CCPC Finals 2021 题解

	命题思路	测试数据&标程	代码验题	英文题面
A Mash	sfiction	sfiction	tun,Riatre,KeyID,yfhuang	chenjb
B Shuttle Bus	AkaiLemon,tun	AkaiLemon	tun	AkaiLemon
C Selection Sort Count	tun	tun	liyang21,KeyID	chenjb
D Tree Partition	liyang21,tun,t90tank	liyang21	tun	chenjb
E Elegant Tetris	xpchf	xpchf	AkaiLemon,tun,Gromah,yfhuang	Riatre
F Modulo	yang12138	yang12138	AkaiLemon,tun,Riatre,Gromah,chenjb	AkaiLemon
G Integer Game	AkaiLemon	AkaiLemon	yang12138,tun,abc473848880,xyz2606	AkaiLemon
H Harie Programming Contest	tun	tun	rowdark	Riatre
I Reverse LIS	tun	tun	chenjb	Riatre
J jwfw.harie.edu	tun	tun	yang12138,Riatre,KeyID,xyz2606	Riatre
K Security Plan	AkaiLemon	AkaiLemon	xpchf,tun,Gromah,yfhuang	AkaiLemon
L Paid Leave	tun	tun	sfiction,abc473848880,yfhuang	chenjb

负责人&组题: tun

polygon&pintia题目配置及教学: sfiction

口头验题: rowdark,t90tank,apiadu,xpchf,liyang21,AkaiLemon,sfiction,yang12138,_noname

验题赛: （第一场）KeyID,Gromah,abc473848880 （第二场）chenjb,xyz2606,yfhuang

组织对接工作: 狗哥

参与讨论: rourou

A. Mash

题意：

定义了一种含两种指令的语言，求执行前k条指令后的输出结果。

题解 (by @tun)：

按题意模拟，注意直接模拟会导致队列Q中的指令达到O(n^2)级别，考虑到我们最多只需要执行k个指令，因此当Q中的指令超过k个时我们就不需要继续往里添加了。时间复杂度O(n + k)。

B. Shuttle Bus

题意：

给定一棵树上每个点的职工数，现在需要选一个点作为工厂新址，并可以设置最多k条从工厂出发的班车线路，每位职工会走到最近的班车经过的点坐班车上班，求每一个点作为选址时最优班车设置下的最少总步行距离。

题解 (by @AkaiLemon)：

先考虑有根树的情况，假设t为根的话，在没有班车的情况，其它节点上的员工就需要步行到t，将总花费转化成每条边经过的人数之和，那么每条边的权值就等于它对应的子树的节点权值之和；问题就转化成求出k条从t出发的链，使其覆盖的边的权值之和最大（重复覆盖只算一次），答案就是总的移动距离减去这个最大权值和。

这里可以贪心地选择从t出发的最大一条链，忽略这条链上的边的基础上选择次大的链，直到选满k条。这个过程可以用一次dfs完成，同时用适当的数据结构维护前k大的链（标程采用两个multiset维护，也可以考虑用平衡树）。回到这个题，只需要在此基础上进行第二次dfs换根，换根的时候要注意边的移动方向发生了改变，对应的权值也需要变化。这里我们可以发现每次换根只有常数条路径的权值和发生了变化。时间复杂度O(n * log(n))。

C. Selection Sort Count

题意：

给出选择排序每轮的交换次数，求有多少种排列符合给出的交换次数。

题解 (by @KeyID)：

首先给出一个性质：如果一个位置上的数字在第i轮被交换过，且它不是i，那么它在第i+1轮一定也会被交换。

证明：首先模拟一下交换的过程，我们可以知道在一轮交换中一个位置参与交换的充要条件是，在此轮交换开始的时候，这个位置上的数是前缀最小值。在第i轮交换中一共有c[i] + 1个位置参与了交换，设这些位置从小到大为p_0, p_1, ..., p_{c[i]},在这一轮交换前，这些位置上的数字为a_0, a_1, ..., a_{c[i]}。模拟交换的过程可以发现，a_{c[i]}（也是全局最小值i）被交换到了最前面，从此不再参与任何交换。其他p_i上的数字都被移动到了p_{i+1}。因为在移动之前，a_{i+1}是a_i之后的下一个前缀最小值，这说明在(p_i, p_{i+1})之间没有数字比a_i更小，因此a_i移动到p_{i+1}之后，它仍然是一个前缀最小值。

