2. Các cách tiếp cận khác nhau để giải quyết bài toàn LCS

2.1. Phương pháp qui hoạch động truyền thống

Đây là phương pháp tiếp cận cơ bản, được trình bày trong hầu hết các tài liệu tin học dánh cho học sinh. Nó là cơ bản bởi vì cách tiếp cận của nó thường được sử dụng cho các bài toán tương tự xử lý xâu (tính khoảng cách giữa các xâu ký tự, biến đổi xâu ký tự....)

Gọi $f[i,j]$ là độ dài của dãy con chung dài nhất của $a_1, a_2, \dots, a_i$ và $b_1, b_2, \dots, b_j$. Dễ thấy xảy ra các trường hợp sau:

- a_i không tham gia vào dãy con chung. Khi đó thực ra chỉ cần xét dãy $a_1, a_2, \dots, a_{i-1}$ và $b_1, b_2, \dots, b_j$. Giá trị độ dài dãy con chung dài nhất lúc này là $f[i-1,j]$
- b_j không tham gia vào dãy con chung. Bằng cách lập luận tương tự trong trường hợp này độ dài dãy con chung dài nhất là $f[i,j-1]$
- Cả a_i và b_j đều tham gia vào dãy con chung. Điều này chỉ xảy ra khi $a_i = b_j$. Phần còn lại chính là dãy con chung của $a_1, a_2, \dots, a_{i-1}$ và $b_1, b_2, \dots, b_{j-1}$. Do vậy độ dài dãy con chung dài nhất là $f[i-1,j-1]+1$

Vậy nên:

$$f[i,j] = \max\{f[i,j-1], f[i-1,j], f[i-1,j-1] + 1 * t(i,j)\} \qquad (*)$$

Trong đó $t(i,j) = \big(a_i == b_j\big)\ ?\ 1 : 0$.

Chú ý rằng khi $a_i = b_j$ thì $f[i-1,j-1]+1 \geq max\{f[i,j-1], f[i-1,j]\}$ nên trong trường hợp $a_i = b_j$ thì $f[i,j] = f[i-1,j-1]+1$.

Ngoài ra hiển nhiên $f[0,j] = f[i,0] = 0$

Giá trị $f[m,n]$ là độ dài dãy con chung dài nhất của hai dãy ban đầu $A = (a_1, a_2, \dots, a_m)$ và $B = (b_1, b_2, \dots, b_n)$

Đoạn code C++ dưới đây mô tả ý tưởng trên:

```
void LCS1() {

    // Tính mảng QHĐ
    for(int j=0;j<=n;j++) f[0][j]=0;
    for(int i=1;i<=m;i++) {
        f[i][0]=0;
        for(int j=1;j<=n;j++) {
            f[i][j]=max(f[i-1][j],f[i][j-1]);
```

```
            if (a[i]==b[j]) f[i][j]=f[i-1][j-1]+1;
        }
    }

    // Tìm một dãy con chung
    int ans=f[m][n];
    int u=m,v=n;
    int i=ans+1;
    while (u>0 && v>0) {
        if (a[u]==b[v]) {c[--i]=a[u];u--;v--;} else
        if (f[u][v]==f[u-1][v]) u--; else v--;
    }

    // In kết quả
    cout << ans << "\n";
    for(int i=1;i<=ans;i++) cout << c[i];
}
```

2.2. Phương pháp qui hoạch động gắn liền với điểm cuối

Ta gọi $f[i,j]$ = độ dài của xâu con chung dài nhất của $a_1, a_2, \dots, a_i$ và $b_1, b_2, \dots, b_j$ <u>thêm điều kiện giá trị phần tử cuối cùng phải là a_i</u>. Ta có hai trường hợp:

