LONGEST INCREASING SUBSEQUENCE

Algorithm Design Course

TA Sessions

By S. Daneshvar

Longest increasing subsequence

Find the length of the longest subsequence of a given array such that all elements of the array are sorted in an increasing order.

For instance:

Input: {10, 22, 9, 33, 21, 50, 41, 60}

Subsequences: {10}, {22}, … , {10, 9, 33} ,{33, 21 , 50, 60}, etc.

Increasing Subsequence: {10}, {22}, … , {33, 50, 60} , … ,{ 10, 22, 33, 50 , 60} ,etc.

Longest Increasing Subsequence: {10,22,33,50,60} => Length(LIS) = 5

Longest increasing subsequence

Brute Force vs Dynamic Programming?

Longest increasing subsequence

Dynamic Programming Approach:

Iterator								
Array	10	22	9	33	21	50	41	60
LIS	1	1	1	1	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 1:

Iterator	j	i						
Array	10	22	9	33	21	50	41	60
LIS	1	1	1	1	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 1:

Iterator	j	i						
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	1	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 2:

Iterator	j		i					
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	1	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 3:

Iterator	j			i				
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	1	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 3:

Iterator		j		i				
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	2	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 3:

Iterator			j	i				
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 4:

Iterator	j				i			
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	1	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 4:

Iterator		j			i			
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 4:

Iterator			j		i			
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 4:

Iterator				j	i			
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 5:

Iterator	j					i		
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	1	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 5:

Iterator		j				i		
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	2	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 5:

Iterator			j			i		
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	3	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 5:

Iterator				j		i		
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	3	1	1

Longest increasing subsequence

Dynamic Programming Approach:

For i = 5:

Iterator					j	i		
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	4	1	1

Longest increasing subsequence

Dynamic Programming Approach:

Final Result:

Iterator							j	i
Array	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	4	4	5

Longest increasing subsequence

```
Integer LongestIncreasingSubsequence( Integer[] array ) {
        Integer[] lis = new Integer[array.length];
        for ( int i = 0 ; i < array.length ; i++)
                lis[i] = 1;

        for ( int i = 0 ; i < array.length ; i++)
                for ( int j = 0; j < i ; j++)
                        if( array[j] < array [i] && lis[i] < lis[j] + 1)
                                lis[i] = lis[j] + 1
        return max( lis ) // O(n)
}
```