|  |  |
| --- | --- |
| **Print all LIS In C++** | |
| #include <iostream>  #include <vector>  #include <deque>  using namespace std;  struct Pair {      int l;    // length of the LIS      int i;    // index in the array      int v;    // value at index i in the array      string psf; // path so far      Pair(int l, int i, int v, string psf) {          this->l = l;          this->i = i;          this->v = v;          this->psf = psf;      }  };  void printAllLIS(vector<int>& arr) {      int n = arr.size();      vector<int> dp(n, 1); // dp array to store the length of LIS ending at each index      int omax = 0; // maximum length of LIS found      int omi = 0; // index where the LIS with maximum length ends      // Finding the length of LIS ending at each index      for (int i = 0; i < n; i++) {          int maxLen = 0;          for (int j = 0; j < i; j++) {              if (arr[i] > arr[j]) {                  if (dp[j] > maxLen) {                      maxLen = dp[j];                  }              }          }          dp[i] = maxLen + 1;          if (dp[i] > omax) {              omax = dp[i];              omi = i;          }      }      deque<Pair> q;      q.push\_back(Pair(omax, omi, arr[omi], to\_string(arr[omi])));      while (!q.empty()) {          Pair rem = q.front();          q.pop\_front();          if (rem.l == 1) {              cout << rem.psf << endl; // print the path when the length of LIS is 1          } else {              for (int j = rem.i - 1; j >= 0; j--) {                  if (dp[j] == rem.l - 1 && arr[j] <= rem.v) {                      q.push\_back(Pair(dp[j], j, arr[j], to\_string(arr[j]) + " -> " + rem.psf));                  }              }          }      }  }  int main() {      vector<int> arr = {10, 22, 9, 33, 21, 50, 41, 60, 80, 3};      printAllLIS(arr);      return 0;  } | **Dry Run Example******Input:**** vector<int> arr = {10, 22, 9, 33, 21, 50, 41, 60, 80, 3}; ****Step 1: Compute**** dp ****Array****  | **Index i** | **arr[i]** | **LIS Length (dp[i])** | **Previous LIS Contributor (dp[j])** | | --- | --- | --- | --- | | 0 | 10 | **1** | - | | 1 | 22 | **2** | 10 (dp[0] + 1) | | 2 | 9 | **1** | - | | 3 | 33 | **3** | 22 (dp[1] + 1) | | 4 | 21 | **2** | 10 (dp[0] + 1) | | 5 | 50 | **4** | 33 (dp[3] + 1) | | 6 | 41 | **4** | 33 (dp[3] + 1) | | 7 | 60 | **5** (Max LIS) | 50 (dp[5] + 1) | | 8 | 80 | **6** (Max LIS) | 60 (dp[7] + 1) | | 9 | 3 | **1** | - |  ****Step 2: Print All LIS Paths**** The **longest increasing subsequence has length 6** and ends at 80. Backtracking from 80, possible LIS paths:  10 -> 22 -> 33 -> 50 -> 60 -> 80  10 -> 22 -> 33 -> 41 -> 60 -> 80 |
| Output:- 10 -> 22 -> 33 -> 41 -> 60 -> 80  10 -> 22 -> 33 -> 50 -> 60 -> 80 | |