- b_j không tham gia vào dãy con chung. Khi đó độ dài là $f[i, j-1]$
- b_j tham gia vào dãy con chung. Điều này chỉ xảy ra khi $a_i = b_j$. Phần còn lại của dãy con chung dài nhất sẽ là dãy con chung của $a_1, a_2, \dots, a_x$ và $b_1, b_2, \dots, b_{j-1}$ với phần tử cuối là a_x $(0 \le x < i)$. Do vậy:

$$f[i,j] = \max\{f[x, j-1]: 0 \le x < i\} + 1$$

Tóm lại:

$$f[i,j] = \max\{f[i, j-1], \max\{f[x, j-1]: 0 \le x < i\} + 1\} \qquad (**)$$

Ngoài ra, hiển nhiên $f[0, j] = 0$

Nếu tính được mảng f thì độ dài dãy con chung dài nhất của $A = (a_1, a_2, \dots, a_m)$ và $B = (b_1, b_2, \dots, b_n)$ chính là $ans = \max\{f[i, n]: 1 \le i \le m\}$

Nếu như code thuần túy theo công thức (**) thì độ phức tạp thuật toán là $O(nm^2)$ (do phải sử dụng ba vòng lặp lồng nhau). Tuy nhiên nếu nhận xét rằng:

$\max\{f[x, j-1]: 0 \le x < i\} = \max\{f[x, j-1]: 0 \le x < i-1, f[i-1, j-1]\}$

thì vòng lặp tìm max có thể tính tích lũy. Do đó ta vẫn có thuật toán với thời gian $O(mn)$. Đoạn code C++ dưới đây mô tả ý tưởng trên:

```
void LCS2() {

    // Lập bảng QHĐ
    for(int j=1;j<=n;j++) {
        int Lmax=0, imax=0;
        for(int i=1;i<=m;i++) {
            f[i][j]=f[i][j-1]; prev[i][j]=prev[i][j-1];
            if (a[i]==b[j] && f[i][j]<Lmax+1) {
                f[i][j]=Lmax+1;
                prev[i][j]=imax;
            }
            if (f[i][j-1]>Lmax) Lmax=f[i][j-1],imax=i;
        }
    }

    // Tìm ngược
    int ans=0, u=0;
    for(int i=1;i<=m;i++) if (f[i][n]>ans) ans=f[i][n], u=i;
    int v=n;
    for(int i=ans;i>=1;i--) {
        c[i]=a[u];
        while (b[v]!=a[u]) v--;
        u=prev[u][v]; v--;
    }

    // In kết quả
    cout << ans << "\n";
    for(int i=1;i<=ans;i++) cout << c[i];
}
```

2.3. Phương pháp qui hoạch động ngược

Trong phương pháp này ta đặt $f[i,k]$ là vị trí j nhỏ nhất trong dãy $b_1, b_2, \dots, b_n$ để cho dãy con chung dài nhất giữa $a_1, a_2, \dots, a_i$ và $b_1, b_2, \dots, b_j$ bằng k. Ta có ha trường hợp:

- a_i không tham gia vào dãy con chung, khi đó vị trí bằng $f[i-1,k]$
- a_i tham gia vào dãy con chung. Phần còn lại có độ dài $k-1$. Vị trí đầu tiên xuất hiện xâu con chung độ dài $k-1$ với $a_1, a_2, \dots, a_{i-1}$trong $b_1, b_2, \dots, b_n$ là $u = f[i-1,k-1]$. Gọi $next(u,i) =$ vị trí đầu tiên xuất hiện a_i trong dãy

