动态规划

fateice

Badania naukowe

- 给定三个数字串 A,B,C，请找到一个 A,B 的公共子序列，满足 C 是该子序列的子串。最大化子序列长度。
- n<=3000。

Badania naukowe

- 设 f(i,j) 为 A[1..i] 与 B[1..j] 的 LCS，g(i,j) 为 A[i..n] 与B[j..m] 的 LCS。
- 若 C 为空串，则 ans = f(n,m)。
- 否则，设 di 为 A 中从 i 开始往右匹配上 C 串的结束位置，ei 为 B 中从 i 开始往右匹配上 C 串的结束位置。
- 那么 ans = max(f(i - 1,j - 1) + g(di + 1,ej + 1) + |C|)。
- 时间复杂度 O(n^2)。

数字之积

- 一个不含前导 0 的数 x 各个数位上的数字之积记为 f(x)。
- 给定 n, L, R , 求 [L, R] 中满足 $0 < f(x) \leq n$ 的数的个数。
- $0 < L \leq R < 10^{18}$, $n \leq 10^9$。

数字之积

- 数字之积非常大，但是这些数字的质因子只可能是 2、 3、5、 7。
- 所以设 f(i, cnt2, cnt3, cnt5, cnt7, j) 为从高到低填了前 i 位，2、 3、 5、 7 的个数分别为 cnt2、 cnt3、 cnt5、 cnt7，是否小于 R 的状态为 j 的数字个数，然后 DP 即可。
- 时间复杂度 O(logRlog^4n)。

充电

- 有n个结点，相互之间通过n-1条边连接，形成一棵树。
- 第i个节点有xi的概率直接充电，第j条边有yj的概率导电。
- 求期望有电的结点数。
- n<=10^6

充电

- 有电的概率比较难求，我们考虑求出每个点没电的概率。
- 用fi表示不考虑父亲的情况下第i个点没电的概率，fi=(1-xi)*π(fj+(1-fj)*(1-yk))。
- 用gi表示第i个点的父亲没有向第i个点导电的概率，t=f[fa[i]]/...*g[fa[i]]，g[i]=t+(1-t)*(1-yk)。
- fi*gi就是i没电的概率。
- 时间复杂度O(n)

石子合并

- 有n堆石子排成一排，第i堆有xi个。
- 每次选定两堆相邻的石子合并成一堆，代价为石子个数。
- 求最小代价。
- n<=5000

石子合并

- 令w[i][j]表示第i~j堆石子的总个数，f[i][j]表示将第i~j堆石子合并为一堆的最小代价。
- f[i][j]=min{f[i][k]+f[k+1][j]}+w[i][j]。
- 使用四边形不等式优化。
- 时间复杂度O(n)

Data Structure You've Never Heard Of

- 给定一个长度为 n 的序列 a1,a2...an，每个元素都是一个 d 维 01 向量，求所有不下降子序列的个数。
- 对于 d 维向量， ai ≤ aj 等价于 ai 的每一维都不大于 aj。
- 1 ≤ n ≤ 100000, 1 ≤ d ≤ 16。

Data Structure You've Never Heard Of

- 一个 d 维向量可以用一个 d 位二进制数来表示，$a \leq b$ 等价于 a 是 b 的子集。
- 考虑两种暴力，一种是记录每种数字的贡献，修改 O(1)，查询 O(2^d)。
- 另一种则是维护一个高维前缀和，修改 O(2^d)，查询 O(1)。
- 两种暴力在修改和查询上各有所长，所以把它们结合在一起即可得到复杂度优秀的算法。

Data Structure You've Never Heard Of

- 考虑 d = 16 的情况，将 16 维划分为前 8 维和 8 维，对于前 8 维的每一个数维护后 8 维的前缀和。
- 修改的时候，只要暴力修改某一个前 8 维里的前缀和，时间复杂度 O(2^8)。
- 查询的时候，暴力枚举前 8 维，后 8 维可以 O(1) 得到，时间复杂度 O(2^8)。
- 总时间复杂度 O(n*2^(d/2))。

Beautiful numbers

- 给定 L, R，求 [L, R] 中所有可以被它所有非零数位整除的数的个数。
- 1 ≤ L ≤ R ≤ 9*10^18。

Beautiful numbers

- 注意到 lcm(1, 2, 3, 4, 5, 6, 7, 8, 9) = 2520，且 1 到 9 中任意一个集合的 lcm 只有 49 种。所以可以考虑设 f(i, j, k, l) 为从高到低填了前 i 位，之前拼成的数字 mod2520 = j，已选用数位的 lcm 为 k，是否小于 R 的状态为 l 的数字个数，然后 DP 即可。
- 对于 k，可以在一开始预处理出所有 49 种情况，对其进行重标号，以剔除无效状态。
- 时间复杂度 O(2520 * 49 log R)。

小P的牧场

- 有n个牧场，编号为1~n，第i个牧场建立控制站的代价为ai，放养量为bi。
- 选择一些牧场建立控制站，每个控制站控制它和上一个控制站之间的牧场（包括它自己所在的牧场）。
- 第i个牧场被第j个牧场的控制站控制的代价为(j-i)*bi。求最小代价。
- n<=10^6