有了上述性质之后，我们考虑从第n轮开始往回倒推初始的状态。假设我们已经知道了第i+1轮是哪些位置参与了交换，第i轮的交换位置就是此次新增的首位置i，以及c[i]个在第i+1轮参与了交换的位置。我们可以从c[i+1] + 1个位置中任意选出c[i]个位置，根据交换过程，可以发现它们有唯一的合法方案：第1个位置上的数字移动到新增的首位置i，后面每个位置上的数都移动到前一个位置，新增的数字i放置到最后一个位置。因此假设我们已知了到第i+1轮时合法方案数为dp[i+1]，那么第i轮的方案数就是dp[i+1] * C(c[i+1] + 1, c[i])。时间复杂度O(n)。

D. Tree Partition

题意：

给定一棵n个节点的树, 树上的点按照1到n标号。要求删除树上的k条边将树划分成k+1连通块, 满足每个连通块中的节点标号所构成的集合都为一段连续的整数，求方案数。

题解 (by @liyang21)：

取1号点为根，将树上的所有边都定向成从儿子指向父亲的有向边。在逻辑上新建0号点以及一条从1号点指向0号点的边。这样由于树的性质，当某个点集只有一条指向点集外的边时，这个点集就构成一个子树。

到点标号的数轴上考虑问题，因为题目要求子树的标号是连续区间，问题也就变成了要将[1， n]这个区间切成k个子区间，满足每一个子区间都有且只有一条指向区间外边的方案数。

用 f[i][j] 表示当前位于 i 号点, 已经划出了 j 个区间的方案数。可以写出一个比较暴力的转移方程：

f[i][j] = sigma{f[k][j – 1] (k + 1 <= i 且 区间(k, i]仅有一条指向外部的边)}

考虑根据区间上边的分布来做一些优化。将从较小编号的点指向较大编号点的边称为后向边，否则称为前向边。

先考虑所有可能跨越右边界i指向外侧的后向边， 把所有出发点小于等于i的后向边按出发点标号大到小排序为 x1, x2, ...

则对于 k >= x1 的转移点，区间 (k, i] 没有指向区间外的后向边, x1 > k >= x2 时 区间 (k, i] 恰有一条指向区间外的后向边。

接着考虑所有可能跨越左边界k+1指向外侧的前向边，任意一条起始点小于等于i的前向边 x –> y (i >= x > y) 都会使 x > k >= y 的区间 (k, i] 拥有一条指向外部的前向边。

最终能转移到i的转移点k共有以下两种:

1. 位于 [x1, i] 且仅有一条后向边的转移点k
2. 位于 [x2, x1) 且没有后向边的转移点k

只要能对拥有0或1条前向边的下标k做快速的区间求和就可以高效的完成dp转移。

位置k对应的前向边数可以随着i的扩大时时维护，让每一条前向边 x –> y (x > y) 在 i = x 时加入，每次加入都是将区间 [y, i – 1] 内转移点的前向边数加一，对当前位置i而言也就是一段后缀的处理。

由于前向边数只增，且某个位置的前向边数大于1后，就不用再维护，所以处理区间加时，可以暴力的遍历每一个点，由于最多被处理两次，整体而言总数不超过2n次。又由于每次操作都是删除末尾/末尾添加，所以只需要对0/1条前向边分别开一个数组就可以O(1)的维护好它们的前缀和，支持快速的区间求和。

取值区间 x1, x2 的维护比较简单，可以用一个堆，随着i的扩大时时维护，让每一条边 x –> y (x < y) 在 i=x 时入堆 i=y 时退出。查询时取堆中的最大值和次大值即可。

