JSOI2019 Round2 评讲

AceSrc

AceSrc@outlook.com

April 28, 2019

Overview

1. T1 predict
2. T2 neural
3. T3 celebrate

题目简析

- 用 (t, x) 来表示命题"t 时刻第 i 个人是生存状态".
- 用 $(t, \bar{x})$ 来表示命题"t 时刻第 i 个人是死亡状态".

题目预测相当于给定若干蕴含式 (implication) $A \to B$ (当且仅当 A 为真命题且 B 为假命题时 $A \to B$ 为假命题).

- 难兄难弟 $0,\ t,\ x,\ y$: $(t, \bar{x}) \to (t+1, \bar{y})$, $(t+1, y) \to (t, \bar{x})$
- 死神来了 $1,\ t,\ x,\ y$: $(t, x) \to (t, \bar{y})$, $(t, y) \to (t, \bar{x})$

同时题目本身自带了一些蕴含式.

- $(t, \bar{x}) \to (t+1, \bar{x})$
- $(t+1, x) \to (t, x)$

将 (t, x) 看成点, 所有蕴含式 $A \to B$ 看成 A 到 B 的有向边, 我们得到了 $2SAT$ 问题中常见的有对称性的有向图.
(对称性指若 $(t_1, x) \to^* (t_2, y)$, 就一定有 $(t_2, \bar{y}) \to^* (t_1, \bar{x})$)

充要条件

同时这个图还有一些性质, 首先这是个 DAG.
其次只有 $(*, x) \to^* (*, \bar{y})$, 不会有 $(*, \bar{x}) \to^* (*, y)$
$Live(x, y) = 1$ 当且仅当满足以下条件

- $(T + 1, x) \not\to^* (T + 1, \bar{x})$.
- $(T + 1, y) \not\to^* (T + 1, \bar{y})$.
- $(T + 1, x) \not\to^* (T + 1, \bar{y})$.

直观做法

直接 BFS - 30

就按上面说的条件，对于第 i 个人，从 $(T+1,i)$ 出发，看能到达哪些 $(T+1,\bar{y})$

离散化后 BFS - 40/50

只考虑那些被预言提到的点.
该做法，由于这里机房比较快，但虚拟机又比较慢，加上各位的码力不同综合一下有一定分数的误差.

简易 bitset - 60

每个预言会产生 4 个关键点，一共 $4m$ 个关键点，对这 $4m$ 个点建 DAG，用每个关键点分配一个 n 位的 bitset，记录第 y 位记录能到达 $(T+1,\bar{y})$.
注意内存使用 $4mn/8$ bytes，非常吃紧.

不那么的直观做法

稍微优化一下 bitset - 80

如果一个点 (t, x) 只能到达 $(t-1, x)$，那 (t, x) 和 $(t-1, x)$ 没有任何区别，可以合并.
如果一个点 $(t, \bar{x})$ 只能到达 $(t+1, \bar{x})$，$(t, \bar{x})$ 和 $(t+1, \bar{x})$ 也没任何区别，可以合并.
然后 $(T+1, \bar{x})$ 这些点也不需要分配 bitset .
把点合并完之后，去掉 $(T+1, \bar{x})$ 所在的点集，在一共会剩下 $2m+n$ 个点集，只给这些点集分配 bitset 即可.
注意内存使用 $(2m+n)n/8$ bytes，稍微不那么吃紧.

题解做法

使劲优化 bitset - 100

在上面做法的基础上，我们考虑拓扑排序，每次选择出度为 0 的点进行拓扑.
维护一个 bitset 池.
当有若干出度为 0 的点，优先选择形如 $(t,\bar{x})$ 的点，如果有一个点 $(t_1,*)$ 指向 $(t,\bar{x})$，且还未拥有一个 bitset，我们从 bitset 池里掏出一个 bitset 给它用.
当处理完 $(t,\bar{x})$ 之后，如果 (t,x) 还未拥有 bitset，那么我们把 $(t,\bar{x})$ 拥有的 bitset 给 (t,x) 用，否则把 $(t,\bar{x})$ 拥有的 bitset 丢回 bitset 池.
可以证明这样子时刻只有 m 个关键点拥有 *bitset*.
因为如果 $(t,\bar{x})$ 是一个关键点，那么一定对应一条形如 $0,t,x,*$ 的预言，如果一个点 (t,y) 的 bitset 是拓扑 $(t,\bar{x})$ 时得到的，那么一定对应一条形如 $1,t,y,x$ 的预言.
仔细思考之后发现对 (t,x) 这样的点我们可以按 t 从小到大选择，这样需要 n 个 bitset 作为辅助.
内存使用 $(m+n)n/8$ bytes，低于内存限制了.

题目简析

大体思路是，首先得计算出，第 i 棵树剖分成 j 条链的代价 $f(i,j)$.
在我们确定每棵树的链划分的情况下，计算哈密顿回路个数.
假设现在有三棵树 A, B, C, 第一棵树分成 3 条链 A_1, A_2, A_3, 第二棵树分成两条链 B_1, B_2, 第三棵树分成两条链 C_1, C_2.
在这种情况下，我们需要计算有多少 $A_1, A_2, A_3, B_1, B_2, C_1, C_1$ 的合法排列.
合法指第一个元素为 A_1, 且相邻两个元素不能来自同一棵树，且最后一个元素不能来自树 A.
比如 $A_1, B_1, A_2, C_1, A_3, C_2, B_2$ 是合法的.
比如 $A_1, A_2, B_1, C_1, A_3, C_2, B_2$ 是不合法的.
如果排列中第 i 个元素和第 $i-1$ 个元素来自同一棵树，那么称第 i 个元素是非法元素，比如上面的 A_2.
对非法元素的个数进行容斥.

