



東北大學



第47届国际大学生程序设计竞赛沈阳站 解题报告

东北大学出题组

A Absolute Difference

首先不考虑某个集合之中所有区间均退化的情况，那么实际上就是从集合A中选取一个区间 $S_1 = [l_1, r_1]$ ，从集合B中选取一个区间 $S_2 = [l_2, r_2]$ ，然后计算 $\int_{l_1}^{r_1} \int_{l_2}^{r_2} |x - y| dx dy$ 加入到最终答案中。

如果 S_1 与 S_2 不(严格)相交，那么可以直接拆绝对值独立对 x 和 y 进行积分，推出公式后可以使用前缀和优化计算。否则可以分别将 S_1 与 S_2 分成相交部分和不相交部分，得到1对相同区间和至多8对不相交区间进行计算，由于单个集合内部的区间都是不交的，相交的 S_1 与 S_2 只有 $O(n+m)$ 对，可以直接计算。

对于某个集合之中所有区间均退化的情况，相当于从集合中若干个单点之中等概率随机选取一个，一种处理方法是在预处理时根据是否所有区间均退化对区间赋予一个1或者是区间长度的权值，这样可以复用不退化时的计算逻辑以减少代码量。

B Binary Substrings

首先分析不同子串的个数，由于长度为 i 的01串至多有 $\min(2^i, n - i + 1)$ 个，因此不同子串的个数不超过 $\sum_{i=1}^n \min(2^i, n - i + 1)$ ，特判 $n = 1$ ，然后选取正整数 k 使得 $2^k + k - 1 \leq n < 2^{k+1} + (k + 1) - 1$ ，那么需要使所有 2^k 种长为 k 的01串都出现过，并且所有长为 $k + 1$ 的子串都互不相同。

当 $n = 2^k + k - 1$ 时就是构造de Bruijn序列，构造方法是构建一张 2^{k-1} 个点的有向图，点从0到 $2^{k-1} - 1$ 编号，每个点表示一个作为相邻两个长为 k 的串公共部分的长为 $k - 1$ 的串，对每个串分别向在结尾新增0或者1得到的新串的长为 $k - 1$ 的后缀连有向边。这个图就是de Bruijn图，可以证明这个图上存在欧拉回路，求出一条欧拉回路就能构造出一个合法方案。

当 $n > 2^k + k - 1$ 时，需要扩展上述序列，先在 2^{k-1} 的图上找一条欧拉回路，这也是 2^k 的de Bruijn图上的哈密顿回路，将哈密顿回路上从0连出去的那条边断开就能得到一条哈密顿路径，沿着这条哈密顿路径可以遍历所有长为 k 的子串恰好一次。在 2^k 的de Bruijn图上删去这条路径之后从0出发找一条欧拉路径来扩展之前的路径，最好的情况是除了 $2^k - 1$ 的自环之外的边都能经过，这样就能构造出长为 $2^{k+1} + (k + 1) - 2$ 的序列，截取长为 n 的前缀即可。值得注意的是，在删边之后原图可能分成了若干个弱连通分量从而得不到理论最长的序列，但是已经能求解部分 n 比较小的情况。

考虑使用另一个算法来尝试求解 n 比较大的情况，直接在 2^k 的de Bruijn图上跑一条欧拉回路，将0作为第一个出现的长为 k 的串来构造序列，找出包含所有长为 k 的本质不同子串的最短前缀，那么对于更大的 n 也是截取长为 n 的前缀即可。

在上述两个算法中求解欧拉回路时引入随机性（例如随机起点和随机出边的顺序）之后进行多次独立重复求解，不断贴近第一个算法能求解的最大 n 与第二个算法能求解的最小 n ，期待最终能对特定的 k 求解所有满足 $2^k + k - 1 \leq n < 2^{k+1} + (k + 1) - 1$ 的 n 。由于不同的 k 只有 $O(\log n)$ 个，可以在本地验证不同随机数种子的运行效率以及最终得到的方案是否合法。

C Clamped Sequence

- 这个题做法很多，标程的做法是把每一个 $|a[i]-a[i+1]|$ 根据 l 的取值拆成一个分段函数，其中分段函数中的每一段都是线性的，所以可以把所有 $|a[i]-a[i+1]|$ 的所有线性函数段提取出来，然后用扫描线去维护当前所有线性函数段的一次项和常数，再在这个过程中把最优值统计出来即可，这样子的复杂度就是 $O(n \log n)$ 的。