时间复杂度O(nk + n * log(n))。

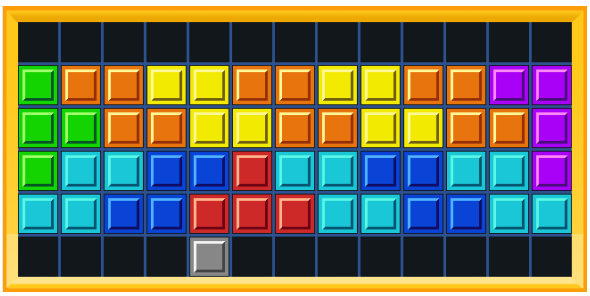
E. Elegant Tetris

题意：

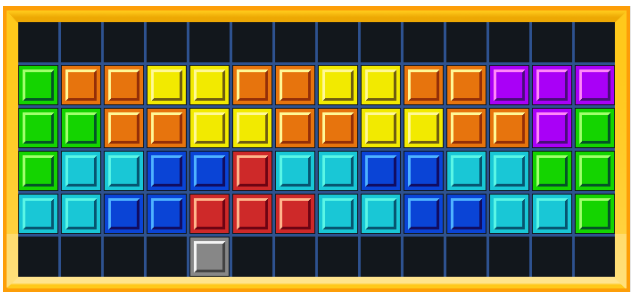
构造一个非空俄罗斯方块操作序列使得操作后的局面与初始局面一致。

题解 (by @Gromah@tun)：

考虑找到最顶上的一个方块，然后在这个方块的上面构造几个整行；首先放上一个倒T作为地基，再在左右分别用S和Z扩展，接着在左侧放一个竖T，再在上层用Z铺满，最后再用一个L（宽度奇数）或两个T（宽度偶数）处理右侧的空余，如下图所示：



宽度为奇数的情况



宽度为偶数的情况

其中当宽为偶数且最顶上方块在最右列时，我们需要将偶数的构造方案左右翻转一下。另外，操作的时候要注意方块的顺序，错误的顺序可能会导致行被提前消空。

F. Module

题意：

给定x和数组a，你可以对a进行重新排列，并依次执行 x %= a[i]，求所有可能的排列中操作后最大的x。

题解 (by @Gromah)：

注意到在取模过程中，对x真正造成影响的模数一定是单调递减的，因为当x对一个数p进行取模之后，结果一定小于p，之后再去模那些大于等于p的数的话这个结果一定是不变的。所以可以把所有数字排序，然后 O(2^n) 枚举哪些模数被用上，其中最小的模数一定是会被用上的，从所有情况的结果取最大值即可。

G. Integer Game

题意：

给定大于1的正整数p和n个正整数集合，第i个集合Si初始包含 Li 到 Ri 的所有整数。两位玩家轮流操作，每次操作选择一个非空的集合Si, 并在集合中挑选一个数x，使得 x * p > max(Si) ，然后将这个集合中大于等于x的数都删掉。删掉所有集合中最后一个数字的人获胜，问先手是否必胜。

题解1 (by @xyz2606)：

首先每个集合可以独立考虑，因为多个集合组成的游戏的sg值就是每个集合各自的sg值的异或。其次每个集合初始都是区间，可以证明每个集合一直是区间。然后计算区间[l,r]的sg值。

令[l,l−1]表示空集，则 l−1<=r<=l*p的时候，由于每次可以删掉区间任意后缀（即形如[?,r]的子区间），sg值为r−l+1。可以归纳证明当r为kp+1(k>=l)时，[l,r]的sg等于[l,k−1]的sg， 除此之外的r按从小到大的顺序sg值依次递增1。r<=l*p时命题成立。当r为kp+1(k>=l)时，[l,r]可以到达[l,k]，[l,k+1]，...，[l,kp]这些状态。由归纳假设，对于所有的[l,j] (j<k)，[l, j*p+p+1]的sg值和[l,j]的sg值相同。j*p+p+1在k和kp之间，除了j=k−1的情况。那么在计算[l,r]的sg值时，可以认为[l,r]能到达[l,j]， 其中l−1<=j<=k−2。即[l,r]可以到达任意[l,x] (l−1<=x<=r)，x=k−1除外。x小于k−1时，[l,x]的sg值取遍了所有小于[l,k−1]sg值的非负整数。x大于等于k（且不超过kp）时，[l,x]的sg值也不等于[l,k−1]的sg值，因为[l,x]可以转移到[l,k−1]。这就证明了[l,r]=[l,kp+1]的sg值等于[l,k−1]的sg值。