$b_u, b_{u+1}, \dots, b_n$ (nếu không xuất hiện đặt $next(u,i) = n+1$ ta có trong trường hợp này:

$$f[i,k] = next(f[i-1,k-1]+1, a_i)$$

Nếu hàm $next$ có thể chuẩn bị từ trước và khi đó ta có thuật toán $O(m^2)$ (không phụ thuộc n):

```
void LCS3() {

    // Lập bảng QHĐ
    for(int i=0;i<=m;i++)
        for(int k=0;k<=m;k++) f[i][k]=n+1;
    for(int i=0;i<=m;i++) f[i][0]=0;
    for(int i=1;i<=m;i++)
        for(int k=1;k<=i;k++)
        f[i][k]=min(f[i-1][k],next(f[i-1][k-1]+1,a[i]));

    // Tìm độ dài
    int ans=0, u=0;
    for(int k=1;k<=m; k++)
    for(int i=1;i<=m;i++)
    if (f[i][k]<n+1 && ans<k) ans=k, u=i;
    for(int v=ans;v>=1;v--) {
        while (f[u][v]==f[u-1][v]) u--;
        c[v]=a[u--];
    }
    // In kết quả
    cout << ans << "\n";
    for(int i=1;i<=ans;i++) cout << c[i];
}
```

Một điều thú vị là cách tiếp cận qui hoạch động ngược là một trong những cách tiếp cận thông minh giúp việc lập trình vượt qua được một số khó khăn trong trường hợp khó xây dựng được các công thức qui hoạch động thông thường. Ý tưởng chính của phương pháp có thể mô tả đơn giản như sau' thay vì tìm z=f(x,y) ta có thể qui về việc tìm y nếu biết z, x, tương tự tìm x nếu biết z, y.

2.4. Phương pháp cố định giá trị cuối

Trong cách tiếp cận này ta đặt $f[i,j]$ bằng độ dài của xâu con chung dài nhất kết thúc tại a_i và b_j. Do đó $fi,j] = 0$ nếu $a_i \neq b_j$. Công thức qui hoạch động có thể diễn tả

$$f[i,j] = \max\{f[x,y]: 1 \le x < i, 1 \le y < j, a_x = b_y\} + 1$$

Nếu $f[i,j] = f[x,y] + 1$ ta có các trường hợp sau:

- $f[x,y] = 0$: Khi đó a_i (hay b_j) là phần tử đầu tiên của dãy con chung
- $f[x,y] > 0$: Khi đó $a_x = b_y$

Nếu code đơn giản theo công thức trên, độ phức tạp thuật toán sẽ là $O(m^2n^2)$. Tuy nhiên có thể giảm độ phức tạp thuật toán xuống $O(mn)$ bằng nhận xét rằng với mỗi j thì trong các giá trị $f[1,j], f[2,j], \dots, f[i,j]$ (các giá trị trong cột j tính đến hàng i) ta chỉ cần lưu giá trị lớn nhất vì chỉ có giá trị này tham gia vào việc tính cho hàng sau/ Do đó ới mỗi j như trên ta chuẩn bị mảng $g[j]$ để lưu các giá trị max của f trên cột j. Giá trị $g[j]$ này sẽ được cập nhật lại sau khi tính xong 1 hàng.

Đoạn code C++ dưới đây mô tả ý tưởng trên:

```
void LCS4() {
    for(int j=0;j<=n;j++) g[j]=0;
    for(int i=1;i<=m;i++) {
        int Gmax=0;
        for(int j=1;j<=n;j++) {
            if (a[i]==b[j]) f[i][j]=Gmax+1; else f[i][j]=0;
            Gmax=max(Gmax,f[i-1][j]);
        }
        for(int j=1;j<=n;j++) g[j]=max(g[j],f[i][j]);
    }

    int ans=0;
    for(int i=1;i<=m;i++)
        for(int j=1;j<=n;j++) ans=max(ans,f[i][j]);
    cout << ans;
}
```

3. Một số bài tập áp dụng

Bài 1: Dãy con chung [VOSLIS.*]

Cho hai dãy số $A = (a_1, a_2, \dots, a_{m)}$ và $B = (b_1, b_2, \dots, b_{n)}$. Gọi $C = (c_1, c_2, \dots, c_k)$ là một dãy con chung của hai dãy trên (các phần tử không nhất thiết phải liên tiếp). Gọi giá trị của dãy con chung này là:

$$f(C) = |c_2 - c_1| + |c_3 - c_2| + \cdots + |c_k - c_{k-1}|$$

Nếu số lượng phần tử của dãy chung nhỏ hơn 2 thì đặt $f(C) = 0$

Hãy xác định dãy con chung có giá trị lớn nhất.

