DP 难题选讲

samjia2000

Liar

给定两个字符串 S 和 T 。你可以把 S 分成若干段，从左到右从 1 开始编号。现在要从你分出来的段中取出不超过 x 段，按编号从小到大依次拼接成字符串 T 。问是否有可行解。

$1 \le |T| \le |S| \le 100000$ $x \le 30$

Liar

考虑 DP 。设 f[i][j] 表示用了 S 的前 i 个字符，取出了 j 段，最大能得到 T 的前多少个字符。

转移分两种情况:
1. f[i][j]—>f[i+1][j] 表示字符 i+1 单独成段且没被选
2. f[i][j]—>f[i+lcp][j+1] 其中 lcp 是 S 的后缀 i+1 和 T 的后缀 f[i][j]+1 的 lcp 。这里表示这一段 lcp 被选中。很显然取 lcp 的长度比取小于 lcp 的长度要更优。

最终复杂度就取决于如何求 lcp 了。设字符串总长为 N ，如果用后缀数组加 rmq 可以做到 O(NlogN) 预处理 +O(1) 查询。不过时限较大，可以用 hash+ 二分，时间复杂度为 O(NxlogN)

相似名字集

有 n 个单词，现在要把它们重新排列。有 m 条形如 (a,b) 的限制，表示第 a 个单词必须是第 b 个的前缀

问有多少种合法的排列。答案对 1e9+7 取模

$2 \le n \le 50, 0 \le m \le 8$，一个单词长度不超过 50

相似名字集

首先前缀关系让人联想到用 trie 来解决问题
其次 m 比较小，涉及到的字符串个数至多为 2*8=16 种
于是可以设 f[i][S]，表示 trie 上以 i 为根的子树中，已确定了涉及到的字符串状态为 S 的方案数
但是复杂度还是很大
优化 1： 在 trie 上只保留作为字符串结尾的节点
优化 2： 若子树 i 包含 a 且 a 是 b 的前缀，儿子是不可能不包含 b 的，可以优化枚举状态数
另外要注意一些 trick，比如规则会形成环

赛艇

在首尔城中，汉江横贯东西。在汉江的北岸，从西向东星星点点地分布着个划艇学校，编号依次为 1 到 n 。每个学校都拥有若干艘划艇。同一所学校的所有划艇颜色相同，不同的学校的划艇颜色互不相同。颜色相同的划艇被认为是一样的。每个学校可以选择派出一些划艇参加节日的庆典，也可以选择不派出任何划艇参加。如果编号为的学校选择派出划艇参加庆典，那么，派出的划艇数量可以在 Ai 至 Bi 之间任意选择（Ai<=Bi）。值得注意的是，编号为 i 的学校如果选择派出划艇参加庆典，那么它派出的划艇数量必须大于任意一所编号小于它的学校派出的划艇数量。

输入所有学校的 Ai 、 Bi 的值，求出参加庆典的划艇有多少种可能的情况，必须有至少一艘划艇参加庆典。两种情况不同当且仅当有参加庆典的某种颜色的划艇数量不同

n<=500

赛艇

- 先从最暴力的dp入手：设f[i][j]表示前i个区间最后一个数是j
- 尝试去优化状态：可以用Li,Ri把数轴分成2n段，上面状态的j意义改为在第j段
- 我们发现这样会算重，那么又要枚举转移过来的f[i'][j']，并乘组合数。复杂度是四次方的
- 考虑给dp再加一维！f[i][j][k]，其中i,j意义不变，k表示在第j段放了k个数
- 设len[j]表示第j段的长度，然后转移如下：
- $f[i][j][k] = f[i-1][j][k] + f[i-1][j][k-1] * {}^{len[j]-k+1}/_{k}$ ，$\mathrm{k} > 1$
- $f[i][j][1] = f[i-1][j][1] + \sum_{j'=0}^{j-1} \sum_{k=0}^{i-1} f[i-1][j'][k] * len[j]$
- 试着从组合数角度理解为什么乘的系数是这样的
- 加上前缀和优化并开滚动数组即可通过此题