对于另一种情况，我们依次考虑r=kp+2,...，kp+p。当r=kp+2时，[l,r]能转移到的状态是所有[l,r−1]能转移到的状态再加上[l,r−1]。即可以认为[l,r]能转移到任意[l,x]，包括刚才[l,r−1]=[l,kp+1]不能转移到的[l,k−1]，因为[l,r−1]的sg值等于[l,k−1]的sg值。那么显然[l,r]的sg值是所有[l,x]的sg值之最大值再加1。r=kp+3，...，kp+p时，[l,r]能转移到的状态每次都比[l,r−1]能转移到的状态多一个[l,r−1]，所以每次sg值都是[l,r−1]的sg值再加1。

题解2 (by @tun)：

实际比赛的时候我们可以考虑打表找规律。比如固定p=2，然后分别枚举l和r打一张二维表，这样就比较容易观察到规律了：

```
 1  2  0  3  1  4  2  5  0  6  3  7  1  8  4  9  2 10  5 11  0 12  6 13  3 14  7 15  1 16  8 17
    1  2  3  0  4  1  5  2  6  3  7  0  8  4  9  1 10  5 11  2 12  6 13  3 14  7 15  0 16  8 17
       1  2  3  4  0  5  1  6  2  7  3  8  4  9  0 10  5 11  1 12  6 13  2 14  7 15  3 16  8 17
          1  2  3  4  5  0  6  1  7  2  8  3  9  4 10  5 11  0 12  6 13  1 14  7 15  2 16  8 17
             1  2  3  4  5  6  0  7  1  8  2  9  3 10  4 11  5 12  6 13  0 14  7 15  1 16  8 17
                1  2  3  4  5  6  7  0  8  1  9  2 10  3 11  4 12  5 13  6 14  7 15  0 16  8 17
                   1  2  3  4  5  6  7  8  0  9  1 10  2 11  3 12  4 13  5 14  6 15  7 16  8 17
                      1  2  3  4  5  6  7  8  9  0 10  1 11  2 12  3 13  4 14  5 15  6 16  7 17
                         1  2  3  4  5  6  7  8  9 10  0 11  1 12  2 13  3 14  4 15  5 16  6 17
                            1  2  3  4  5  6  7  8  9 10 11  0 12  1 13  2 14  3 15  4 16  5 17
                               1  2  3  4  5  6  7  8  9 10 11 12  0 13  1 14  2 15  3 16  4 17
                                  1  2  3  4  5  6  7  8  9 10 11 12 13  0 14  1 15  2 16  3 17
                                     1  2  3  4  5  6  7  8  9 10 11 12 13 14  0 15  1 16  2 17
                                        1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  0 16  1 17
                                           1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16  0 17
                                              1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
                                                 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
                                                    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
                                                       1  2  3  4  5  6  7  8  9 10 11 12 13 14
                                                          1  2  3  4  5  6  7  8  9 10 11 12 13
                                                             1  2  3  4  5  6  7  8  9 10 11 12
                                                                1  2  3  4  5  6  7  8  9 10 11
                                                                   1  2  3  4  5  6  7  8  9 10
                                                                      1  2  3  4  5  6  7  8  9
                                                                         1  2  3  4  5  6  7  8
                                                                            1  2  3  4  5  6  7
                                                                               1  2  3  4  5  6
                                                                                  1  2  3  4  5
                                                                                     1  2  3  4
                                                                                        1  2  3
                                                                                           1  2
                                                                                              1
```