東北大學

D DRX vs. T1

- 签到题





東北大學



E Graph Completing

首先对原图进行边双连通分量缩点，由于原图是连通的，缩点之后会得到一棵树。

现在加一条非树边 (u, v) 会对 u 到 v 的树上路径进行覆盖，当所有树边都被覆盖的时候就得到了一个边双连通图。

考虑对树边是否被覆盖的情况进行容斥，枚举一定不会被覆盖的边集，删去这些边之后树上包含若干个连通块，只能在每个连通块内部任意连非树边。

使用动态规划进行计算这个容斥过程， $dp_{u,i}$ 表示以 u 为根的子树的连通块大小是 i 时子树内的方案数，转移时考虑一个子树 v 是否连上来，如果要连上那么可以在两个连通块之间任意连边并合并连通块，时间复杂度看上去是 $O(n^3)$ ，但是转移时第二维最多枚举到子树大小，可以证明时间复杂度是 $O(n^2)$ 。



東北大學



F Half Mixed

首先，总的子矩阵个数是 $\frac{n(n+1)}{2} \times \frac{m(m+1)}{2}$ ，然后如果这个数字是奇数，那么显然是无解的，否则 $\frac{n(n+1)}{2}$ 和 $\frac{m(m+1)}{2}$ 至少有一个是偶数，不妨设 $\frac{m(m+1)}{2}$ 是偶数，那么我们让每一列的所有行都是相同的数字就可以把问题转化成 $1 \times m$ 版本的了。

然后对于 $1 \times m$ 版本的问题，首先总的子矩阵个数，其实也就是子区间个数，是 $\frac{m(m+1)}{2}$ ，然后把这个序列拆成若干数字相同的段，设每一段的长度分别是 l_1, l_2, \dots, l_k ，那么 pure 的子区间个数就是 $\sum_{i=1}^k \frac{l_i(l_i+1)}{2}$ ，所以本质上只需要找到一个方案使得满足一下两个条件即可：

- $\sum_{i=1}^k l_i = m$
- $\sum_{i=1}^k \frac{l_i(l_i+1)}{2} = \frac{m(m+1)}{4}$

这就可以转化成一个背包问题，但其实只需要贪心，每次取尽可能大的 l_i 去凑这个 $\frac{m(m+1)}{4}$ 的和就可以了。



東北大學



G Meet in the Middle

第一棵树上的点 u 到 v 的距离记为 $dis_1(u, v)$ ，同理第二棵树上的点 u 到 v 的距离记为 $dis_2(u, v)$ 。

考虑对询问离线，在第一棵树上dfs询问点 a ，然后假设在第二棵树上每个点 j 挂一个新点 j' 且边权为 $dis_1(a, i)$ ，问题转化为在第二棵树加点得到的新树上查询 b 的最远点，这一定是新树上的任意一条树直径的两端点之一。

考虑如何维护新树的直径，对第一棵树的dfs序建线段树，每个线段树节点维护第一棵树上dfs序在区间内的所有点 i 对应第二棵树上的点集直径，但是不具体维护直径长度，当 a 在第一棵树上移动时，在某个dfs序区间内点 i 对应的 $dis_1(a, i)$ 会整体加减一个值，区间外也会整体加减一个值，只需要沿着dfs区间在线段树上走的同时触发区间点集直径的更新即可，未触发更新的区间只是区间内 $dis_1(a, i)$ 整体加减一个值并不会改变直径，对于查询 (a, b) 只需要查询新树上的直径然后枚举端点 j' 用对应的 j 计算 $dis_1(a, j) + dis_2(b, j)$ 并更新答案即可。

使用欧拉dfs序和ST表支持LCA查询进而支持树上路径长度查询，时间复杂度是 $O(n \log n + q)$ 。



H P-P-Palindrome

- 本题要求选出两个回文子串，拼接后仍然是回文子串。结合回文串的重要性质：回文串的border也一定是回文串，加以简单分析可知：能拼接成PPP回文串的两个回文子串P和Q一定需要具有相同的最小循环节。进而由于回文串的循环节一定是border，所以一定也是回文串。进一步考虑：假设一个回文串得最小循环节是p（注意p一定也是回文串），并循环了k次，表示为 $S = p^k$ ，那么所有的p, p^2 , p^3 , ..., p^k 一定都是合法的回文子串。因此本题得解法是使用manacher或者pam等方法提取所有本质不同的回文子串，然后使用hash等方法按照回文串长度从小到大以此求出他们的最小循环节，即可统计答案。

