

Solution

T1

来源 UCUP

n 较小的情况可以搜索或者状压解决，期望可以得到一些暴力分。

2^{n^2} 做法根本没啥出头啊，没有 $n = 4$ 但是 $n = 5$ 又跑不动的。

一般遇到这种选出一些东西最小化，要求满足一些东西的问题，一种常见的想法是往 dp 上面靠。

即设 f_j 表示当前满足局面是 j 的情况，最小的代价是多少。

但是这个 j 对应的状态是不是有点太多了？如果要记下整个棋盘每个格子又没有被覆盖，虽然说跑不满，但依然有着巨大的复杂度。

我们从尝试简化 j 的状态入手，即我们试图找到一个较简便的状态，使得只要这个状态被满足，那么整个棋盘就被覆盖，以及如果整个棋盘被覆盖一定有这个状态被满足，**换句话说就是找到棋盘被覆盖的一些简单的充要条件。**

接下来选手可以通过在这一步上的推导来不断优化复杂度，从而得到不同的分数。

我们直接给出关键结论：**棋盘被覆盖的充要条件是棋盘的四个角落被覆盖。**

证明较简单，此处不再赘述。

有了这个结论，我们便可以这样设计 DP：

设 $f_{i,j,0/1,0/1,0/1,0/1}$ ，表示目前 DP 到第 (i, j) 的格子，并且用 4 个 0/1 变量表示四个角落被覆盖的状态，此时的最小代价。

转移是简单的。

可以使用状态压缩来优化后面 4 个 0/1 变量的存储。

注意最终答案在 `int` 范围内（直接选中间的不就好了），但是转移过程如果实现的不够精细可能会有爆 `int` 的风险。

实际上得出了这个结论后还能得出多种做法：

std 的实现是：对于某个格子，按照它能覆盖的角落进行分区，可以将棋盘分成九个区，每个区显然只会取一个权值最小的，然后 2^9 搜索。

不清楚是否有一些乱搞做法通过。

注意可能需要特判 $n = 1$ ，否则可能出现正确性的问题，或者 TLE（test22 几乎全是 $n = 1$ ，如果此时在跑的话复杂度就变成 $O(T \times 2^4)$ 甚至 $O(T \times 2^9)$ 了，再算上读入上的开销，这其实较为危险）。

T2

原创，但是缝合怪题。

题目背景借鉴了唐场。

对于 $n, a_i \leq 9$ 的部分分，可以考虑搜索出所有可能的情况，复杂度 $O(n^n)$ 。

不难发现， x 左右两边是独立的，因为左边的人分数始终不超过 a_x ，右边的人分数始终不低于 a_x ，因此可以考虑将问题拆成 x 左边和 x 右边，然后用乘法原理计算答案。

在 $x = \lfloor \frac{1+n}{2} \rfloor$ 的情况，不难发现搜索的规模可以减半。

我们接下来单独考虑 x 分别在左右侧的情况，也就是 $x = 1$ 和 $x = n$ 的部分分。

考虑 $x = 1$ ，也就是要求 x 右侧的不降序列，再减去一个不增序列，仍然是一个不降序列，且要求最终序列第一个元素不小于 a_x 。

不难发现，在这种情况下，如果最终序列第一个元素不小于 a_x 的要求被满足，后面的元素依然还是不降的（后面更大的元素，减去了一个更小的数，显然还是不会比你小）。

令 $len = n - x, d = a_{x+1} - a_x$ ，那么问题变成了数长度为 len ，值域在 $[0, d]$ 的序列个数。

这个很经典，答案为 $\binom{d+len}{len}$ 。

考虑 $x = n$ 咋做。

也就是要求一个不降序列，减去一个不降序列还是不降序列。

这个限制就没有那么简单了。

但是还是可以考虑 dp 并且使用前缀和优化做到 $O(nd)$ 。

设现在被减掉的序列是 b ，那么 b 序列合法的必要条件是。

$$\forall i \in [1, x), a_i - b_i \geq a_{i-1} - b_{i-1}$$

(不妨假设, $a_0 = b_0 = 0$)。

移项得到：

$$\forall i \in [1, x), a_i - a_{i-1} \geq b_i - b_{i-1}$$

不难发现这个条件也是充分的。

令 $a'_i = a_i - a_{i-1}$, $b'_i = b_i - b_{i-1}$ ，那么最终形式就是：

$$\forall i \in [1, x), a'_i \geq b'_i$$

注意到 b 和 b' 一一对应，因此只需要算出 b' 的方案数，就可以算出 b 的方案数，而 b' 的方案数显然是：

$$\prod_{i=1}^{x-1} (a'_i + 1)$$

直接算即可。

最后只需要将前面部分和后面部分拼起来，就可以得到答案。

你会发现这个题可能没什么意思。

是的。

这个题最有意思的地方，就在于，题目背景来源于生活。。。。

嗯

T3

来源 CEOI。

最暴力的暴力就是 $O((S!)^n)$ 枚举所有情况，显然可以通过子任务 1。

考虑 $S = 2$ 且满足每个颜色的球都是偶数怎么做。

(显然可以先把两个球颜色相同的行扔掉)