Input:

- Dòng 1 ghi hai số nguyên m, n
- Dòng 2 ghi m số nguyên biểu diễn dãy A
- Dòng 3 ghi n số nguyên biểu diễn dãy B

Output: Một dòng duy nhất ghi kết quả là giá trị của dãy chung tìm được

Example:

input	output
4 4 **1 15 8 7** **15 1 7 8**	**8** **(Dãy số chung là 15 7)**

Ghi chú:

- Subtask 1: $1 \le m, n \le 20$
- Subtask 2: $1 \le m, n \le 200$
- Subtask 3: $1 \le m, n \le 2000$
- Subtask 4: $1 \le m, n \le 5000$

Trong tất cả các test thì $|a_i|, |b_j| \le 10^9$

Thuật toán:

Ta sử dụng phương pháp thứ tư (qui hoạch động cố định điểm cuối).

Đặt $f[i,j]$ là giá trị lớn nhất của xâu con chung với tiền tố $a_1, a_2, \dots, a_i$ và tiền tố $b_1, b_2, \dots, b_j$ và giá trị của phần tử cuối cùng là $a_i = b_j$. Nếu $a_i \neq b_j$ thì đặt $f[i,j] = 0$.

Ta có công thức qui hoạch động:

$$f[i,j] = \max_{\substack{1 \le u < i \\ 1 \le v < j}} \{f[u,v] + |a_i - a_u|\}$$

Ở đây chú ý $a_u = b_v$.

Để có được thuật toán O(n^2) chúng ta làm như sau:

Đặt $g[j] = \max(f[u,j]: 1 \le u \le i)$. Mảng $g[\dots]$ sẽ được cập nhật khi tăng giá trị của i, Ta có vòng lặp sau:

```
for(int j=0;j<=n;j++) g[j]=-1;
```

```
for(int i=1;i<=m;i++) {
    lint Gmax=-1;
    for(int j=1;j<=n;j++) {
        if (a[i]!=b[j]) f[i][j]=-1; else
        f[i][j]=(Gmax==-1) ? 0: Gmax;
        if (g[j]>-1) Gmax=max(Gmax,g[j]+abs(a[i]-b[j]));
    }
    for(int j=1;j<=n;j++) if (a[i]==b[j])
g[j]=max(g[j],f[i][j]);
}
```

Dưới đây là chương trình C++11 của em Lê Mai An (thành viên đội tuyển VOI 2017, Hải Dương) mô tả ý tưởng trên:

```
#include <bits/stdc++.h>
#define maxn 5005

using namespace std;
typedef long long lint;

int m, n, a[maxn], b[maxn];
lint f[maxn][maxn], g[maxn];

int main() {
    freopen("voslis.inp","r",stdin);
    freopen("voslis.out","w",stdout);
    ios::sync_with_stdio(false);
    cin >> m >> n;
    for(int i=1;i<=m;i++) cin >> a[i];
    for(int j=1;j<=n;j++) cin >> b[j];
    for(int j=0;j<=n;j++) g[j]=-1;
    for(int i=1;i<=m;i++) {
        lint Gmax=-1;
        for(int j=1;j<=n;j++) {
            if (a[i]!=b[j]) f[i][j]=-1; else
            f[i][j]=(Gmax==-1) ? 0: Gmax;
            if (g[j]>-1)
            Gmax=max(Gmax,g[j]+abs(a[i]-b[j]));
        }
        for(int j=1;j<=n;j++)
        if (a[i]==b[j]) g[j]=max(g[j],f[i][j]);
    }
    lint ans=0;
    for(int i=1;i<=m;i++)
```

```
        for(int j=1;j<=n;j++) ans=max(ans,f[i][j]);
    cout << ans;
}
```

