Sunday, February 27, 2022

Assignment 4

CSCI 411

1.

a. Given A = [5, 8, 8, 3, 4, 1, 7, –3, 2, 9, 12],
A sequence is called Bitonic if it is first increasing, then decreasing.
Hence, the longest Bitonic sequence of A should be:

[8, 3, 1, -3, 2, 9, 1]

b. A sequence is called Bitonic if it is first increasing, then decreasing.
For example,
S = [1, 11, 2, 10, 4, 5, 2, 1]

Skip ===> 11, 5

[——>i——->d]
LBS = [1, 2, 10, 4, 2, 1]

A sequence, sorted in increasing order is considered Bitonic with the decreasing part as empty. Similarly, decreasing order sequence is considered Bitonic with the increasing part as empty. Using two subsequences arrays, the Longest Increasing Subsequence(LIS) and the Longest Decreasing Subsequence(LDS). The LIS will hold the length of increasing subsequence ending with array[i]. The LDS will store the length of decreasing subsequence starting from array[i]. Using these two arrays, we can get the length of longest bitonic subsequence.

c.

```
LBS(a):
    inc_sub_seq.size = a.size
    For i = 1…… a.size -1:
        For j = 0…….i - 1:
            If a[i] > a[j] and inc_sub_seq[i] < inc_sub_sum[j] + 1:
                inc_sub_sum[i] = inc_sub_sum[j] + 1

    dec_sub_seq.size = a.size

    For i = a.size - 2……0:
        For j = a.size - 1……i + 1:
            If a[i] > a[j] and dec_sub_seq[i] < dec_sub_sum[j] + 1:
                dec_sub_seq[i] = dec_sub_seq[j] + 1

    M = inc_sub_seq[0] + dec_sub_seq[0] - 1
    For i = 1….. a.size:
            If inc_sub_seq[i] + dec_sub_seq[j] - 1 > M:
                M = inc_sub_seq[i] + dec_sub_seq[i] - 1

    Return M
```

d. Time complexity is O(n^2), where n is the size of A because we are looping twice for the calculation of LIS and LDS.

2.

a. “Exponential” and “Polynomial”

```
0 1 2 3 4 5 6 7 8 9 10 11
1 1 2 2 3 4 5 6 7 8  9 10
2 2 2 3 2 3 4 5 6 7  8  9
3 3 3 3 3 3 4 5 6 7  8  8
4 4 4 4 4 4 4 5 6 7  8 10
5 5 5 5 5 4 5 4 5 6  7  8
6 6 6 6 5 5 5 5 5 6  7  8
```

7 7 7 7 6 6 6 6 6 6 7 8
8 8 8 8 7 7 7 7 7 6 7 8
9 9 9 9 8 8 8 8 8 7 6 7
10 10 10 10 9 9 9 9 9 8 7 6

6 is the answer!

b. Given A = “cat" and B = “cut”,
ins = 3, sub = 2, del = 6

replacing ‘a’ with ‘u’
cost = 2

Given A = “sunday" and B = “saturday”,
ins = 3, sub = 2, del = 6

Convert ‘un’ with ‘atur’
Replace ’n’ with ‘r’
Insert ’t’
Insert ‘a’
cost = 2 + 3 + 3 = 8

c.

```
editDistance(A, B, ins, del, sub):
    buffer[A.size][B.size]
    buffer[0][0].score = 0
    buffer[0][0].act = “match”

    For i=1……A.size:
        buffer[0][i].act = “delete”
        buffer[0][i].score = buffer[i-1][0].score + del

    For i=1……B.size:
        buffer[I][0].act = “insert”
        buffer[i][0].score = buffer[0][i-1].score + ins

    For i=1……A.size:
        For j=1……A.size:
```

```
            If (a[i-1] != b[j-1]:
                buffer[i][j].act = min_cost()
                buffer[i][j].score = min_cost()

    Result->score = buffer[A.size][B.size].score

    While A.size > 0 || B.size > 0:
        If “insert”:
            Replace with ‘_’
            A.size—
        Else If “delete”:
            Replace with ‘_’
            B.size—
        Else:
            Replace with ‘_’
            A.size—
            B.size—

    Result->a = modified_a
    Result->b = modified_b

    Return Result
```

d. Time complexity of this algorithm is O(m X n) where m and n are the sizes of string A and B respectively.