

Samar Ibrahim Antar
Dynamic Programming for Longest Increasing Subsequence

- **LIS() function:** to find max length
- ✓ Create a 1D array `lis[]` of size `N`.
- ✓ `lis[] = {1, 1, 1, 1,...}` (initially).
- ✓ For each element, iterate elements with indexes lesser than current element in a nested loop, for example, `i >> start from the second array element & j >> do for each element in subarray `arr[0...i-1]``.
- ✓ In the nested loop, if the element's value is less than the current element `>> (arr[i]>arr[j])`, assign `lis[i]` with `(lis[j]+1)` if `(lis[j]+1)` is greater than `lis[i]` `>>(max(lis[i],lis[j]+1))`
- ✓ Traverse the entire `lis[]` array to extract the maximum element which will be our answer (max length).

- **Complexity Analysis:**
- ✓ For each element, we traverse all elements on the left of it.
- ✓ So, Time Complexity: $O(N^2)$.
- ✓ Space Complexity: $O(N)$, for storing the auxiliary array.

- **find_lis() function:** to print LIS
- ✓ Create a 1D array `subSequence[]` of size `[max length]`.
- ✓ `size_subSequence` is equal to max length.
- ✓ This function takes `>> lis[]=Table[]={1, 1, 2 ,2,3,1}` which we get from `LIS()` function to get max length from it, `array[] (sequence) = { 3, 2, 6, 4, 5, 1 }`; & size of array.
- ✓ Then we make iteration starting from `>> i = max index that is index of max length in lis[] >> (int i = max index; i >= 0; i--)`, `max index in this case= 4 and max length=3`.

✓ At 1st iteration:

- max length = Table[i] = Table[4]=3
- check if max length == [Table[max index] - increment]
3 == [3 -0] >> so it gives us true
- make max length=i=4
- then do this >>
subSequence[Table[max length]-1] = Sequence[max length]
subSequence[Table[4]-1] = Sequence[4];
subSequence[3-1] = Sequence[4];
So, subSequence[2]= 5 & increment=1

✓ At 2nd iteration:

- max length = Table[i] = Table[3]=2
- check if max length == [Table[max index] - increment]
2 == [3-1] >> true
- make max length=i=3
- then do this>>
subSequence[Table[max length]-1] = Sequence[max length]
subSequence[Table[3]-1] = Sequence[3];
subSequence[2-1] = 4
So, subSequence[1]= 4 & increment=2

✓ At 3rd iteration:

- max length = Table[i] = Table[2]=2
- check >> 2 ==[3-2] >> false
- increment >> remains equal to 2

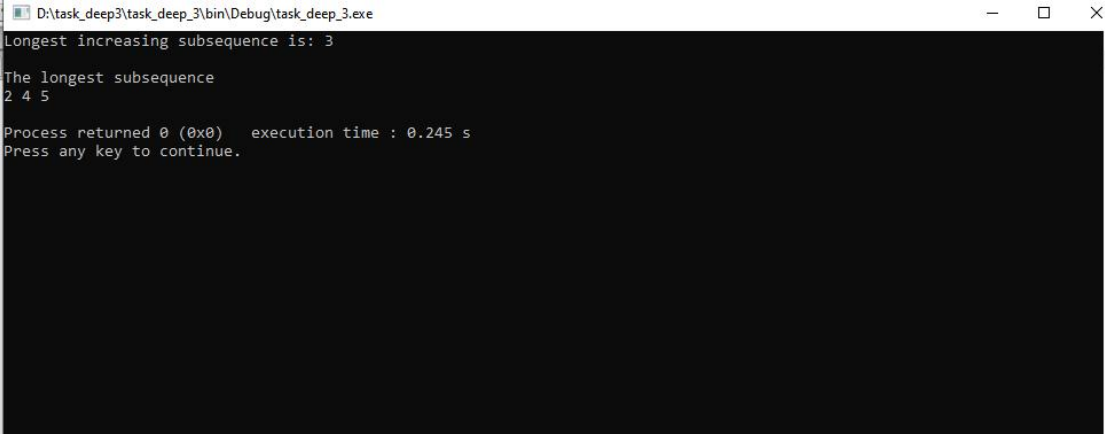
✓ 4th iteration:

- max length= Table[i] = Table[1]=1
- check >> 1==[3-2] >> true
- make max length=i=1
- subSequence[Table[max length]-1] = Sequence[max length]
subSequence[Table[1]-1] = Sequence[1];
subSequence[1-1] = 2
So, subSequence[0]= 2 & increment=3

✓ 5th iteration:

- max length= Table[i] = Table[0]=1
- check >> 1==[3-3] >> false

✓ Thus, longest subsequence is {2, 4, 5}



```
D:\task_deep3\task_deep_3\bin\Debug\task_deep_3.exe
Longest increasing subsequence is: 3
The longest subsequence
2 4 5
Process returned 0 (0x0) execution time : 0.245 s
Press any key to continue.
```