Part 1

给定一棵树 T, 要计算将这棵树分成 j 条链的方案数, 是一个典型的树形 DP(树上背包), 可以在 $O(n^2)$ 内计算出来.
注意, 每条链是有方向的, $u \rightarrow v$ 和 $v \rightarrow u$ 是不同的方案, 因为从别的树跳到 u 再走到 v 和从别的树跳到 v 再走到 u 属于不同的哈密顿回路. 所以做背包的时候每条点数大于 1 的链答案要乘以 2.
还有一个点也算一条链, 不用乘 2.

Part 2

假设没有"第一个元素为第一棵树的第一条链，最后一个元素不能来自第一棵树"的限制.（等价于求解哈密顿路径个数.）
容斥部分，最简单的想法是，记 $g(i,j,k)$ 为考虑前 i 棵树，总共分出了 j 个自由组，非法元素的个数至少是 k 的方案数.
如何表示至少有 k 个非法元素？用类似错排问题的做法，进行捆绑.
比如我们现在枚举第 x 棵树要分成 y 条链，我们想让这 y 条链生成至少 z 个非法元素，那么我们可以把 y 个数分成 $y-z$ 类自由组，每一类捆绑起来，表示在排列中该类总是排在一起的.
这和第一类斯特林数有点类似，但和第一类斯特林数的区别在于该问题是将 y 个元素分成 $y-z$ 个排列，而不是圆排列.
令 $S(i,j)$ 是将 i 个元素分成 j 个排列的方案数，递归式是：
$S(i,j) = S(i-1,j-1) + (i-1+j)S(i-1,j)$

Part 2

g 最简单的方程是
$g(i, j+y-z, k+z) += g(i-1, j, k) * f(i, y) * S(y, y-z)$
最后我们结果是 $\sum_j j! \times (\sum_{2|k} f(m, j, k) - \sum_{2|k+1} f(m, j, k)))$.
可以看出我们没必要记录 k, 我们只需要记录 k 的奇偶性.
这样就得到了一个 $O(n^3)$ 的做法.

Part 2

修改之后再观察下方程:
$g(i, j+y-z, (k+z) \bmod 2) += g(i-1, j, k \bmod 2) * f(i, y) * S(y, y-z)$
注意我们并不需要枚举 y 和 z, 我们只需要枚举 $y-z$.
对于每个 $y-z$ 以及 $z \bmod 2$, 先对 $f(i, y) * S(y, y-z)$ 求和得到 $C(i, y-z, z \bmod 2)$.
上面式子就变成 $g(i, j+(y-z), (k \bmod 2 + z \bmod 2) \bmod 2) += g(i-1, j, k \bmod 2) * C(i, y-z, z \bmod 2)$.
这样就是 $O(n^2)$ 的了.

最后我们把" 第一个元素为第一棵树的第一条链, 最后一个元素不能来自第一棵树" 限制加上, 改写成" 最后一个自由组为第 m 棵树的第一条链所在的组, 第一个自由组不能来自第 m 棵树", 由于回路的性质, 这是一一对应的.

那么只要在 DP 最后一棵树的时候稍作改变就可以了.

题目简析

AceSrc 曾这样说

这大概是省选历史上最简单的 T3 了.

** 大学的 *sb 曾这样说

我已经预见到知乎" 如何评价 JSOI2019 Round 2 T3 被花式拿分" 的回答了.

题意很清晰, 对一个串的每个前缀求最小循环表示的开始位置, 如果有多个, 输出最小的.
暴力分拿满是 30, 即对每个前缀线性求最小表示, 或者后缀结构也可以.
各种花样后缀结构或者贪心视常数大概能有 $50/60$.
后缀数组的 $O(n \log n)$ 算法视常数可能会 T 最后几个点.
标程复杂度 $O(n \log n)$, 算法 z-algorithm, 就是俗称的扩展 KMP.

性质 1

假设我们现在考虑前缀 $S[1..k]$, 我们考虑有哪些位置可能称为 $S[1..k]$ 最小循环表示的开始位置, 我们称这些位置为候选点, 我们需要使用一些性质减少候选点的个数.
设 n 是字符串长度.

性质 1

假设两个位置 $i < j$. 如果 $common - prefix(S[i..n], S[j..n]) \le k - j$, 那么 i, j 之间肯定有一个不是候选点.

假设你已经利用这个性质, 得到 $S[1..k-1]$ 的候选点集 P, 对于两个数 $i \in P, j \in P, i < j$, 肯定有 $common - prefix(S[i..n], S[j..n]) > (k-1) - j$, 我们现在想知道是否有 $common - prefix(S[i..n], S[j..n]) > k - j$, 那么我们只需要比较 $S[i + k - j]$ 和 $S[k]$ 即可.
用这个性质大概能有 $50/60$ 分.

性质 2

还是假设我们现在考虑前缀 $S[1..k]$.

性质 1

对于两个点 $i < j$, 假设 $common - prefix(S[i..n], S[j..n]]) > k - j$, 如果有 $k - j \geq j - i$, 那么 j 不是候选点.

这个性质是有最小循环表示的某个性质得来的, 假设串 $S = S_1 S_1 S_2$, 其中 S_1, S_2 是子串.
要么有 $S_1 S_1 S_2 < S_1 S_2 S1 < S_2 S_1 S_1$,
要么有 $S_1 S_1 S_2 > S_1 S_2 S1 > S_2 S_1 S_1$
如果有 $k - j \geq j - i$, 那么 $S[1..k]$ 就肯定形如 $ABBC$, 其中 A, B, C 是子串.
这个性质直接保证了, 候选点个数不会超过 $\log n$ 个.

解决!

对于候选点, 我们直接使用 z-algorithm 就能比较循环同构串的字典序大小.
如果用后缀数组做这一步, 有可能超时.

The End