H. Harie Programming Contest

题意：

求给定的二分图中有多少个点对满足删掉这两个点后最大匹配数不变。

题解 (by @tun)：

首先建图跑最大流。

观察残余网络，现在我们考虑如何才能在不改变流量的情况下删掉一个点，以连接源点S的左侧点u为例：

1. 当边(S, u)残余容量大于0，则直接删除边(S, u)
2. 当边(S, u)残余容量等于0，则先找到一个包含边(u, S)的增广环进行增广，让边(S, u)恢复容量，然后再删除边(S, u)

可以发现，左侧点u能否被删除等价于是否存在一条从S到边(S，u)或边(u，S)的增广路；方便起见，我们可以将边(S，u)拆点变成边(S，u')和边(u'，u)，于是现在只需要判断是否存在一条从S到虚点u'的增广路。类似的，右侧点只需要判断是否存在从T到u'的“反向增广路”即可。

现在讨论点对的两种情况：

1. 异侧点：我们可以分别求出S可达的左侧虚点个数和T走反向边可达的右侧虚点个数，乘起来即为异侧删点的方案数（由于网络被割分成两部分，两侧的删点互不影响）。
2. 同侧点：以左侧点对u和v为例，是否可以从S到u'和v'分别进行一次增广等价于是否存在两条没有公共边的路径(S，u')和(S，v')。接下来，我们可以对剩下所有未拆点的边也进行拆点，并以S为起点求支配树。存在这样的两条路径 当且仅当 这两个虚点在支配树上的所有公共祖先均不是拆边的虚点（可以参考Codechef GRAPHCNT题的证明）。最后，点对的计数只需要在支配树上简单统计一下即可。

以上方法需要把点对分成异侧点、左同侧点和右同侧点三类情况讨论。实际上我们可以将连接汇点的残余网络的边反向后接在源点上，这样只需要对源点求一次支配树即可算出所有情况。时间复杂度O(m * sqrt(n) + m * log(m))。

I. Reverse LIS

题意：

给定一个01串s，你可以对其做不超过k次翻转子串操作，求所有可能的操作结果串的最长不下降子序列的长度最长是多少。

题解1 (by @t90tank@tun)：

首先我们将01串s的'0'赋权值+1，给'1'赋权值−1，权值数组记为w，设最长不下降子序列的'0'/'1'切换点为i（<=i的位置只取'0'，>i的位置只取'1'），那么这个子序列的长度可以表示为 count(s，'1') + sum(w[1, i])；于是求最长不下降子序列等价于求max(sum(w[1, i]))，即w数组的最大前缀和。

回到原问题，在子串翻转k次后的串中选出一个前缀，相当于在翻转前的原串中选出互不相交的一个前缀和k段区间。现在问题变成求权值和最大的k+1段不相交区间，其中有一段为前缀。

我们可以贪心求这个值：首先选取一个权值和最大的前缀，并将前缀中的所有权值取反，然后每次选取一个权值和最大的子区间，同样也将权值取反；这里求权值和最大的子区间可以用线段树快速维护。时间复杂度O(n * log(n))。

题解2 (by @tun)：

另一种方法是把问题转换成在一个数组中选出k个不相邻的元素使得总和最小，然后用链表+堆来模拟费用流。（可以参考BZOJ1150的做法）

J. jwfw.harie.edu

题意：

小测试有10道选择题，给出若干次尝试的选项和总得分，求符合情况的正确答案的方案数。

题解 (by @xyz2606)：

枚举前5道题的答案是什么，算出每次作答在前5题上分别得了多少分，得分组成一个长度为n的vector，然后我们将结果记为v[a]，a表示前5题的答案（例如 v["ABACD"]={10,20,0,10,40,20,20}）。对后5题的每种答案也算一个vector，记为u[b]。当v[a]+u[b]（向量加）等于输入的vector（记为w）时，(a,b)是一组可行的答案。对于每个b，计算w-u[b]，统计有多少v[a]=w-u[b]即可。统计时可以用哈希或trie，注意32位整数的单哈希是不够的。