对于每种颜色考虑一下，就是它所在的行中，恰好有一半这个球在第一列，另一半这个球在第二列。

这个问题是不是非常眼熟？

我们考虑**使用图论方法**解决上面的问题。

对于一个包含 u, v 两个球的行，连接一条 u 到 v 的无向边。

然后给每条边定向，如果是 $u \rightarrow v$ 相当于是一行 u 在第一列 v 在第二列，反之亦然。

问题变成给定一张无向图，每个点度数都是偶数，求一种定向方案，使得每个点入度等于出度。

欧拉回路，按照路径的方向定向即可。

如果某种颜色的球出现了奇数怎么办？

就是相当于上面那个图不是欧拉图，但是要求变成入度出度差不超过 1。

建立一个超级源点，令所有奇数度数的点向这个源点连边。

然后图就变成了欧拉图，然后跑欧拉回路定向，因为每个点最多有一条额外的边，那么剩下的边就还是满足入度出度不超过 1。

考虑 S 更大怎么做。

你注意到这里 $S = 2^k$ ，那么不妨来考虑一些跟这个有关的算法，比如分治，倍增之类的。

考虑分治。

如果在每种颜色出现数量是 S 倍数的时候，我们可以把这些颜色的球一半扔到左边，一半扔到右边，然后递归地往下做，最后总能找到答案。

如果不是 S 的倍数，容易发现这样做还是合法的。

那么这个问题的形式就十分类似于 $S = 2$ 的部分。

发现我们不能再用一条边来表示某个行的状态。

考虑将边拆点。

考虑对于每一行，建立一个点，这个点向其包含的所有颜色连边（某个颜色有多就连多条）。

然后调整奇度点，欧拉回路定向，如果一条边是从行对应的点连向颜色对应的点，那么把这个颜色的一个球扔到左边，否则扔到右边。

这样就可以单次在线性与问题大小的复杂度解决将颜色分成两部分的问题，然后递归分治下去，复杂度最终是 $O(ns \log(s))$ 的。

注意不要把数组开小，以及单次分治不要把复杂度写错，必须是严格关于当前分治问题规模的大小的线性（或者带个 \log ），也就是说不能每次分治带上 $O(T)$ 的复杂度。

T4

诶来源还是 CF

CF 难度 3500，大家，嗯，懂得都懂。

下面我们记对 $[l, r]$ 区间跳跃一次后的区间为 $f(l, r)$ ，跳跃 k 次后的区间为 $f^k(l, r)$ 。

首先显然可以对着题目模拟，即每次询问都模拟这个跳跃的过程，使用 st 表优化 RMQ（或者更加暴力一点反正数据范围很小直接 n^2 预处理）就可以做到复杂度 $O(nq)$ 。

跳跃的次数其实并不是 $O(n^2)$ 而是 $O(n)$ 的，这里稍微感性理解一下？

考虑 n 很小但是 q 很大怎么办。

首先假装 $f(1, n) = [1, n]$ ，否则除了 $[1, n]$ 区间答案是 0 外其他都不可达。

这时候区间个数被限制到了很少。

考虑将每个区间向其一次跳跃后的区间连一条有向边。这形成了每个点出度恰好为 1 的图，也就是**内向基环树森林**。其中 $[1, n]$ 这个区间连的是一个自环。

那么询问相当于是求每个区间到 $[1, n]$ 的距离，如果这个区间不和 $[1, n]$ 在一个连通块则无法到达。

如果你对基环树的处理足够炉火纯青，那么应该不难得到 $O(n^2 + q)$ 的做法，但是这样写起来略显繁琐。

考虑这一类问题常用的套路：**倍增**。

即我们尝试处理出所有 $f^{2^k}(l, r)$ ，询问时，从大到小枚举 k ，如果 $f^{2^k}(l, r) \neq [1, n]$ ，则我们就跳 2^k 次到达一个新的区间，不断这样操作，如果最后能到达 $[1, n]$ 则返回，否则说明无解。