I Quartz Collection

可以看成每堆石头先选的获得 $a_i - b_i$ ，后选的获得0。

那么对于 $a_i - b_i$ 大于0的都想先选， $a_i - b_i$ 小于0的都想后选。以及都想先选 $a_i - b_i$ 更大的那些石子堆。

对于 $a_i - b_i > 0$ 的，肯定是先后手依次按照 $a_i - b_i$ 排序选择。

对于 $a_i - b_i < 0$ 的，肯定都想把后手选的先选了。

那么对于 $a_i - b_i > 0$ 的，按照模4，为1和0的A先选，否则B先选。

$a_i - b_i < 0$ 的，按照模2，为1的A先选，否则B先选。

那么可以用值域线段树来维护修改即可。



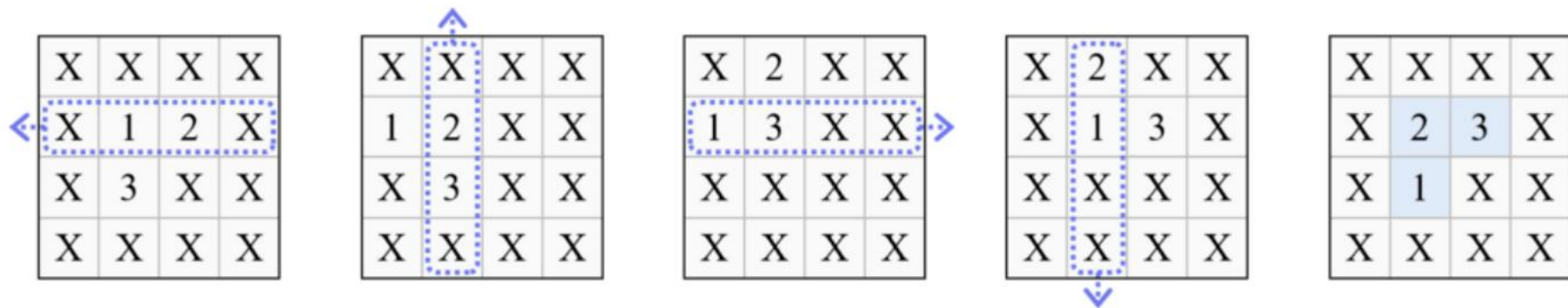
東北大學



J Referee Without Red

按行的置换能拆分成若干个轮换，因此可以交换列使同个轮换的列连续地排成一组，这样置换等价于每组内同时循环移位，按列的置换同理。设行列分别有 n', m' 组，考虑新矩阵由此划分出的 $n' \times m'$ 个矩形盘面。从大小为 1 的组入手，其涉及的盘面长或宽为 1，想产生不同方案只能朝单一方向循环移位。单个盘面的答案等于最小周期，不难用 KMP 算法求解（本题的置换较特殊，直接枚举周期复杂度亦正确）。而整个方案数等于所涉及盘面的 LCM。

对于长宽均大于 1 的盘面，称所有数互异的为“互异盘面”，反之为“非互异盘面”。注意到：第一，各盘面包含的数永远进不到其他盘面。第二，多次置换能实现逆元，因此循环移位有上下左右四种方向。第三，定义互异盘面的奇偶性为所有数以固定方式生成序列（如从上到下顺次拼接）的逆序对奇偶性，显然交换任意两个数都会改变奇偶性，因此一次循环移位改变奇偶性当且仅当其移了偶数个数。于是我们合理推测，互异盘面能在不变动其他盘面的条件下得到任意奇偶性不变的排列方案，非互异盘面能直接得到任意排列方案。证明只需如图实现 L 形三数轮换（L 形方向、轮换方向取决于操作顺序），逐个复位直至剩下两个数即可。



总体上我们先通过若干次循环移位锁定各互异盘面的奇偶性，再调整各盘面内部的方案。若长或宽为偶数，互异盘面的奇偶性可变，但碍于循环移位必须同步，同组行列盘面的奇偶性可能受之影响。例如，大小为奇数的组，其与大小为偶数的组交叉形成的盘面只能同时改变奇偶性，方案数为 2。再如，给大小为偶数的组分配虚点，它们交叉形成的互异盘面为相应的虚点连边，这些盘面的奇偶性方案数即等于生成森林边数的二次幂。结合乘法原理，我们分别统计全局的奇偶性方案数，并用组合相关知识计算各盘面内部的方案数乘出最终答案，时间复杂度能做到 $O(nm)$ 。

