CDQ分治 第二课

重庆南开信竞基础课程

CDQ与DP

NKOJ4429 拦截导弹[SDOI2011]

有n颗来袭导弹，每颗导弹的高度hi和速度vi已知，现只有一套导弹拦截火炮，其第一发炮弹可以拦截任意高度，任意速度的导弹，但之后拦截的每发导弹的高度和速度都不能超过前一颗，要**计算拦截最多的导弹**，当有多种方案时，随机选择一种，问**每一颗导弹被拦截的概率。**

$n \le 5\times10^4, hi \le 10^9, vi \le 10^9$

考虑第一问

令$F1[i]$表示以i结尾的最长下降子序列长度

令$F2[i]$表示以i开头的最长下降子序列长度

有$F1[i] = \max\{F1[j] \mid j \le i, h_j \ge h_i, v_j \ge v_i\} + 1$

且$F2[i] = \max\{F2[j] \mid j \ge i, h_j \le h_i, v_j \le v_i\} + 1$

做法一：树套树 $O(nlog^2n)$

做法二：CDQ分治，三维偏序

第一维是导弹编号，默认有序;

第二维是导弹高度，对高度hi进行分治;

第三维是导弹速度，对vi建立树状数组;

$O(nlog^2n)$

考虑第二问

令$G1[i]$表示以i结尾的长度为$F1[i]$的下降子序列个数

令$G2[i]$表示以i开头的长度为$F2[i]$的下降子序列个数

有$G1[i] = \sum G1[j], (j \le i, h_j \ge h_i, v_j \ge v_i, F1[j] + 1 = F1[i])$

且$G2[i] = \sum G2[j], (j \ge i, h_j \le h_i, v_j \le v_i, F2[j] + 1 = F2[i])$

令最长上升子序列长度为Len，令最长上升子序列总数为Sum

对于点i，当$F1[i] + F2[i] - 1 = Len$时，答案就是$\frac{G1[i]\times G2[i]}{Sum}$

进行CDQ分治时。每次将序列分成左和右两部分。

每次考虑左边对右边的影响。

递归处理左部;

更新当前节点的值;

递归求解右部。

NKOJ4599 序列[TJOI / HEOI2016]

题目描述

有一个数列，数列中某些项的值可能会变化，但**同一个时刻最多只有一个值发生变化。**
能否选出一个子序列，使得在任意一种变化中，这个子序列都是不降的？计算这个子序列的最长长度。

输入格式

第一行有两个正整数n和m，分别表示序列的长度和变化的个数。
接下来一行有n个数，表示这个数列原始的状态。
接下来m行，每行有两个数x，y，表示数列的第x项可以变化成y这个值。

输出格式

一行，一个整数，表示答案。

数据规模

对于所有的数据，1<=x<=n，所有数字均为正整数，且小于等于10^5。

NKOJ4599 序列[TJOI / HEOI2016]

一个数列，某些项的值会变化，**同一个时刻最多只有一个值发生变化**。选出一个最长子序列，任意一种变化中，这个子序列都是不降的。

动态规划：Dp[i]表示以第i项作为结尾的不降子序列的最长长度。
Dp[i] = max{ Dp[j] } + 1
条件：j<i 且 Max[j]<=Val[i] 且 Val[j]<=Min[i]
Max[j]表示第j项可能取到的最大值　Min[i]表示第i项可能取到的最小值　Val[i]表示第i项的原始值
因为同一时刻最多有一个值变化。所以要么j变为Max[j]，i不变仍为Val[i]。要么j不变仍为Val[j]，i变为Min[i]。

观察方程：**Dp[i] = max{ Dp[j] } + 1** 直接求解耗时$O(n^2)$
条件1：j<i　条件2：Max[j]<=Val[i]　条件3：Val[j]<=Min[i]
构成三维偏序，考虑CDQ分治优化DP。条件1默认有序，条件2用CDQ维护，条件3用树状数组维护

CDQ处理：
对于区间[L,R]，也就是计算Dp[L],Dp[L+1],…,Dp[R]
将区间[L,R]分为左右两部分[L,Mid]和[Mid+1,R]
j为左部[L,Mid]中的某一项
i为右部[Mid+1,R]中的某一项

默认有j<i
根据方程，Max[j]<=Val[i]才有可能更新Dp[i]，所以左部用Max[]为关键字排序，右部用Val[]为关键字进行排序
用左部的Dp[]去更新右部的Dp[]
若Max[j]<=Val[i]，则将Dp[j]添加到树状数组的Val[j]处（树状数组维护前缀最大值）

NKOJ4599 序列[TJOI / HEOI2016]

一个数列，某些项的值会变化，**同一个时刻最多只有一个值发生变化**。选出一个最长子序列，任意一种变化中，这个子序列都是不降的。

Dp[i] = max{ Dp[j] } + 1

条件1：j<i　　条件2：Max[j]<=Val[i]　　条件3：Val[j]<=Min[i]

