MSTI2020 信心赛

Day2 solution

数据恢复

题目来源：codeforces gym 292796 J
正如题目所说，因为时间来不及，所以这题是搬的。

算法 1（图是一条链）

显然选择方法只有一种，因此直接模拟即可。

算法 2（$n \le 20$）

不清楚搜索算法可不可以通过，但是没有任何剪枝应该是不行的。
状压 dp 是可以通过的。
但是不知道神奇的贪心能不能通过，因为数据可能不是很强。

算法 3（菊花图）

第一次一定选 1 号点，然后剩下的 $n-1$ 个点就没有依赖关系了。

不难证明，当所有点按照 $\frac{a}{b}$ 从小到大排序后选取最优。

算法 4（全部子任务）

考虑拓展算法 3 的做法，我们记 $v_i = \frac{a_i}{b_i}$，那么我们贪心的选择 v_i 最小的。

此时分两种情况，一种是 i 的依赖已经被恢复了，那么直接恢复 i 一定是最优的。
而如果 i 的依赖没有被恢复，那么在 i 的依赖被恢复后直接恢复 i 一定是最优的。
因此我们可以把 i 和 f_i 进行一个捆绑，也就是把 i 和 f_i 合并成一个新的点，这个点按照 $f_i - i$ 的顺序包含了这两个点。而新的这个点的 a 和 b 是原来两个点的值的和。因此新点的 v 也可以计算了。
如果把这个新的点代替原来两个点，重新建树，那么新问题可原问题是等价的。
因此只需要不断重复这个操作，直到所有点都合并成一个点即可。
找依赖关系可以使用并查集维护，找最小值可以用堆维护。
时间复杂度：$O(n \log n)$

注： CF 官方题解中所谓的 $O(n)$ 并没有考虑堆的效率，所以时间复杂度还是 $O(n \log n)$ 的。

下落的小球

算法 1（$n \le 10$）

枚举并且按照题意模拟即可。

算法 2（全部子任务）

先来考虑一个简单的问题，如何判断方案是否存在。

我们记 s_i 为点 i 的子树大小，b_i 为子树中所有叶子的 a_i 的和。

我们定义剩余量 $r_i = b_i - s_i$。显然，如果在每一步操作后，都有 $\forall i, r_i \ge 0$，那么该方案合法。

以上判断方式启发我们对 r_i 进行思考。

我们认为 x_i 为使点 i 从有球变成没球的那次操作。

如果考虑从子树往根做，对于点 i，我们先假设子树 i 内的 b_i 次操作直接的相对顺序已经排好了。此时我们想要计算 f_i 的方案。

对于点 i，我们能发现 x_i 为子树中那 b_i 次操作第 $r_i + 1$ 次操作。

换句话说，子树 i 中那 b_i 次操作，前 r_i 次操作并不会让对子树 i 中点的状态受影响，而第 $r_i + 1$ 次操作让 i 从有球变成没球，而之后的操作只会改变子树 i 中点的状态。

因为后 s_i 次操作只会影响子树中的状态，所以子树之间这部分的操作可以随意合并。

而前 r_i 次操作都相当于更改子树外的，所以这一部分可以任意合并。

因为在子树外是所有操作都操作完后才能操作子树内的，所有前面那部分和后面那部分的顺序是不能更改的。

因此这题的解法很明显了，对于每一个子树的操作，可以把前 r_i 次当且 A 部分，把后 s_i 次当成 B 部分。子树合并时，A 部分与 A 部分合并，B 部分与 B 部分合并，然后把合并完的 A 部分接在 B 部分前面。

至于计数，只需要用到一个式子：对于一个相对顺序确定的长度为 x 的序列和一个相对顺序确定的长度为 y 的序列，它们合并的方案数为 C_{x+y}^{x}。

消失的运算符

算法 1（$k = 0$）

白送 20 分？对！
直接把所有数字乘起来，得到的就是答案。

算法 2（$m \leq 20$）

先把多余的括号去掉（事实上数据很良心，多余的括号不去也不会超时）。
然后枚举每一个符号是加号还是乘号。
然后就是表达式求值问题了。

算法 3（没有括号）

设 $F[i][j]$ 表示前 i 个数字，已经有 j 个加法。可以直接 DP，加个前缀和优化即可。

当然还有一个对正解有启发作用的解法。可以设 $F[i][j]$ 表示前 i 个数字，已经有 j 个加法，不包括最后一段乘法的表达式的值， $G[i][j]$ 表示最后一段乘法的值。
这两个 DP 互相转移即可。

算法 4（全部子任务）

建出树，然后开始树上背包。
子树合并用算法 3 的第二种做法。

古老的序列问题

算法 1（$s_i \le s_{i+1}$）

对于所有区间 $[l,r]$，最小值一定是 s_l，最大值一定是 s_r。所以这就变成简单的计数题了。

算法 2（$s_i \le 5$）

计算 $mi = a, mx = b$ 的区间数比较困难，那就计算 $mi \ge a, mx \le b$ 的区间数，最后只需要减一下就好了。

由于值域只有 $[1,5]$，所以可以直接枚举 a, b 然后求解。

算法 3

枚举右端点，维护每个左端点的答案。

从有往左维护两个单调栈，一个存最大值，一个存最小值。

左端点答案的维护可以使用线段树，存的值可能会比较多。

算法 4

先考虑 $m = 1$ 的情况。

考虑分治，每一次求左端点在 $[l, mid]$，右端点在 $[mid+1, r]$ 的答案的和。

枚举左端点，那么右端点可以根据最小值和最大值由谁决定分成三段。每一段可以用前缀和。

对于多个询问，则可以把总询问拆分成若干个小询问。每一个小询问类似从左区间在 $[l', mid]$，右区间在 $[mid+1, r']$ 的所有区间的答案（$l \le l' \le mid < r' \le r$）。

在分治时使用树状数组可以解决这个问题。