K. Security Plan

题意：

一个仓库可以划分为 n 行 m 列个单元格，标记左上角 (1, 1)，右下角 (n, m)；现在需要在仓库中选一些单元格装监控，每一个监控可以分别选定一个终点，监控该点和终点为端点的矩形范围的单元格；规定每一个单元格只能装一个监控。当每一个单元格都被至少一个矩形覆盖时，方案被称为完美方案。如果一个完美的监控方案中任何一个监控被移走，无论如何修改终点，都无法满足完美监控的条件，那么就称这个方案是个极小的完美监控方案。求有多少种不同的极小的完美监控方案。两个方案只要选取的单元格的集合相同，即为相同方案，不管监控设置的参数。

题解 (by @AkaiLemon)：

可以证明，最多只需要4个监控，即可覆盖所有单元格。假设任选的四个单元格分别为(x1，y1)，(x2, y2), (x3, y3) 和 (x4, y4)，不妨设x1 <= x2 <= x3 <= x4，参数可以设定为(n, *), (n, *), (1, *) 和 (1, *)，对于(n, *) 和 (n, *)这两组参数，可以根据y1和y2的大小一个设置为1，一个设置为m，这样就覆盖了以(x2，1)为左上角，(n，m)为右下角的整个子矩形；同理对于(x3,y3)和(x4，y4)也可以让其覆盖以(1, 1)为左上角，(x3, m)为右下角的子矩形；这样合并就覆盖了所有单元格；

在此基础上，考虑选取1，2，3，4个单元格的极小完美方案：

1. 选取1个：只能是在4个角选择（n=1 或 m=1需要注意少于4个，特别地，n=1并且m=1时只有1种选择）；
2. 选取2个：只需要在对边上且非4个角上任选两个单元格，即可完美覆盖；并且可以证明，不存在其它方案可以只选2个就满足。
3. 选取3个（邻边各选一个，中间选一个）：邻边上的两个点可以覆盖到3个角，只要让两个矩形都覆盖住中间的点，中间的点去覆盖最后一个角即可完成;
4. 选取3个（边上选一个，中间选两个）：不妨设边上的点为 (n, y)，中间两个点(x1, y1) 和(x2, y2)，不妨设y1 <= y2，只需要满足y2 <= y+1，或者 y1 >= y-1。正面计算会比较复杂，可以考虑反向计算，先在中间任选2个点，然后排除不符合的情况：y1 < y-1 && y2 > y+1；由于n和m的值较大，此处需要将式子展开，转成求平方和的形式；
5. 选取4个：只需要在中间任选4个即可；在中间任选4个一定是极小的， 因为只要少了任意一个，就一定会有一个角没被覆盖到。而如果选取的单元格有一个在边上，中间3个一定可以选取两个使得满足4的情况；

把这5种情况盘算清楚，注意下一些边界情况就可以了。

L. Paid Leave

题意：

给出n天的工作与休息安排，问最少请几天假，使得：

1. 任意一段连续工作的天数不超过x
2. 任意一个单休日前后的连续工作天数之和不超过y

题解 (by @liyang21)：

容易发现，在满足x，y限制的情况下，尽可能的将休假安排在往后的位置能够最大程度的利于将来的规划，是一定不劣的休假策略。

具体实现时，从前往后的遍历每一个原有节假日，维护一个值表示最近一个休息日前的连续工作天数days。

根据这个天数和 x, y 的大小关系可以分两种情况讨论，每一种情况都是反复某个策略进行休假：

1. y − days <= x：后续的安排为 [(工作y−days天) (休1) (工作days天) (休1)] 循环
2. y − days > x：后续的安排为 [(工作x天) (休1) (工作y−x天) (休1)] 循环

所以可以O(1)快速算出两个原有节假日之间需要额外休假几天，同时维护好连续工作天数days，整体时间复杂度O(m)。