小P的牧场

- 假设只在n建站，费用为sigma(b[i]*(n-i))。
- f[i]表示只在i~n建站，且i有建的最大节省代价。
- f[i]=max{f[j]+s[i]*(j-i)}-a[i] (s[i]是b[i]的前缀和)
- 斜率优化。
- 时间复杂度O(n)

Nim z utrudnieniem

- A 和 B 两个人玩游戏，一共有 m 颗石子， A 把它们分成了n 堆，每堆石子数分别为 a1~an。每轮可以选择一堆石子，取掉任意颗石子，但不能不取。谁先不能操作，谁就输了。
- 在游戏开始前， B 可以扔掉若干堆石子，但是必须保证扔掉的堆数是 d 的倍数，且不能扔掉所有石子。A 先手，请问 B 有多少种扔的方式，使得 B 能够获胜。
- n ≤ 500000, 1 ≤ d ≤ 10, 1 ≤ ai ≤ 10^6, m ≤ 10^7。
- 时间限制3s，空间限制64M。

Nim z utrudnieniem

- B 获胜的充要条件为剩下的石子每堆数量的异或和为 0。
- 设 f(i,j,k) 表示考虑了前 i 堆的石子，当前扔掉的堆数模 d 为 j，没有扔掉的石子的异或和为 k 的方案数，转移显然。
- 最后注意特判 n 是 d 的倍数的情况，此时答案应该减去 1。
- 时间复杂度 O(ndmax(ai))，不能承受。

Nim z utrudnieniem

- 将石子数从小到大排序，那么有效状态数降为 O(md)，可以承受。
- 即使用滚动数组优化，空间使用也达到了大约 80MB，不能承受。
- 注意到 f(i,...,k) 和 f(i,...,k xor ai) 的转移是互补的，于是可以同时处理。
- 空间使用大约为 40MB，可以承受。

Shopping

- 给定一棵有 n 个点的树，第 i 个点有 di 件商品，价格为 ci，价值为 wi。
- 你手头有 m 块钱，且你要保证你买过的点在树上互相连通，问买到的物品的总价值最多是多少。
- $1 \le n \le 500$, $1 \le m \le 4000$, $di \le 100$。

Shopping

- 枚举一个点必选，以它为根求出dfs序，在dfs序上dp。
- f[i][j]表示后i个点花j块钱的最大价值，如果不选就跳过整个子树。
- 多重背包可以用单调队列优化至O(nm)。
- 时间复杂度O(n^2m)。
- 点分治，每次考虑重心必选的方案。
- 时间复杂度O(nmlogn)

Druzyny

- 体育课上，n 个小朋友排成一行 (从 1 到 n 编号)，老师想把他们分成若干组，每一组都包含编号连续的一段小朋友，每个小朋友属于且仅属于一个组。
- 第 i 个小朋友希望它所在的组的人数不多于 di，不少于 ci，否则他就会不满意。在所有小朋友都满意的前提下，求可以分成的组的数目的最大值，以及有多少种分组方案能达到最大值。
- $1 \leq n \leq 1000000$。

Druzyny

- 设 fi 为将前 i 个小朋友分组的最优解，则 fi = max(fj + 1)，其中 1 ≤ j < i，且满足max(cj+1,cj+2,...,ci) ≤ i - j ≤ min(dj+1,dj+2...di)
- 设 gi = min(j)，且 i - j ≤ min(dj+1,dj+2,...,di)，容易发现 gi单调不下降，可以通过双指针以及维护线段树在 O(n log n)的时间内预处理出来。
- 那么对于 [gi, i - 1] 中的 j，只需要满足 c 的限制即可。

Druzyny

- 考虑通过分治优化 DP。
- 在处理l~r时，求出 [l + 1, r] 中 c 最大的位置，设为 k，以k 为分界线可以递归处理 l~k-1 和 k~r。现在只需用 [l, k - 1] 的决策更新 [k, r]。
- 由于 ck 最大，所以 ck ≤ i - j， i 从 max(ck + l, k) 开始，决策 j 一开始的取值范围为 [max(l, gi), i - ck]，此时用线段树求出区间最大值即可。
- 先考虑l>gi的情况，每当 i 往右移一位时， j 的上限也往右移一位，可以做到O(1) 更新。

Druzyny

- l ≤ gi ≤ k – 1时，由于所有更新 i 的区间 [l, k - 1] 均不相交，所以只存在一个区间[l, k - 1] 满足 l ≤ gi ≤ k - 1，直接线段树查询区间内的最优解即可。
- gi>k-1时没有转移，直接结束即可。
- 当 i 循环到 k + ck 时，[k + ck, r] 内所有 i 的可行决策 j 的上限都为 k - 1，而g值按以上情况是分成三段的，二分找到分界点，对于gi<l的i转移是一样的，用线段树进行区间更新即可。
- 时间复杂度 O(n log n)。