条件1默认有序　条件2用CDQ维护，　　条件3用树状数组维护

```
// A[i].Max  数列第i项可能的最大值
// A[i].Min  数列第i项可能的最小值
// A[i].Val  数列第i项的原始值
// A[i].Pos  数列第i项的原始下标
// A[i].Dp   以数列第i项作为结尾的最长不降子序列长度，初始值为1
void CDQ(int L,int R)
{
    if (L==R) return;
    int Mid=L+R>>1;
    CDQ(L,Mid);
    sort(A+L,A+Mid+1,cmp1);     //[L,Mid]按A[].Max排序
    sort(A+Mid+1,A+R+1,cmp2);   //[Mid+1,R]按A[].Val排序
    int j=L,i=Mid+1;
    while(i<=R)
    {
       while(j<=Mid&&A[j].Max<=A[i].Val)Modify(A[j].Val,A[j].Dp),j++;
       A[i].Dp=max(A[i].Dp,GetMax(A[i].Min)+1); i++;
    }
    for(int k=L;k<=j;k++)Modify(A[k].Val,0); //清除树状数组被修改的部分
    sort(A+Mid+1,A+R+1,cmp3); //[Mid+1,R]按A[].Pos排序
    CDQ(Mid+1,R);
}
```

//主函数调用CDQ(1,n)即可

问题1：为何会有第三次排序（代码中紫色部分）？

问题2：为何是下面步骤？

首先：CDQ(L,Mid)

然后：用左部去更新右部的Dp

最后：CDQ(Mid+1,R)

NKOJ3208 奶牛跳格子

奶牛们参加跳格子游戏。
游戏在一个R*C的网格上进行，每个格子有一个取值在1-k之间的整数标号，奶牛开始在左上角的格子，目的是通过若干次跳跃后到达右下角的格子，当且仅当格子A和格子B满足如下条件时能从格子A跳到格子B：
1.B格子在A格子的严格右方(B的列号严格大于A的列号)
2.B格子在A格子的严格下方(B的行号严格大于A的行号)
3.B格子的标号和A格子的标号不同
请你帮助奶牛计算出从左上角的格子到右下角的格子一共有多少种不同的方案

输入格式
第一行包含两个整数R C K
接下来的R行，每行C个整数表示格子的标号

输出格式
一行，代表有多少种不同的方案，由于答案很大，请输出答案对1000000007取模的结果

1<=k<=R*C
2<=R,C<=750

NKOJ3208 奶牛跳格子

一个R*C的网格上，每个格子有1-k之间的整数标号，开始在左上角的格子，通过若干次跳跃到达右下角的格子，满足如下条件时能从格子A跳到格子B：
1.B格子在A格子的严格右方(B的列号严格大于A的列号) 2.B格子在A格子的严格下方(B的行号严格大于A的行号) 3.B格子的标号和A格子的标号不同
请你帮助奶牛计算出从左上角的格子到右下角的格子一共有多少种不同的方案

设f[i][j]表示从起点到达坐标(i,j)格子的方案数

f[i][j]=∑f[x][y]
=∑f[x][y] -∑f[x'][y']

条件 x<i 且 y<j 且 a[i][j]!=a[x][y]
条件 x<i 且 y<j x'<i 且 y'<j 且 a[i][j]==a[x'][y']

从dp条件来看，是个三维偏序问题

行间进行前缀和优化操作。每一行，对列分治，统计左边对右边的贡献
用cdq分治处理列时，先计算左部的f值，再讨论左部对右部的贡献，然后分治计算右部的f值。
cdq分治本质是在保证在处理[mid+1,r]区间时，所有[l,mid]区间贡献全部计算完了

右侧代码中：
Sum1记录∑f[x][y]
Sum2记录∑f[x'][y']
复杂度O(n*m*logn)

```
//main()
for(i=1;i<=n;i++)
    for(j=1;j<=m;j++)cin>>a[i][j];
f[1][1]=1;
cdq(1,m);
cout<<f[n][m];
```

```
void cdq(int l,int r)
{
    if(l==r) return;
    int mid=l+r>>1;
    cdq(l,mid);
    for(int i=1; i<=n; i++)
        for(int j=l; j<=r; j++) Sum2[a[i][j]]=0;
    int Sum1=0;
    for(int i=2; i<=n; i++)
    {
        for(int j=l; j<=mid; j++)
        {
            Sum1+=f[i-1][j];
            Sum2[a[i-1][j]]+=f[i-1][j];
        }
        for(int j=mid+1; j<=r; j++)f[i][j]+=Sum1-Sum2[a[i][j]];
    }
    cdq(mid+1,r);
}
```

奋斗吧少年

巨大的成功需要付出巨大的代价

no sacrifice, no success

nkoj6652,3238,3571,4599