Maximum Sum Increasing Subsequence | DP-14

Given an array of n positive integers. Write a program to find the sum of maximum sum subsequence of the given array such that the integers in the subsequence are sorted in increasing order. For example, if input is {1, 101, 2, 3, 100, 4, 5}, then output should be 106 (1 + 2 + 3 + 100), if the input array is {3, 4, 5, 10}, then output should be 22 (3 + 4 + 5 + 10) and if the input array is {10, 5, 4, 3}, then output should be 10

Solution: This problem is a variation of the standard Longest Increasing Subsequence (LIS) problem. We need a slight change in the Dynamic Programming solution of LIS problem. All we need to change is to use sum as a criteria instead of a length of increasing subsequence.
Following are the Dynamic Programming solution to the problem :

C++

```
/* Dynamic Programming implementation
of Maximum Sum Increasing Subsequence
(MSIS) problem */
#include <bits/stdc++.h>
using namespace std;

/* maxSumIS() returns the maximum
sum of increasing subsequence
in arr[] of size n */
int maxSumIS(int arr[], int n)
{
    int i, j, max = 0;
    int msis[n];

    /* Initialize msis values
```

```
    for all indexes */
    for ( i = 0; i < n; i++ )
        msis[i] = arr[i];

    /* Compute maximum sum values
    in bottom up manner */
    for ( i = 1; i < n; i++ )
        for ( j = 0; j < i; j++ )
            if (arr[i] > arr[j] &&
                msis[i] < msis[j] + arr[i])
                msis[i] = msis[j] + arr[i];

    /* Pick maximum of
    all msis values */
    for ( i = 0; i < n; i++ )
        if ( max < msis[i] )
            max = msis[i];

    return max;
}

// Driver Code
int main()
{
    int arr[] = {1, 101, 2, 3, 100, 4, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Sum of maximum sum increasing "
```

```
            "subsequence is " << maxSumIS( arr, n ) << endl;
    return 0;
}
```

```
Sum of maximum sum increasing subsequence is 106
```

Time Complexity: O(n^2)
Space Complexity O(n)