区间 DP 选讲

黄敬哲

2016 年 8 月 13 日

区间 DP 是什么

区间 DP 是什么

- 直观地感受下大概就是一个连续的区间的结果可以通过已经得到的它的连续子区间的结果快速转移过来

区间 DP 是什么

- 直观地感受下大概就是一个连续的区间的结果可以通过已经得到的它的连续子区间的结果快速转移过来
- 自认为求最值的区间 DP 相对容易求方案数的区间 DP 在防止重复统计上比较折磨人

一些显然是区间 DP 不过有点麻烦的问题

CodeForces 149D Coloring Brackets

题意简述

给出一串两两匹配的括号 (合法的括号序列)，现在要给它们涂色，每个括号可以选择蓝，红，无色三种情况，但是相邻的括号如果都有颜色则不能相同 (可以同为无色)[条件 1]，每一对匹配的括号都有且仅有一个括号染色 [条件 2]。

序列长度 n <= 700

input

(())

(()())

output

12

40

CodeForces 149D Coloring Brackets

思考过程

很明显的区间 DP

于是首先要开两维 (700 * 700) 记录左右端点位置, 为了满足 [条件 1], 我们可以再开两维 (3 * 3) 记录左右端点的颜色, 如果没有 [条件 2] 的限制就把此题解决了, 但是出题人显然不希望自己出的题目被太快过掉, 于是有了 [条件 2] 的限制

CodeForces 149D Coloring Brackets

思考过程

很明显的区间 DP

于是首先要开两维 (700 * 700) 记录左右端点位置, 为了满足 [条件 1], 我们可以再开两维 (3 * 3) 记录左右端点的颜色, 如果没有 [条件 2] 的限制就把此题解决了, 但是出题人显然不希望自己出的题目被太快过掉, 于是有了 [条件 2] 的限制

再考虑 [条件 2] 这里可以通过一个栈来线性模拟出匹配关系

1. 若 L,R 为匹配的括号则对区间 [L+1,R−1] 进行讨论

2. 否则对 [L,m[L]],[m[L]+1,R] 进行讨论 (m 为该左括号匹配的右括号位置)(此处的区间分割方案也是防止重复统计的)

最后对于 L+1=R 的区间特判即可

CodeForces 149D Coloring Brackets

思考过程

很明显的区间 DP

于是首先要开两维 (700 * 700) 记录左右端点位置, 为了满足 [条件 1], 我们可以再开两维 (3 * 3) 记录左右端点的颜色, 如果没有 [条件 2] 的限制就把此题解决了, 但是出题人显然不希望自己出的题目被太快过掉, 于是有了 [条件 2] 的限制

再考虑 [条件 2] 这里可以通过一个栈来线性模拟出匹配关系

1. 若 L,R 为匹配的括号则对区间 [L+1,R−1] 进行讨论

2. 否则对 [L,m[L]],[m[L]+1,R] 进行讨论 (m 为该左括号匹配的右括号位置)(此处的区间分割方案也是防止重复统计的)

最后对于 L+1=R 的区间特判即可

实现的时候用记忆化搜索比较方便

HDU 5396 Expression

题意简述

给 n 个数, 以及中间的 n-1 个运算符 (只含 +-*), 现在可以每次选两个数按照两数之间的运算符先算出结果, 问所有可能的操作顺序得到的结果之和是多少 (mod 1e9 + 7)

n <= 100

input

3 2 1

-+

1 4 6 8 3

+*-*

output

2

999999689

HDU 5396 Expression

思考过程
看完数据范围后，应该有不少人会立即反应这是 n^3 的区间 DP，
以 F[i][j] 表示从 i 到 j 这个区间所有情况之和，
然后再枚举中间点 k 从 F[i][k] 到 F[k+1][j] 转移过来，
但此题绝不是想到区间 DP 就能 A 了)

HDU 5396 Expression

我们假设合并时，左区间所包含的情况为 a1,a2……ap 右区间所包含的情况为 b1,b2……bq

对于乘法运算，由于乘法分配率这个性质直接把两边所有情况之和乘起来就行了

对于加减法运算，左边的区间每种情况贡献次数为 q 右边区间每种情况贡献次数为 p

进一步，我们可以发现，出现次数 p、q 其实也就等于 (区间长度-1) 的阶乘

看起来该做的事情已经做完了，然而如果就这样写完代码，会发现连样例都过不了

HDU 5396 Expression

多想想之后，我们发现这样一个问题，虽然合并时，左右区间都是确定的，然而达到同一个左右区间的方案并不是唯一的

我们先考虑一个比较小的情况，假设左区间通过 c1,c2 这两个操作得到，右区间通过 d1,d2 这两个操作得到

那么我们只要保证同一区间内操作的有序性即可使得最后得到的两个区间分别相同

比如 c1,c2,d1,d2 和 c1,d1,c2,d2 得到的区间就是相同的

这样可能的情况就有 C_4^2 种推广到其他情况便是

C((左区间长度-1)+(右区间长度-1),(左区间长度-1))(注意到区间长度-1 即为合并过程中的操作数)

