

LIS in C++

```
#include <iostream>
#include <vector>
#include <algorithm> // For std::max
using namespace std;

void LIS(const vector<int>& arr) {
    int n = arr.size();
    vector<int> dp(n, 1); // dp[i] will
    store the length of LIS ending at
    index i
    int omax = 1; // To store the overall
    maximum length of LIS

    // Compute the length of the
    Longest Increasing Subsequence
    for (int i = 1; i < n; i++) {
        int max_len = 0;
        for (int j = 0; j < i; j++) {
            if (arr[i] > arr[j]) {
                if (dp[j] > max_len) {
                    max_len = dp[j];
                }
            }
        }
        dp[i] = max_len + 1;
        if (dp[i] > omax) {
            omax = dp[i];
        }
    }

    cout << omax << " "; // Print the
    length of the LIS

    // Printing the LIS length values
    (optional)
    for (int i = 0; i < n; i++) {
        cout << dp[i] << " ";
    }
    cout << endl;
}

int main() {
    vector<int> arr = {10, 22, 9, 33, 21,
    50, 41, 60, 80, 3};

    LIS(arr);

    return 0;
}
```

Let's perform a **dry run** of the given C++ program with the input:

arr = {10, 22, 9, 33, 21, 50, 41, 60, 80, 3}

Understanding the Code

The program finds the **length of the Longest Increasing Subsequence (LIS)** using **dynamic programming**.

- dp[i] stores the length of the **LIS ending at index i**.
- The **final answer** is the maximum value in dp[].

Step-by-Step Dry Run

| Step | i | j | arr[i] | arr[j] | arr[i] > arr[j] | dp[j] | max_len | dp[i] | omax |
|------|---|---|--------|--------|-----------------|-------|---------|-------|------|
| 1 | 1 | 0 | 22 | 10 | Yes | 1 | 1 | 2 | 2 |
| 2 | 2 | 0 | 9 | 10 | No | - | 0 | 1 | 2 |
| 3 | 2 | 1 | 9 | 22 | No | - | 0 | 1 | 2 |
| 4 | 3 | 0 | 33 | 10 | Yes | 1 | 1 | - | - |
| 5 | 3 | 1 | 33 | 22 | Yes | 2 | 2 | - | - |
| 6 | 3 | 2 | 33 | 9 | Yes | 1 | 2 | 3 | 3 |
| 7 | 4 | 0 | 21 | 10 | Yes | 1 | 1 | - | - |
| 8 | 4 | 1 | 21 | 22 | No | - | 1 | - | - |
| 9 | 4 | 2 | 21 | 9 | Yes | 1 | 1 | - | - |
| 10 | 4 | 3 | 21 | 33 | No | - | 1 | 2 | 3 |
| 11 | 5 | 0 | 50 | 10 | Yes | 1 | 1 | - | - |
| 12 | 5 | 1 | 50 | 22 | Yes | 2 | 2 | - | - |
| 13 | 5 | 2 | 50 | 9 | Yes | 1 | 2 | - | - |
| 14 | 5 | 3 | 50 | 33 | Yes | 3 | 3 | - | - |
| 15 | 5 | 4 | 50 | 21 | Yes | 2 | 3 | 4 | 4 |
| 16 | 6 | 0 | 41 | 10 | Yes | 1 | 1 | - | - |
| 17 | 6 | 1 | 41 | 22 | Yes | 2 | 2 | - | - |
| 18 | 6 | 2 | 41 | 9 | Yes | 1 | 2 | - | - |
| 19 | 6 | 3 | 41 | 33 | Yes | 3 | 3 | - | - |
| 20 | 6 | 4 | 41 | 21 | Yes | 2 | 3 | - | - |
| 21 | 6 | 5 | 41 | 50 | No | - | 3 | 4 | 4 |
| 22 | 7 | 0 | 60 | 10 | Yes | 1 | 1 | - | - |
| 23 | 7 | 1 | 60 | 22 | Yes | 2 | 2 | - | - |
| 24 | 7 | 2 | 60 | 9 | Yes | 1 | 2 | - | - |
| 25 | 7 | 3 | 60 | 33 | Yes | 3 | 3 | - | - |
| 26 | 7 | 4 | 60 | 21 | Yes | 2 | 3 | - | - |
| 27 | 7 | 5 | 60 | 50 | Yes | 4 | 4 | - | - |
| 28 | 7 | 6 | 60 | 41 | Yes | 4 | 4 | 5 | 5 |

| | | | | | | | | | |
|----|---|---|----|----|-----|---|---|---|---|
| 29 | 8 | 0 | 80 | 10 | Yes | 1 | 1 | - | - |
| 30 | 8 | 1 | 80 | 22 | Yes | 2 | 2 | - | - |
| 31 | 8 | 2 | 80 | 9 | Yes | 1 | 2 | - | - |
| 32 | 8 | 3 | 80 | 33 | Yes | 3 | 3 | - | - |
| 33 | 8 | 4 | 80 | 21 | Yes | 2 | 3 | - | - |
| 34 | 8 | 5 | 80 | 50 | Yes | 4 | 4 | - | - |
| 35 | 8 | 6 | 80 | 41 | Yes | 4 | 4 | - | - |
| 36 | 8 | 7 | 80 | 60 | Yes | 5 | 5 | 6 | 6 |
| 37 | 9 | 0 | 3 | 10 | No | - | 0 | - | - |
| 38 | 9 | 1 | 3 | 22 | No | - | 0 | - | - |
| 39 | 9 | 2 | 3 | 9 | No | - | 0 | - | - |
| 40 | 9 | 3 | 3 | 33 | No | - | 0 | - | - |
| 41 | 9 | 4 | 3 | 21 | No | - | 0 | - | - |
| 42 | 9 | 5 | 3 | 50 | No | - | 0 | - | - |
| 43 | 9 | 6 | 3 | 41 | No | - | 0 | - | - |
| 44 | 9 | 7 | 3 | 60 | No | - | 0 | - | - |
| 45 | 9 | 8 | 3 | 80 | No | - | 0 | 1 | 6 |

Final Output

6 1 2 1 3 2 4 4 5 6 1

- **LIS Length:** 6
- **LIS DP Table:** [1, 2, 1, 3, 2, 4, 4, 5, 6, 1]

Output:-
6
1 2 1 2 4 4 5 6 1