有限背包计数问题

你有一个大小为 n 的背包，你有 n 种物品，第 i 种物品的大小为 i ，且有 i 个，求装满这个背包的方案数有多少
两种方案不同当且仅当存在至少一个数 i 满足第 i 种物品使用的数量不同

1<=n<=10^5 ，答案对 23333333 取模

有限背包计数问题

将数字按照小于等于根号 n 和大于根号 n 分成两类。
那么第二类的可以看成选择每种可以选择无限个，对答案没有影响。
对于前面的部分，可以直接背包。
对于后面的部分，最多可以选择根号 n 个
记 f[i][j] 表示从 1..n 中选择了 i 个可重复的数，使得这 i 个数的和为 j 的方案数。
那么：
----
f[i][j] ==> f[i][i+j]
f[i][j] ==> f[i+1][j+1]

从 sqrt(n)+1..n 中选择 k 个和为 x 的方案数则为 f[k][x-sqrt(n)*k]
时间复杂度 O(n sqrt n)

Almost Increasing Array

定义一个序列 A 是 "Almost Increasing" 的，当 A 满足从 A 中去掉一个位置之后剩下的序列形成了一个严格递增序列。
给出一个长度为 n 的序列，问最小需要改动多少个位置的值，使得这个序列变成 "Almost Increasing" 的。
$n<=200000$

Almost Increasing Array

先来考虑这样一个问题：最少需要多少操作才可以将这个序列变成严格递增的序列。
假如我们有 m 个位置不会发生变动， p[1],p[2]...p[m]，其中， p[i]<p[i+1]
那么，需要满足的就是 a[p[i]]-a[p[i-1]]>=p[i]-p[i-1]
即 a[p[i]]-p[i]>=a[p[i-1]]-p[i-1]
所以，只需要求 a[i]-i 的最长不下降子序列就可以了。
回到本题，由于 "Almost Increasing" 的定义里，出现了删除一个位置的条件，那么，对应到原序列中，其实就是所有的 a[i]-i 变成了 a[i]-i+1，所以可以维护两个树状数组，一个表示前面还没有删过数字，另一个表示前面已经删过，用这个优化 DP 即可。
时间复杂度 O(n log n)

Exciting Finish!

有 n 个选手在竞赛，第 i 个人的分数现在是 a[i] 。

最后一轮中，第 i 个人的得分是 b[i] ，满足 sigma(b[i])=k 且 b[i] 都是非负整数。

将人的编号按照 b[i] 从小到大排序，如果 b[i] 相同，那么 a[i] 较小的排前面，记排序得到的编号序列为 num[i]

如果满足 $a[num[1]]+b[num[1]]>max\{a[i],1<=i<=n\}$ 且对于 $2<=i<=n$ 都有 $a[num[i-1]]+b[num[i-1]]<a[num[i]]+b[num[i]]$ ，那么就说这个序列 b 是优美的。

对于每一种可能的最终排名 P ，如果存在一个优美的序列 b 得到的最终排名是 P ，那么就说排列 P 是优美的。

问优美的排列 P 的数量。

$n<=12,k<=700$

Exciting Finish!

由于要求的是优美的排列 P 的数量，但是我们发现一个优美的排列可能对应多个优美的序列 b，所以要增加限制来使得排列与序列形成一个单射的映射（即对于每种优美的排列只有唯一对应的序列 b）

那么，一个可行的思路就是，对于每个 i，强制 b[num[i]] 为最少的使得 a[num[i]]+b[num[i]]>a[num[i-1]]+b[num[i-1]] （对于 i=1 时后面的部分就是 max{a[i]}）的值，那么，显然我们就得到一个合法的单射。

接着考虑状压 DP，从后往前确定最终排名中每个数是什么，记 f(S,x,p) 表示现在排名最大的若干个的状态是 S，最小的一个是第 x 个人，他们的 b[i] 的和是 p 且强制 b[x]=0

那么，转移的时候就是枚举一个 y 满足 y 不在 S 中，然后记 d=max(0,a[y]+1-a[x])，那么 d 就是使得 a[x]+b[x]>a[y]+b[y] 最小的 b[x]-b[y]，转移到 f(S|{y},y,p+|S|*d)

