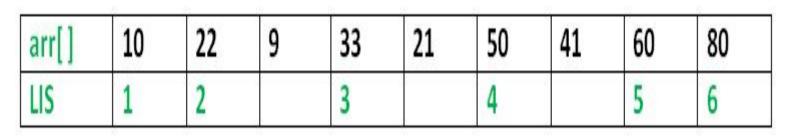
LIS DP

Problem Statement

The Longest Increasing Subsequence (LIS) problem is to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order

Input: arr[] = {3, 10, 2, 1, 20} Output: Length of LIS = 3 The longest increasing subsequence is 3, 10, 20

- Given Array {10, 22, 9, 33, 21, 50, 41, 60, 80}
- Then LIS length is 6
- Since LIS Sequence {10, 22, 33, 50, 60, 80}
- Approach : For calculating LIS till ith index we will check all the number from 1 to (i-1)th where is smaller than arr[i] and then add 1 to LIS till that index.
- What is meaning of dp[i] in our approach??.
- Time Complexity ??



Code of LIS

void solve()

```
ll n,maxi=0;
cin>>n;
ll arr[n+1],dp[n+1];
for(int i=0;i<n;i++)</pre>
Ł
    cin>>arr[i];
    dp[i]=1;
}
for(int i=0;i<n;i++)</pre>
{
    for(int j=0;j<i;j++)</pre>
     {
         if(arr[i]>arr[j])
              dp[i]=max(dp[i],dp[j]+1);
     }
for(int i=0;i<n;i++)</pre>
    maxi=max(maxi,dp[i]);
cout<<maxi<<"\n";</pre>
```

{

Better Approach

- Approach discuss is 0(n^2). We can optimize it to O(nlog(n))
- Go through these link
 - <u>CPAlgo_LIS</u>
 - Video tutorial <u>Tushar Roy LIS</u>