SCOI2007 压缩

题意

给一个由小写字母组成的字符串，我们可以用一种简单的方法来压缩其中的重复信息。

压缩后的字符串除了小写字母外还可以（但不必）包含大写字母 R 与 M，其中 M 标记重复串的开始，R 重复从上一个 M（如果当前位置左边没有 M，则从串的开始算起）开始的解压结果（称为缓冲串）。

bcdcdcdcd 可以压缩为 bMcdRR,abcabcdabcabcdxyxyz 可以被压缩为 abcRdRMxyRz. 求压缩后的字符串的最小长度

输入字符串长度 $<= 50$

input

bcdcdcdcdxcdcdcdcd

output

12

SCOI2007 压缩

思考过程

首先我们可以很明显地观察到即使缓冲串长度只有 1，用一个 R 来重复它也不会比什么都不用差

因此如果能用 R 就尽量去用

SCOI2007 压缩

思考过程

首先我们可以很明显地观察到即使缓冲串长度只有 1，用一个 R 来重复它也不会比什么都不用差

因此如果能用 R 就尽量去用

但是这题毕竟不是能够直接贪心去做的题目，用 M(即清空缓冲串)的位置还是得用 DP 来判断下

SCOI2007 压缩

因此我们可以用 f[L][R][k] 来表示区间 [L,R] 的字符串压缩后的最短长度

k = 1 代表 [L,R] 内放了 M,k = 0 代表 [L,R] 内没放 M 如果区间内放了 M 就暴力枚举放 M 的位置

f[L][R][1] = min(f[L][R][1], f[L][i][k1] + f[i + 1][R][k2] + 1)

如果区间是由两个完全相同的字符串拼起来的话就直接在中间放一个 R

if(check(L, R))f[L][R][k] = min(f[L][R][k],f[L][mid][0] + 1)

LightOJ 1422 Halloween Costumes

题意简述

按照时间顺序给定要参加的 n 个聚会分别要穿的衣服的类型，可以套在外面穿

但是一旦脱下就不可再穿，问参加 n 个聚会至少要准备几件衣服。

n <= 100

input

4

1 2 1 2

7

1 2 1 1 3 2 1

output

3

4

LightOJ 1422 Halloween Costumes

思考过程

根据样例模拟一下整个过程

当要去另一场聚会时，实际上可能得到最优解的只有两种决策

1. 直接把应该穿的衣服套在最外面

2. 如果穿在身上的衣服中有一件的类型与需要的类型相同，可以把这件衣服外的衣服都脱下

LightOJ 1422 Halloween Costumes

思考过程

根据样例模拟一下整个过程

当要去另一场聚会时，实际上可能得到最优解的只有两种决策

1. 直接把应该穿的衣服套在最外面

2. 如果穿在身上的衣服中有一件的类型与需要的类型相同，可以把这件衣服外的衣服都脱下

我们会发现这是一个栈

所以这和区间又有什么关系？

LightOJ 1422 Halloween Costumes

多想想或许可以发现，对于一个栈内元素，显然会有一堆连续的元素在它之后入栈和出栈

(栈的特殊性就是，在它之后入栈的一定比它先出栈，所以是连续的一段)

而我们此时则可以根据这个元素入栈后再一次成为栈顶元素的时候来对区间进行划分

(实际上也就是题意中，穿上某件服装后，再一次地使得它成为最外面的一件)

LightOJ 1422 Halloween Costumes

多想想或许可以发现，对于一个栈内元素，显然会有一堆连续的元素在它之后入栈和出栈

(栈的特殊性就是，在它之后入栈的一定比它先出栈，所以是连续的一段)

而我们此时则可以根据这个元素入栈后再一次成为栈顶元素的时候来对区间进行划分

(实际上也就是题意中，穿上某件服装后，再一次地使得它成为最外面的一件)

于是转移方程可以写为

f[i][j]=f[i+1][j]+1(第 i 个聚会所穿衣服未被多次利用)

f[i][j]=min(f[i][k−1]+f[k+1][j])(第 i 个聚会所穿衣服在第 k 个聚会被再次利用)