最后判断每个状态的开头是否满足 a[num[1]]+b[num[1]]>max{a[i]} 即可。

Checkers

设 $X=10^{100}$，有 n 个点在数轴上，第 i 个点的坐标是 X^i

每次操作可以选择两个点 A 和 B，然后作出点 A' 与 A 关于 B 对称，保留 A' 删掉点 A 和 B

经过 n-1 操作后可以得到一个点，问这个点的坐标取值可能有多少。

$n<=50$

答案对 10^9+7 取模。

Checkers

发现 X 是一个很大的量，所以对于一个坐标，只需要关心 X^i 的系数。
对于一个点，他可以被表示成 sum(c[i]*x^i) ，即一个关于 x 的多项式
考虑一次对称，实际上就是对于多项式 f_i(x) 和 f_j(x) 保留 2*f_i(x)-f_j(x)
考虑 x^i 对最后坐标的贡献，即 x^i 最后的系数，容易发现，一定是 (-1)^p*2^k 的形式
如果我们确定系数中每个 2^k 以及 -2^k 的个数就可以计算出答案了。

Checkers

设 f[k] 表示系数为 2^k 的个数， g[k] 表示系数为 -2^k 的个数。
那么，当 f 和 g 满足什么关系的时候是可能出现的呢?
对于 k， 2^k 和 -2^(k-1) 合并会得到 2^(k-1)，即 g[k-1] 中会有 f[k] 个与 2^k 合并变成 -2^(k-1)，反过来也是类似的，所以就是要满足 f[k]+f[k-1]>=g[k],g[k]+g[k-1]>=f[k]。
对于最后 k=0 的时候，要求最后剩下的那个数一定要使 2^0，即 f[0]=1,g[0]=0
于是就可以 Dp 了，设 f[i][x][y] 表示现在已经有 i 个元素，在目前最小的 k 里， f[k]=x,g[k]=y，转移的时候枚举 a=f[k-1] 和 b=g[k-1]，转移到 f[i+a+b][a+x-y][b+y-x] 即可。
时间复杂度 O(n^5)，实际上更改一下枚举顺序可以将复杂度降至 O(n^4).

Tree ans Robots

有一棵 n 各节点的根为 1 的有根树，每条边有边权，初始根节点上面有 r 个机器人，每个机器人从 1 开始走一条路径然后在任意一个节点结束，可以经过重复节点以及重复的边，要求每条边都至少要被经过一次，一条路径的权值为经过的边的边权和（经过多次算多次），问最小的总权值是多少。

n<=50000,r<=50

Tree ans Robots

考虑 r=1 的时候，实际上答案就是， 2*sigma(边权)-[距离根节点最远的节点距离]
如果已经知道了结束的 r 个节点是什么（称为关键点），那么假设一条边下面的子树里有 k 个关键点，这条边就要被计算 k 次，否则就要被计算两次，然后就可以写一个 O(nk) 的 DP 了。
这个同时也是斜率单调的，可以二分斜率做。

Ceiling Numbers

有 n 个长度至多为 1000 的数字 B，以及一个长度至多为 1000 的数字 A，对于 0~9 中的数字 i 有花费 c[i]，其中 A 有一些位置是？，现在要求在？的位置填上数字，然后将所有 B 都加上 A，然后要求得到的所有数字每一位数字的花费和最大。

n<=1000

Ceiling Numbers

对于第 i 位，如果将所有数按照 1~i 位组成的数字排序，那么进位的一定是这个序列的一段后缀，于是可以 DP ，设 f[w][i] 表示现在已经确定了 A 的 1~w 这些位，然后进位的数是后面 i 个，转移的时候可以先把每个数按照 1~w+1 组成的数的大小关系从小到大排序，然后可以根据最高位被分成 10 段，枚举 A 中当前位填什么，然后需要讨论的是进位的是哪些数字，可以发现，随着 w 中进位的数的位置的递增， w+1 中每段会得到进位的位置也是递增的，于是可以用一个指针维护一下即可。时间复杂度 O(n*1000*10)

A

B

END