Bài 2: Dãy con chung dài nhất [LCS.*]

Cho hai dãy số $A = (a_1, a_2, \dots, a_m)$ và $B = (b_1, b_2, \dots, b_n)$. Dãy số $C = (c_1, c_2, \dots, c_k)$ được gọi là dãy con chung của A và B nếu tồn tại hai bộ chỉ số:

$$1 \le i_1 < i_2 < \cdots < i_k \le m$$
$$1 \le j_1 < j_2 < \cdots < j_k \le n$$

sao cho $c_p = a_{i_p} = b_{j_p}$ $(\forall\ p = 1,2, \dots, k)$

Yêu cầu: Tìm dãy số C là dãy con chung của A, B với độ dài lớn nhất có thể

Input:

- Dòng 1: chứa hai số nguyên dương m, n $(m \le 10^3; n \le 10^6)$
- Dòng 2: chứa m số nguyên $a_1, a_2, \dots, a_m$ $(\forall\ i: |a_i| \le 10^9)$
- Dòng 3: chứa n số nguyên $b_1, b_2, \dots, b_n$ $(\forall\ j: |b_j| \le 10^9)$

Output:

- Dòng 1: Độ dài dãy con chung tìm được (k)
- Dòng 2: Ghi các số $c_1, c_2, \dots, c_k$

Example:

input	output
9 9 1 2 7 3 4 8 5 6 9 1 2 3 4 5 6 7 8 9	7 1 2 3 4 5 6 9

<u>Thuật toán:</u>

Do $m \le 10^3, n \le 10^6$ nên nếu trực tiếp áp dụng phương pháp cổ điển (phương pháp 1) thì thuật toán không đảm bảo thời gian cho phép. Do vậy ta áp dụng phương pháp 3 (qui hoạch động đảo ngược). Chú ý là với cách tiếp cận này thì độ phức tạp chỉ còn phụ thuộc vào m mà không phụ thuộc vào n.

Đặt $f[i,k]$ là chỉ số nhỏ nhất trên B của dãy con chung độ dài k với phần tiền tố $a_1, a_2, \dots, a_i$. Ta có hai trường hợp:

TH1: Dãy con chung không chứa a_i. Khi đó $f[i,k] = f[i-1,k]$

TH2: Dãy con chung chứa a_i: Khi đó $f[i,k] = \text{next}(f[i-1, k-1], a_i)$. Trong đó hàm next xác định vị trí tiếp theo > f[i-1,k-1] trên mảng B có giá trị bằng a_i

Công thức $f[i,k] = \min(f[i-1,k], next(f[i-1,k-1], a_i))$

Tất nhiên $f[i,0] = 0$ và $f[i,k] = n+1$ nếu không tồn tại dãy độ dài k với i tiền tố của A.

Đáp số là giá trị k lớn nhất mà $f[i,k] < n+1$

Để truy vết ta sử dụng mảng $prv[i,k] = 0$ nếu bước này ta đi đến $(i-1,k)$ và bằng i nếu giá trị này là a_i (tiếp theo đi dến $(i-1,k-1)$)
Để thuật toán hiệu quả ta phải xây dựng được hàm Next(i,val): Tìm vị trí đầu tiên xuất hiện val trong dãy hậu tố $b_i, b_{i+1}, \dots, b_n$.
Đặt x[1]≤x[2]≤...≤x[m] là mảng giá trị nén của a. Xây dựng mảng vector
vector<int> g[maxM] trong đó g[u] là danh sách chỉ số của giá trị x[u] xuất hiện trong b[1],...,b[n] theo thứ tự tăng dần:

```
for(int i=1;i<=n;++i) {
    int u=lower_bound(x+1,x+m+1,b[i])-x;
    if (u<=m && x[u]==b[i]) { // b[i] có xuất hiện trong mảng
a
        g[u].push_back(i);
    }
}
```

