Title: Favorite Color Stripe

Description:

Eva is trying to make her own color stripe out of a given one. She would like to keep only her favorite colors in her favorite order by cutting off those unwanted pieces and sewing the remaining parts together to form her favorite color stripe.

It is said that a normal human eye can distinguish about less than 200 different colors, so Eva’s favorite colors are limited. However the original stripe could be very long, and Eva would like to have the remaining favorite stripe with the maximum length. So she needs your help to find her the best result.

Note that the solution might not be unique, but you only have to tell her the maximum length. For example, given a stripe of colors {2 2 4 1 5 5 6 3 1 1 5 6}. If Eva’s favorite colors are given in her favorite order as {2 3 1 5 6}, then she has 4 possible best solutions {2 2 1 1 1 5 6}, {2 2 1 5 5 5 6}, {2 2 1 5 5 6 6}, and {2 2 3 1 1 5 6}.

Input Specification:

Each input file contains one test case. For each case, the first line contains a positive integer N ( N≤200) which is the total number of colors involved (and hence the colors are numbered from 1 to N). Then the next line starts with a positive integer M (M≤200) followed by MM Eva’s favorite color numbers given in her favorite order. Finally the third line starts with a positive integer L (L≤104​​ ) which is the length of the given stripe, followed by L colors on the stripe. All the numbers in a line a separated by a space.

Output Specification:

For each test case, simply print in a line the maximum length of Eva’s favorite stripe.

Sample Input:

6

1. 2 3 1 5 6

12 2 2 4 1 5 5 6 3 1 1 5 6

Sample Output:

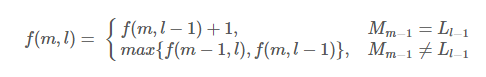
7

解法说明

设Eva喜欢的序列为M[0...m)，给出条纹的序列为L[0...l)。   
经过裁减，符合条件序列的最大长度（最大公共序列长度）的解为f(m,l)。   
对于状态空间 {(m,l)|m∈M,l∈L}，   
平凡态：

f(0,l)=f(m,0)=0

有状态转移方程：



平凡态比价容易理解：空序列的长度为0。

现在考虑当Eva喜欢的序列不变时，每当给出的条纹多了一个，即Ll−1，考虑它是否与Eva喜欢的序列的最后一个(Mm−1)相同，即它是否可以拓展，如果可以，它可以直接加到f(m,l−1)上，这个情况也比较好理解。

最后考虑当这个新来的条纹不匹配时，它，或者Eva喜欢的最后一个颜色，两者之一总有一个没有卵用的：当新来的条纹不存在(f(m,l−1))或者Eva喜欢的最后一个颜色不存在(f(m−1,l))，比较一下哪个比较大就取哪个即可。

http://blog.csdn.net/zccz14/article/details/51144598

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

这个问题的精简描述可以定义为：两个序列S,T。求出S,T的最大公共子序列。其中S中的子序列可以可以重复。

递归搜索版本：

#include <cstdio>

#include <algorithm>

const int MAXM = 200 + 5;

const int MAXL = 10000 + 5;

int dp[MAXM][MAXL];

int M[MAXM], L[MAXL];

// the solve of M[0...m) and L[0...l)

int solve(int m, int l){

if(!m || !l) return 0;

if(dp[m][l]) return dp[m][l];

if(M[m - 1] == L[l - 1]) return dp[m][l] = solve(m, l - 1) + 1;

return dp[m][l] = std::max(solve(m - 1, l), solve(m, l - 1));

}

// main I/O

int main(){

int Mc, Lc;

scanf("%\*d%d", &Mc);

for(int i = 0; i < Mc; i++)

scanf("%d", M + i);

scanf("%d", &Lc);

for(int i = 0; i < Lc; i++)

scanf("%d", L + i);

printf("%d\n", solve(Mc, Lc));

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Version 2：空间优化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <cstdio>

const int MAXM = 200 + 5;

const int MAXL = 10000 + 5;

int dp[MAXM], M[MAXM], L[MAXL];

int main(){

int Mc, Lc;

scanf("%\*d%d", &Mc);

for(int i = 0; i < Mc; i++)

scanf("%d", M + i);

scanf("%d", &Lc);

for(int i = 0; i < Lc; i++)

scanf("%d", L + i);

for(int l = 0; l < Lc; l++)

for(int m = 0; m < Mc; m++)

if(M[m] == L[l]) dp[m + 1]++;

else if(dp[m + 1] < dp[m]) dp[m + 1] = dp[m];

printf("%d\n", dp[Mc]);

return 0;

}

/\*\*\*\*\*\*\*\*\*Max Subsequence Sum DP sulutio**n**\*\*\*\*\*\*\*/

#include<vector>

#include<iostream>

#include<cstdio>

#include<algorithm>

#define MAX\_SIZE 10001

using namespace std;

typedef struct Node{

int left;

int right;

int value;

}Node;

bool cmp(const Node &a,const Node &b){

if(a.value!=b.value){

return a.value>b.value;

}

else{

if(a.left!=b.left) return a.left<b.left;

else return a.right<b.right;

}

}

int main(){

//freopen("D:\\in.txt","r",stdin);

int n;

scanf("%d",&n);

Node temp;

vector<Node> result\_vec;

vector<int> vec(n);

/\* dp stores the maximum subsequence sum that contants the last digit \*/

int dp[MAX\_SIZE];

for(int i=0;i<n;i++){

scanf("%d",&vec[i]);

}

dp[1]=vec[0];

temp.left=0;

temp.right=0;

temp.value=dp[1];

int cur\_left=0;

int right;

result\_vec.push\_back(temp);

for(int i=1;i<n;i++){

// dp[i+1]=max(dp[i]+vec[i],vec[i]);

if(dp[i]+vec[i]>vec[i]){

dp[i+1]=dp[i]+vec[i];

}

else{

cur\_left=i;

dp[i+1]=vec[i];

}

right=i;

temp.left=cur\_left;

temp.right=right;

temp.value=dp[i+1];

result\_vec.push\_back(temp);

}

sort(result\_vec.begin(),result\_vec.end(),cmp);

vector<Node>::iterator iter=result\_vec.begin();

if(iter->value<0) printf("%d %d %d\n",0,vec[0],vec[n-1]);

else

printf("%d %d %d\n",iter->value,vec[iter->left],vec[iter->right]);

return 0;

}

The dp[i] (I starts from 1) which stands for the in 1~I section the max subsequence sum which are continuous and must contain the last i-th emelent in the source array.

So dp[i+1]=max(dp[i]+vec[i],vec[i])

You must keep in mind that dp[n] is definitly not the final result we want, since the max subsequence sum is not necessarily contains the last element, but every dp[i] is the potentially answer we want, so just store them and find the largest value the last of the algorithm. #