Scribed Notes 13 (11th October)

Subsequence

A subsequence of a given sequence is a sequence that can be derived from the given sequence by deleting some or no elements without changing the order of the remaining elements.

It is different from subarrary or substring which are contiguous, but in subsequence we can jump/skip elements without changing the order.

Increasing/decreasing subsequence in a list of distinct integers

Here we are finding the longest, either increasing or decreasing, monotonic subsequence.

Example 1,

13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Longest increasing subsequence = 1

Longest decreasing subsequence = 13.

The longest increasing/decreasing subsequence for n number distinct integers will always be greater than 0. So even for extreme cases where the elements are in decreasing order the length of increasing is 1 as if you pick any element it would be a subset of the list and there would be nothing to compare this element with so it's increasing. This is also called as **vacuously** true which means the case does not arise where the length will be zero.

Problem Statement

If you have (n^2 + 1) length sequence of distinct integers, there must be **either** a decreasing subsequence or an increasing subsequence of length n.

For n = 5, $n^2 + 1 = 26$,

There must be either increasing subsequence of 5 integers or decreasing subsequence of 5 integers.

Proof for this problem is given by the Pigeon Hole Principle.

(L_i, L_d)

L_i => Longest Increasing Subsequence

L_d => Longest Decreasing Subsequence

Example 2,

13, 4, 3, 5, 11, 9, 7, 8, 6, 2, 1, 10, 12

Approach 1 :

We take one element at a time and find L_i and L_d for that element, then add one element to the existing list and based on the new list find L_i and L_d of the new element out of which only 1 will increase not both.

So as per our example 13, 4, 3, 5, 11, 9, 7, 8, 6, 2, 1, 10, 12 First take 13

	A
1	(1,1)
2	13

As we saw earlier 1 element can be called increasing and decreasing as there is nothing to compare with, L_i and L_d will be (1, 1).

Now we create a new subsequence by adding one element which is 4 so new list is as follows:-

1	(1,1)	
2	13	
3	(1,1)	(1,2)
4	13	4

For 4 length is (1, 2) as value is less than 13 so Ld will increase by 1 but not Li.

Now the problem arises when you add 9 to the list.

	A	B	C	D	E	F
1	(1,1)					
2	13					
3	(1,1)	(1,2)				
4	13	4				
5	(1,1)	(1,2)	(1,3)	(2,3)	(3,3)	(3,3)
6	13	4	3	5	11	9
7						

For 9 L_i or L_d is neither increasing nor decreasing but as it negates our assumption so this approach cannot be used.

Approach 2 :

In this approach we are using Dynamic Programming Table and Recursive Formulation.

Example 3

8, 11, 9, 1, 4, 6, 12, 10, 5, 7

Sequence contains 10 terms. So that $10 = 3^2 + 1$. (here n =3)

There are four strictly increasing subsequences of length four :

1) 1, 4, 6, 12

2) 1, 4, 6, 7

3) 1, 4, 6, 10

4) 1, 4, 5, 7

There is also a strictly decreasing subsequence of length four :

1) 11, 9, 6, 5.

Longest Common Subsequence (LCS) problem :

Define LCS[n,m] to be the length of the longest common subsequence of S1[1..n] and S2[1..m].

Allow for 0 as an index, so LCS[0,k] = 0 and LCS[k,0]=0, to indicate that the null part of S1 or S2 has no match with the other.

Then we can define LCS[n,m] in the general case as follows:

1. If S1[n] = S2[m], then LCS[n,m] = 1 + LCS [n-1,m-1] (we can add this match)

2. If S1[n] ≠ S2[m], then LCS[n,m] = max{ LCS[n-1,m], LCS[n,m-1]} (we have no match here)