Khi đã có mảng vector g (chuẩn bị ngay sau khi đọc dữ liệu), hàm Next(i,x) có thể viết:

```
int Next(int i,int val) {
    int u=lower_bound(x+1,x+m+1,val)-x;
    if (u<=m && x[u]==val) {
        vector<int>::t=iterator
q=upper_bound(g[u].begin(),g[u].end(),i)
        if (q!=g[u].end()) return *q;
    }
    return n+1;
}
```

Từ đây có thể viết đoạn code qui hoạch động đảo ngược như sau:

```
void LCS3() {
    // Lập bảng QHĐ
    for(int i=0;i<=m;i++)
        for(int k=0;k<=m;k++) f[i][k]=n+1;
    for(int i=0;i<=m;i++) f[i][0]=0;
    for(int i=1;i<=m;i++)
    for(int k=1;k<=i;k++) {
        int u=Next(f[i-1,k-1]+1,a[i]);
        f[i][k]=min(f[i-1][k],u);
    }
```

```
    // Tìm độ dài
    int ans=0, u=0;
    for(int k=1;k<=m;k++)
    for(int i=1;i<=n;i++) if (f[i][k]<n+1 && ans<k) ans=k,
u=i;
    for(int v=ans;v>=1;v--) {
        while (f[u][v]==f[u-1][v]) u--;
        c[v]=a[u--];
    }
 }
```

Chương trình dưới đây là bài code của em Vũ Duy Mạnh - thành viên đội tuyển VOI Hải Dương 2016 (ngôn ngữ C++11):

```
#include <bits/stdc++.h>
using namespace std;
int  n,m,a[1005],a1[1005],s[1000005],
f[1005][1005],_prev[1005][1005];
vector <int> g[1005];
int x[1005];
int main()
{
    freopen("lcs.inp","r",stdin);
    freopen("lcs.out","w",stdout);
    ios::sync_with_stdio(false);
    cin>>n>>m;
    for(int i=1;i<=n;i++){cin>>a[i];a1[i]=a[i];}
    for(int i=1;i<=m;i++)cin>>s[i];
    sort(a1+1,a1+n+1);
    int nP=1;
    for(int i=2;i<=n;i++)if(a1[i]!=a1[i-1])a1[++nP]=a1[i];
    for(int i=1;i<=n;i++)
    a[i]=lower_bound(a1+1,a1+nP+1,a[i])-a1;
    for(int i=1;i<=m;i++)
    {
        int q=lower_bound(a1+1,a1+nP+1,s[i])-a1;
        if(a1[q]==s[i])
        {
            g[q].push_back(i);
        }
    }
    for(int i=0;i<=n;i++)f[i][0]=0;
    for(int j=1;j<=n;j++)f[0][j]=-1;
```

```
    int ds=0;
    for(int j=1;j<=n;j++)for(int i=1;i<=n;i++)
    {
        f[i][j]=f[i-1][j];
        _prev[i][j]=_prev[i-1][j];
        if(f[i-1][j-1]!=-1)
        {
            __typeof((g[a[i]].begin()))
            q=upper_bound(g[a[i]].begin(),g[a[i]].end(),
            f[i-1][j-1]);
            if(q!=g[a[i]].end())
            {
                if(f[i][j]==-1){f[i][j]=*q;_prev[i][j]=i;}
                else if(f[i][j]>*q)
                {
                    f[i][j]=*q;
                    _prev[i][j]=i;
                }
                ds=j;
            }
        }
    }
    cout<<ds<<"\n";
    int u=n,v=ds;
    while(v)
    {
        x[v]=_prev[u][v];
        u=x[v]-1;
        v--;
    }
    for(int i=1;i<=ds;i++)cout<<a1[a[x[i]]]<<" ";
}
```

Bài 4: Dãy con chung tăng dài nhất [LCIS.*]

Một dãy số $a_1, a_2, ..., a_n$ được gọi là dãy tăng dần nếu như $a_i < a_{i+1}$ với mọi $i<n$.

