字符串DP

PKU Hzyoi

01 区间DP

- 区间型动态规划的一个明显特征，就是信息可以维护在区间的端点上。
- 多数题目的状态都是由区间（类似于dp[l][r]这种形式）构成的
- 转移一般是把大区间转化成小区间来处理，然后对小区间处理后再回溯的求出大区间的值

石子归并

- 有n堆石子排成一列，每堆石子有一个重量w[i]，每次合并可以合并相邻的两堆石子，一次合并的代价为两堆石子的重量和w[i]+w[i+1]。问安排怎样的合并顺序，能够使得总合并代价达到最小。

- $n \leqslant 200$

- 用f[i][j]表示合并第i堆到第j堆石子的最小代价
- 考虑状态间的转移
- 你要合并第i堆到第j堆石子，思考最后一次合并的两堆石子，它们一定是第i堆到第k堆以及第k+1堆到第j堆石子合并后的产物，因此得到一下转移方程
- 转移方程：f[i][j]=max{f[i][k]+f[k+1][j]}+sum(i, j) (i≤k<j)
- sum(l, r)=$\sum_{i=l}^{r} w[i]$

[Codeforces712E] Memory and Casinos

- 对于一类在数轴上往返规则运动，进入一个区间必须经过区间端点的问题，可以考虑区间型动态规划。
- 数轴上有N+1个位置，如果位于第i个位置，则有pi的概率下一步移动到第i+1个位置，有1-pi的概率移动到第i-1个位置。要求完成Q个操作：
- ①修改某一个pi；
- ②询问从第L个位置开始，不经过第L-1个位置，最后到达第R+1个位置结束的概率，其中L≤R≤N。
- N≤100000,Q≤100000。

- 对于每组询问，设F[i]为从L出发移动到i的概率。可以列出动态规划方程F[i]=(1-pi)*F[i+1]+Pi*F[i-1]。用高斯消元解出F[l..r]即可。
- 观察上面的方程，可以发现，F[l+1..r]均可以用含F[l]的多项式表示。因此可以线型推出F。

- 在线段树上维护两个标记，从区间一侧开始另一侧移出的概率。因为移出区间的总概率是1，因此可以得到从区间一侧开始同一侧移出的概率。
- 合并区间时，设左区间左边开始左边移出的概率为p1，左区间右边开始右边移出的概率为p2，右区间左边开始左边移出的概率为p3，母区间左边开始左边移出的概率即为：
- p1+(1-p1)*p3*(1-p2) +(1-p1)*p3*(1-p2)*((p2*p3)^1)+(1-p1)*p3*(1-p2)*((p2*p3)^2)+……
- 而这个式子可以作差化简。同理可以得到母区间右边开始右边移出的概率。

02

字符串DP

[Codeforces607B] Zuma

- 给出一个字符串，每次可以从中提取一个回文子串。求至少要提取多少次，才能把整个串取完。
- N 500

- f[i][j]表示取完区间[i,j]的所有字符需要的最少操作数。
- 可以讨论c[i]和c[j]是否相等来进行转移。

[USACO07OPEN]便宜的回文Cheapest Palindrome

- 字串S长M，由小写字母构成。欲通过增删字母将其变为回文串，增删特定字母花费不同，求最小花费
- M 1000

- 比较经典的字符串DP了。
- 设dp[i][j]表示区间[i,j]变成回文串的最小花费，需要用到区间DP的思想。
- 考虑如何用一个小区间更新一个大区间。
- 如果大区间是dp[i][j]，若s[i]=s[j]，那么dp[i][j]=dp[i+1][j−1]，即当前大区间可以由去掉其两端的小区间更新而来而不用花费。
- 不等的时候，dp[i][j]=min(dp[i+1][j]+min(add[s[i]],del[s[i]]),dp[i][j−1]+min(add[s[j]],del[s[j]]))

一类消除子串问题

- [Croatian2010] Zuma
- 给出一个长度为N的字符串，要求插入尽量少的字符，满足其能够从中不断删去连续相同长度不小于K的子段，直到字符串为空。$N \le 100$，$K \le 5$。

- 如果直接用状态F[i][j]表示字符串区间[i,j]被删去需要插入的最小字符，方程很难转移。
- 因此需要增设两维k、l，表示字符串区间[i,j]被删到只剩下k个字符l需要插入的最小字符数量。但是，考虑转移，这个状态设计存在冗余。
- 最后的状态是：F[i][j][k]表示字符串区间[i,j]在前面挂着k个和s[i]相同的字符时，需要插入的最小字符数量。
- 转移有3种
- dp[l][r][k−1]=dp[l+1][r][0]
  dp[l][r][c]=dp[l][r][c+1]+1
  dp[l][r][c]=dp[l+1][i−1][0]+dp[i][r][c+1] (s[i]=s[l])
- 时间复杂度O(N^3K)。

神奇的数列

- 一个正整数数列，可以将它切割成若干个数据段，每个数据段由值相同的相邻元素构成。
  该数列的神奇之处在于，每次切除一个数据段后，
  该数据段前后的元素自动连接在一起成为邻居。例如从数列“2 8 9 7 7 6 9 4”中切除数据段“7 7 ”后，余下的元素会构成数列“2 8 9 6 9 4”
  请问若要将该数列切割成若干个数据段，则至少会切出来几个数据段？
- 样例：按下列顺序切割数列“2 8 9 7 7 6 9 4”，只要切割成6段
  切割出“7 7”，余下“2 8 9 6 9 4”
  切割出“6”，余下“2 8 9 9 4”
  切割出“9 9”，余下“2 8 4”
  切割出“2”，余下“8 4”
  切割出“8”，余下“4”
- N<=200