K Security at Museums

问题即为选取至少两个简单多边形的顶点，使得选取的顶点之间两两连线都完全落在简单多边形内部（含边界），可以发现由于简单多边形是单连通区域，选取的顶点构成的凸包内部（不含边界）不能有其他顶点。

首先预处理出任意两个顶点间连线是否在简单多边形内部，如果连线与某条边严格有交，那么一定不在多边形内部，否则在连线内部找一个简单多边形上的顶点递归划分连线直到连线内部没有任何顶点，此时只需要检查连线中点是否在简单多边形内部，记忆化之后这部分的时间复杂度是 $O(n^3)$ 。

然后枚举选取的顶点之中坐标 (x, y) 字典序最小的，不考虑字典序更小的点，将两两之间的合法连线按照极角序排序，按照极角序不减的顺序选一些连线首位尾相连之后可以得到一个凸包，对应的凸包顶点就是一组可行的方案。由于存在共线的情况，为了避免来回走重复计算的情况，要求要么只在一个极角上选边，要么至少选出三种不同极角的边。

这样在事先对所有边进行极角排序之后，每次枚举一个起点 s ，记 $dp_{i,j}$ 表示从 s 出发沿着 i 种极角的边走到了 j ，这里 i 只需要记录到3，然后根据边的极角序做分组背包即可，需要注意同一组边的枚举顺序以避免漏算多次沿组内的边走的方案，这部分的时间复杂度也是 $O(n^3)$ 。

L Tavern Chess

- 本题为标准的酒馆战旗规则，为了减少复杂度，增加了所有随从都是水桶腰身材（初始生命等于攻击）的设定。如此一来，任何两个随从对撞，一定会死掉一个，也可能两个一起死。于是可以直接使用简单的搜索回溯遍历每种局面即可。复杂度不超过 $(7!)^2$ 。本题定位为简单铜牌题。



東北大學



M Vulpecula

显然每个结点建立线性基表示能异或出的数集，然后大概要搞出所有树上 d 级邻域的线性基交。看似信息量爆炸，稍作分析发现：对于同个结点，随着 d 的增大，线性基交的张成只会不断取子空间且最多变化 $\log W$ 次。但瓶颈在于市面上的求交算法很难有效维护每次剥离的基。

相比之下其孪生问题容易得多：树上邻域的线性基并。对于各基底，我们额外维护 d 至少多大才能并入它（这一技巧也被称作"线性基的可持久化"）。若基底能直接被当前结点异或出，该值便为 0，否则使用树上 BFS（从外到根，再从根到外）更新相邻结点的线性基。如此我们在 $O(n \log^2 W)$ 的时间复杂度内得到了每个点 $O(\log W)$ 个线性基每次并入的基。对所有线性基查询最大值，总时间复杂度仍是 $O(n \log^2 W)$ 。

有方法将"剥离"转换成"并入"吗？回想异或相关算法，似乎能在 FWT 上找门路。不妨引入集合幂级数描述线性基，其第 x 位为 1/0 取决于 x 是/否在线性空间内。

两线性基 A, B 并的幂级数 C ，等于各自幂级数 xor 卷积再除以同个基数相关值后的结果，记作 $C = \frac{1}{k}(A \oplus B)$ ，亦作

$\text{FWT}(C) = \frac{1}{k}(\text{FWT}(A) * \text{FWT}(B))$ （ $*$ 即点值乘法）。注意到 $\text{FWT}(A)$ 描述的实际是其正交补（证明见 codeforces 1336E2 题解），令表示该正交补的线性基为正交线性基 A^\perp ，则点值乘法实际上计算了 A^\perp, B^\perp 的交。反过来，多个线性基交等价于多个正交线性基并，于是变成了孪生问题。

然而求正交线性基须 $O(\log^2 W)$ ，查询时我们不可能把每个邻域的线性基都正交回来。因此，和原线性基相反，在正交线性基中我们把每个基都存在最低位，这样利用 $x \in A, y \in A^\perp \Rightarrow |x \& y| \bmod 2 = 0$ 能直接在正交意义下贪心，时间复杂度 $O(n \log^2 W)$ 。