Một dãy $s_1, s_2, ..., s_k$ được gọi là dãy con của dãy $a_1, a_2, ..., a_n$ nếu tồn tại một tập các chỉ số $1\le i_1<i_2<...<i_k\le n$ thỏa mãn $a_{i_j} = s_j$. Nói cách khác, dãy số s nhận được từ việc loại bỏ đi một số phần tử của dãy a.

Cho hai dãy số nguyên, bạn hãy tìm dãy con chung tăng dài nhất của chúng.

Input:

- Dòng đầu tiên ghi số nguyên N là số phần tử của dãy thứ nhất ($1\le N\le 5000$)

- Dòng thứ hai ghi N số nguyên thể hiện các phần tử trong dãy số thứ nhất. Các số nằm trong khoảng $[1,10^9]$.
- Dòng thứ ba ghi số nguyên M là số phần tử của dãy thứ hai ($1 \le M \le 5000$)
- Dòng thứ tư ghi M số nguyên thể hiện các phần tử trong dãy thứ hai. Các số nằm trong khoảng $[1,10^6]$.

Output: Một số nguyên duy nhất là số phần tử của dãy con chung tăng dài nhất tìm được

Example:

input	output
7 2 3 1 6 5 4 6 4 1 3 5 6	3
5 1 2 0 2 1 1 0 1	2

Thuật toán:

Đặt $f[i,j]$ là độ dài dãy con chung tăng dài nhất của $a_1, a_2, \dots, a_i$ và $b_1, b_2, \dots, b_j$ với giá trị phần tử cuối là $a_i = b_j$. Ta có $f[i,j] = 0$ nếu $a_i \neq b_j$ ngược lại:

$$f[i,j] = \max\{f[x,y] + 1 : a_x = b_y < a_i, 0 \le x < i, 0 \le y < j\}$$

(Phương pháp qui hoạch động thứ tư)

Để có được thuật toán $O(n^2)$ ta duy trì mảng $g[j] = \max\{f[x,j] : 0 \le x < i, b_j < a_i\}$ trong quá trình tính:

```
for(int j=0;j<=n;j++) g[j]=0;
for(int i=1;i<=m;i++) {
    int Lmax=0;
    for(int j=1;j<=n;j++) {
        if (a[i]!=b[j]) f[i][j]=0; else f[i][j]=Lmax+1;
        if (a[i]>b[j]) Lmax=max(Lmax,g[j]);
    }
    for(int j=1;j<=n;j++) g[j]=max(g[j],f[i][j]);
}
```

Chương trình C++11 dưới của em Trần Minh Hoàng (thành viên đội tuyển VOI 2018, Hải Dương:

```
#include <bits/stdc++.h>
#define maxn 5005

using namespace std;

int n, m, a[maxn], b[maxn];
int g[maxn], f[maxn][maxn];
```

```
int main() {
    freopen("lcis.inp","r",stdin);
    freopen("lcis.out","w",stdout);
    ios::sync_with_stdio(false);
    cin >> m;
    for(int i=1;i<=m;i++) cin >> a[i];
    cin >> n;
    for(int i=1;i<=n;i++) cin >> b[i];
    for(int j=0;j<=n;j++) g[j]=0;
    for(int i=1;i<=m;i++) {
        int Lmax=0;
        for(int j=1;j<=n;j++) {
            if (a[i]!=b[j]) f[i][j]=0; else
            f[i][j]=Lmax+1;
            if (a[i]>b[j]) Lmax=max(Lmax,g[j]);
        }
        for(int j=1;j<=n;j++) g[j]=max(g[j],f[i][j]);
    }
    int ans=0;
    for(int i=1;i<=m;i++)
        for(int j=1;j<=n;j++) ans=max(ans,f[i][j]);
    cout << ans;
}
```

Một trong những bài tương tự hoàn toàn có thể giải giống như bài trên là bài:
http://vn.spoj.com/problems/LCS2X/