预处理 $f^{2^k}(l, r)$ 时，有 $f^{2^k}(l, r) = f^{2^{k-1}}(f^{2^{k-1}}(l, r))$ ，从小到大枚举 k 即可。

因此总复杂度变为 $O((n^2 + q) \log n)$ ，但是在处理上比基环树舒服不少。

当 n 很大的时候，我们倍增就需要处理 $O(n^2)$ 个区间，而 $\log n^2 = 2 \log n$ 因此我们不需要在意 \log 上的常数而考虑怎么简化倍增的信息。

然后你发现不会做。

所以让我们看看出题人有没有给一些特殊性质吧！

仔细阅读一下特殊性质发现也没做法.....

发现区间交非空这个东西很奇怪，他严格强于 $l_i = 1$ ，但是出题人一没给 $l_i = 1$ 的部分，二 $l_i = 1$ 也不是特别好做。

思考一下区间交非空有啥用，一个经典的套路是如果满足这个条件，则必然存在一个点 k ，使得所有区间都可以表示为 $[l, k] \cup [k, r]$ 的形式。

仔细思考你会发现： $\forall l \leq k \leq r, f(l, r) = f(l, k) \cup f(k, r)$ 。

证明是简单的，考虑 $[l, k]$ 和 $[k, r]$ 中，要么有一个区间同时包含了 $[l, r]$ 的最大最小值，这个时候显然成立。要么其中一个包含了最大值，一个包含了最小值。又因为他们都包含了 k 这个点，因此最终他们跳跃一次并起来也是 $f(l, r)$ 。

这个结论更强的形式是：令 $[l, r] = [l_1, r_1] \cup [l_2, r_2]$ ，若 $[l_1, r_1] \cap [l_2, r_2] \neq \emptyset$ ，则 $f(l, r) = f(l_1, r_1) \cup f(l_2, r_2)$ 。

这有什么用呢？

进一步你发现，在 $[l, r]$ 跳跃一次后， $f(l, r)$ 和 $f(l_1, r_1), f(l_2, r_2)$ 依然满足上述结论的适用条件！

也就是归纳一下，可以得到：

令 $[l, r] = [l_1, r_1] \cup [l_2, r_2]$ ，若 $[l_1, r_1] \cap [l_2, r_2] \neq \emptyset$ ，则
 $\forall k \geq 1, f^k(l, r) = f^k(l_1, r_1) \cup f^k(l_2, r_2)$ 。

只有这个你会发现你还是不会做。

有了上述结论，我们不妨继续细分区间。

即我们显然有 $f^k(l, r) = f^k(l, r-1) \cup f^k(r-1, r)$ ，再不断划分 $(l, r-1)$ ，可以得到：

$$f^k(l, r) = \bigcup_{i=l}^{r-1} f^k(i, i+1)$$

这说明了，我们倍增的信息都可以用所有长度为二的区间表出来。

注意到上面那个是一个区间并的形式，考虑使用 st 表或者线段树维护这个倍增的过程。

首先枚举 k ，假设我们已经知道了所有 $f^{2^{k-1}}(i, i+1)$ ，现在要进行递推
 $f^{2^k}(i, i+1) = f^{2^{k-1}}(f^{2^{k-1}}(i, i+1))$ ，直接把 $f^{2^{k-1}}(i, i+1)$ 拉到 st 表或线段树上做一个区间查询即可，我们是已经知道 $f^{2^{k-1}}$ 的所有信息的。

查询时，从大到小枚举 k ，扔到线段树或 st 表上询问 f^{2^k} 满不满足条件，如果满足就直接跳，否则缩小 k 继续枚举，和之前暴力倍增做法完全相同。

如果使用线段树维护，则时间复杂度 $O((n+q)\log^2 n)$ ，空间复杂度 $O(n\log n)$ 。

如果使用 st 表维护，则时间复杂度 $O(n\log^2 n + q\log n)$ ，空间复杂度 $O(n\log^2 n)$ 。

这两种写法应该都能通过，总不能有人写猫树吧。

注意上面所说的处理 $f(1, n) \neq [1, n]$ 的情况，以及处理跳跃后出现 $l = r$ 的情况。

放这个题最大的难点在于，好像除了知道正解的性质外完全没有办法设计出来一个能做的部分分。

以及如果你知道了正解的性质，那么做出来会超级快，能够拉写暴力的不少分（吗

预估 1= 线**不高于** 212 分，队线**不低于** 344 分。

GL & HF