**NOTES**

1. LIS

|  |  |  |  |
| --- | --- | --- | --- |
| Generate all subsequences and then (2^n). | Recursion (take, not takerule but one pattern to take previous element) | DP(O(N^2)) | BS(N LogN)  Lower bound- if found then return first index of matching.  If all elements are greater than returns first index of element.  If all are smaller then returns last (not last index of element) |

1 5 **4** 2 6- it tells 4 ko include karke yahan tak ka sbse longest.

Dp[i]=max(dp[i],dp[j]+1)-ye ho skta hai(par jaruri nhi) but not required in case of longest string chain and largest divisible subset as we required subsets.

Lower bound- 30 40 agar 35 dhundhana hai to it will return index of 40.

1. Maximum sum increasing subsequence

|  |  |  |
| --- | --- | --- |
| Generate all subsequences and then (2^n). | DP(O(N^2)) | BinaryIndexed Tree not do these two are sufficient. |

1. Longest Bitonic subsequence

|  |  |
| --- | --- |
| Dp(O(N^2)) | O(NLOGN) USING BS AND STORING AS WELL |

1. Largest Divisible Subset

|  |  |
| --- | --- |
| Generate all subsets and check then. | Sort first then o(N^2)) DP solution. |

1. Number of LIS

|  |  |
| --- | --- |
| GENERATE ALL THEN MAP LGAO | O(N^2)- LIS + MAINTAINING COUNT. |

1. Longest string chain – [O(N^2)\*L(for checking) solution].