i1] == y[0] and x[i2] == y[0]:
        a <- checkTwiceOccurrence(x, y[1:], i1 + 1, i2)
        b <- checkTwiceOccurrence(x, y[1:], i1, i2 + 1)
        c <- checkTwiceOccurrence(x, y[1:], i1, i2)
        return a or b or c
    elif x[i1] == y[0]:
        return checkTwiceOccurrence(x, y[1:], i1 + 1, i2) or checkTwiceOccurrence(x, y[1:], i1, i2)
    elif x[i2] == y[0]:
        return checkTwiceOccurrence(x, y[1:], i1, i2 + 1) or checkTwiceOccurrence(x, y[1:], i1, i2)
    return checkTwiceOccurrence(x, y[1:], i1, i2)
```

Question 3

(a)

```
def LSCS(x):
    currentMax = -math.inf
    maxSoFar = 0
    for i in x:
        currentMax <- currentMax + i
        maxSoFar <- max(currentMax, maxSoFar)
        currentMax <- max(currentMax, 0)
    return maxSoFar
```

the algorithm loops on the array once and does $\mathcal{O}(1)$ work thus the algorithm has a runtime of $\mathcal{O}(n)$

(b)

```
def prod(x):
    mostNegative = x[0]
    mostPositive = x[0]
    current = x[0]
    for int i= 0, i < len(x), i++:
        if x[i] < 0:
            mostPositive, mostNegative = mostNegative, mostPositive
        mostPositive = max(x[i], mostPositive * x[i])
        mostNegative = min(x[i], mostNegative * x[i])
        current = max(current, mostNegative, mostPositive)
    return current
```

the algorithm loops on the array once and does $\mathcal{O}(1)$ work thus the algorithm has a runtime of $\mathcal{O}(n)$