Maximum sum increasing subsequence from a prefix and a given element after prefix is must

> 给定数组arr，索引i与k，找到arr的和最大的递增子序列——该子序列必须为arr[0 ~ 索引i] + arr[索引k]组成的最长子序列，并返回该子序列的和

dp[i][j]表示arr[0 ~ i] + arr[j] 所组成的递增子序列的和，对于iIndex和kIndex，有：

1. iIndex > kIndex
2. iIndex < kIndex

i从1～N - 1，j从0 ～N - 1，这就隐含了iIndex与kIndex的大小关系了

对于dp[i][j]来说：

1. 如果 j > i 且 arr[j] > arr[i]
    1. 那么如果 dp[i - 1][i] + arr[j] > dp[i - 1][j]，则dp[i][j] = dp[i - 1][i] + arr[j] （意思是从arr[0 ~ i - 1] + arr[i] 组成的递增子序列再加上arr[j] 为当前的dp[i][j]）
    2. 否则，dp[i][j] = dp[i - 1][j]
2. 否则，dp[i][j] = dp[i - 1][j]

```
#include <bits/stdc++.h>

using namespace std;
int LIS(vector<int> &arr, int &N, int &iIndex, int &kIndex) {
  vector<vector<int>> dp(N, vector<int>(N));
  for (int i = 0; i < N; ++i) {
    arr[i] > arr[0] ? dp[0][i] = arr[i] + arr[0] : dp[0][i] = arr[i];
  }
  for (int i = 1;i < N; ++i) {
    for (int j = 0; j < N; ++j) {
      if (arr[j] > arr[i] && j > i) {
        if (dp[i - 1][i] + arr[j] > dp[i - 1][j])
```

```
          dp[i][j] = dp[i - 1][i] + arr[j];
        else
          dp[i][j] = dp[i - 1][j];
      } else
        dp[i][j] = dp[i - 1][j];
    }
  }
  return dp[iIndex][kIndex];
}
int main() {
  vector<int> arr = {1, 101, 2, 3, 100, 4, 5};
  int N = 7, iIndex = 4, kIndex = 6;
  printf("%d\n", LIS(arr, N, iIndex, kIndex));
  return 0;
}
```