- 类似上一题，但消除时和序列长度无关，于是最后一位可以不记
- F[l][r]表示区间[l,r]，前面挂着若干个和s[l]相等的字符时的答案
- 转移2种
- F[l][r]=f[l+1][r]+1
- F[l][r]=min(f[l][i-1]+f[i][r]) (s[i]==s[l])

洛谷P1279 字串距离

- 设有字符串X，我们称在X的头尾及中间插入任意多个空格后构成的新字符串为X的扩展串，如字符串X为” abcbcd”，则字符串“abcb□cd”，“□a□bcbcd□”和“abcb□cd□”都是X的扩展串，这里“□”代表空格字符。
- 如果A1是字符串A的扩展串，B1是字符串B的扩展串，A1与B1具有相同的长度，那么我扪定义字符串A1与B1的距离为相应位置上的字符的距离总和，而两个非空格字符的距离定义为它们的ASCII码的差的绝对值，而空格字符与其他任意字符之间的距离为已知的定值K，空格字符与空格字符的距离为0。在字符串A、B的所有扩展串中，必定存在两个等长的扩展串A1、B1，使得A1与B1之间的距离达到最小，我们将这一距离定义为字符串A、B的距离。
- 请你写一个程序，求出字符串A、B的距离。

- dp[i][j] 表示第一个串的前i个字符和第二个串的前j个字符的最优值
- 两个空格对应显然没有意义，那么有3种转移：
- dp[i−1][j]+K，dp[i][j−1]+K，dp[i−1][j−1]+abs(S1[i]−S2[j])
- 分别表示S1[i]与空格匹配，S2[j]与空格匹配，S1[i]与S2[j]匹配。

NOIP2015 D2T2

- 有两个字符串A和B。现在要从字符串A中取出k个互不重叠的非空子串，然后把这k个子串按照其在字符串A中出现的顺序依次连接起来得到一个新的字符串，问有多少种方案可以使得这个新串与字符串B相等？
- A的长度≤1000，B的长度≤200，k≤200。

- 考虑最朴素的DP，f[i][j][k]表示A匹配到第i个位置，B匹配到第j个位置，当前已经匹配了k个不重叠子串的方案数。
- 然后转移：

$$f[i][j][k]=\sum_{l1=1}^{x}\sum_{l2}^{i-l1} f[i-l1-l2][j-l1][k-1]$$

x表示$prefix(A[i]), prefix(B[j])$的最大后缀匹配

- 这样朴素算法的效率是O(nkm^3)
- 然后因为是和，所以我们可以求一个前缀和，然后转移的效率就会优化到O(nkm)。
- 因为内存限制，这题还要用滚动数组压掉k的一维。

一个题

- 有两个长度分别为n,m的数字串，求他们公共的递增子序列的最大长度
- 样例输入：
- 5 1 4 2 5 -12
- 4 -12 1 2 4
- 样例输出：
- 2
- 数据范围：
- 30%数据1<=n,m<=50；100%的数据,1<=n,m<=1000,Ai，Bi在长整型范围内。

题解：

- 此题就是求两个串的最长公共上升子序列长度。
- 设f[i][j]表示第一个串做到i，第二个串匹配到j的最大长度。(i不一定匹配)
- 转移的时候f[i+1][j]=max(f[i+1][j],f[i][k]+(a[i+1]==b[j]))
- (1<=k<j且b[k]<a[i+1])。
- 答案就是f数组的最大值。
- 实际做的时候可以把数组的第一维去掉。
- 时间复杂度O(nm)

*[51Nod1202] 子序列个数

- 子序列的定义：对于一个序列a=a[1],a[2],......a[n]。则非空序列a'=a[p1],a[p2]......a[pm]为a的一个子序列，其中1<=p1<p2<.....<pm<=n。
- 例如4,14,2,3和14,1,2,3都为4,13,14,1,2,3的子序列。对于给出序列a，有些子序列可能是相同的，这里只算做1个，请输出a的不同子序列的数量。由于答案比较大，输出Mod 10^9 + 7的结果即可。

解题报告

- 我们的问题是求整个a序列的不同子序列个数，那么不妨考虑其子问题。我们记a序列中前i个数组成的序列a1,a2...ai为前缀序列i，考虑求解前缀序列i的不同子序列个数，记为dp[i]
- 首先考虑总的子序列个数怎么求（不考虑重复），dp[i]=dp[i−1]×2，因为前缀序列i和前缀序列i−1相比只多了ai,那么ai的选取与否就会导致子序列的不同，所以dp[i]=dp[i−1]×2
- 但现在我们要求的是不同的子序列个数，因此要减去重复的子序列个数。对于选取了ai的子序列，会出现重复的序列一定是以某一个满足(ai=aj)的aj结尾的，那么重复的子序列个数就是dp[j−1]，其中j是最大的满足aj=ai并且j<i的数。
- 因此我们得到转移方程：
- dp[i]=dp[i−1]×2−dp[j−1]
- 以序列(1,3,4,2,3,2,1,4)为例，a5=3，因此 dp[5]=dp[4]×2−dp[1]，因为在a5之前的上一个3出现在位置2，利用容斥原理，我们需要剪掉重复的部分。

非常感谢您的观看

THANK YOU FOR YOUR WATCHING