Notes

A. Lily

签.

把旁边没有 L 的 . 变成 C.

C. Array Concatenation

字符串倍增找规律.

每次的操作有两种:

1. copy 现有的 n 个数放在后面,
2. copy 现有的 n 个数放在后面, 并且把前面原有的 n 个数翻转.

求在取模之后的最大所有前缀和之和.

可以发现, 一但进行了一次第二种操作, 之后无论哪一种操作都一样了.

这里就可以得到一种做法, 枚举何时进行第二种操作进行暴力计算.

进一步找规律可以发现, 一旦进行了第二种操作, 无论何时进行, 得到的结果是一样的, 所以只需要计算两种情形就行.

接着只需要找到每一次倍增后所有前缀和之和怎么计算, 需要维护哪一些量即可.

E. Draw a triangle

exgcd. 给定两个点, 求第三个点使得构成的三角形面子最小.

设给定点 A, B, 需要计算 C, 不妨设 $\overrightarrow{AB} = (x, y), \overrightarrow{AC} = (u, v)$, 那么三角形 ABC 的面积就是 $\frac{1}{2}|xv - yu|$.

要求三角形最小的面积实际上就是最小化 $|xv + y(-u)|$, 这就是一个裸的 exgcd, 注意符号.

G. Group Homework

换根 dp. 选两条链, 使在这两条链上出现一次的点权值和最大.

首先分出两大类, 一类是两条链不交, 另一类则是两条链有交集.

先说第二类, 如果两条链有交集, 则至多交集是一个点, 否则可以优化成这种情形使权值和更大. (结论题?)
那么现在需要求的是从每一个点出发的四条最长链, 可以换根 dp.

第一遍 dfs 可以求出子树中最长的四条链.
第二遍 dfs 处理来着父亲的链, 我只能想到用 $\log n$ 的办法.

```
// 从 u 转移到 u 的儿子 v //
set<pair<int, int>> chains;
chains.insert(longest_chain_from_father[u], u);

for (v is the son of u) {
```

```
    chains.insert(longest_chain_in_subtree[v], v);
}

for (v is the son of u) {
    chains.erase(longest_chain_in_subtree[v], v);

    longest_chain_from_father[v] <- max value in set;

    chains.insert(longest_chain_in_subtree[v], v);
    dfs(v, u);
}
```

至于结果只需要在每个点的子树中最长的四个链和来着父亲的一共五条链中取最长的四条.

再说第一类, 貌似是古早的 codeforces 中的原, 一道 2600 的 dp?
依旧是换根 dp, 需要考虑断开 $u - v$ 之间的连边, 上下两棵树的直径之和.

第一遍 dfs 可以处理出子树中的最长链和子树的直径.
第二遍 dfs 考虑如何换根, v 为根的下面那一棵树的部分已经处理完了, 只需要考虑上面那一棵树新加入点 u 该如何变化.

实际上多加入的点 u 只有三种可能:

1. 不作为上面那一棵树的直径,
2. 作为上面那一棵树的直径的一个端点,
3. 作为上面那一棵树的直径的中间的某一个点.

```
// 从 u 转移到 u 的儿子 v //
set<pair<int, int>> best_set, longest_chain_set;
best_set.insert(best, u);
longest_chain_set.insert(longest, u);

for (v is the son of u) {
    best_set.insert(diamater_of_subtree[v], v);
    longest_chain_set.insert(longest_chain_in_subtree[v], v);
}

for (v is the son of u) {
    best_set.erase(diamater_of_subtree[v], v);
    longest_chain_set.erase(longest_chain_in_subtree[v], v);

    case1 <- max value in best_set; // 直径不过 u //
    case2 <- the largest value in longest_chain_set + the second largest value in
longest_chain_set + w[u]; // 直径经过 u //

    best_set.erase(diamater_of_subtree[v], v);
    longest_chain_set.erase(longest_chain_in_subtree[v], v);

    dfs(v, u, max(case1, case2), the largest value in longest_chain_set + w[u]);
```

```
}
```

J. Permutation Puzzle

拓排加贪心. 给定一些位置的初值以及一些位置对的大小关系, 构造一个 permutation.

给出的 (u_i, v_i) 表示 u_i 上的数小于 v_i, 可以建立一条 u_i 到 v_i 的边, 题目保证这个图是一个 DAG.

很容易想到拓扑, 可以算出每一个位置上是数的下界, 同理, 构造一个反向的图可以得到每一个位置上的数的上界.

现在的问题就变成了每一个位置都要满足 $p_i \in [l_i, r_i] (i = 1, 2, \cdots, n)$ 以及一些拓扑关系.

如果不考虑拓排关系, 从 1 到 n 分别考虑, 将 l_i 不超过当前值的位置挑出来, 在其中选择 r_i 最小的位置填上当前值总是合理的. (结论题?)

这个策略放到存在需要满足的拓扑关心的问题中依旧是合理的, 这只需要用优先队列维护即可, 无解就是某一个时刻填不下去, 存在两种情形:

1. 某一个时刻不存在 l_i 不超过当前值的位置,
2. 将 l_i 不超过当前值的位置挑出来后, 其中最小的 r_i 小于当前值.

K. Barrel Theory

离谱构造. 给你数的数量和总和, 要求构造出给定数量且和为给的值的数, 满足所有数的异或和小于所有数的最小值.

(结论题? 很多离谱结论?)

首先 $n = 1, 2$ 的情形可以结论和暴力, 接下来对 n, m 分奇偶考虑.

最直接的想法, 尽可能把 n 个数是异或和变成 0 或 1.

1. n, m 都是偶数.
   只需要填 $1, 1, \cdots, 1, \frac{m-n+2}{2}, \frac{m-n+2}{2}$, 这时异或和为 0, 最小值为 1.
2. n 为奇数, m 为偶数.
   先填写 $n - 3$ (偶数) 个 1, 再考虑后面三个数怎么填.
3. 如果 $k = m - n + 3$ (偶数) 的二进制中至少有两个 1, 可以考虑填写 $\frac{k}{2}, \frac{k}{2} - \text{lowbit}\left(\frac{k}{2}\right), \text{lowbit}\left(\frac{k}{2}\right)$.

例如

```
          11001,
110010 -> 11000,
              1
```

这样异或和仍是 0.

2. 否则 $k = 2^p$, 这里再分情况:
   1. $n \geqslant 5$, 可以将前面任意两个 1 变成 2, 接着 k 的二进制中自然就有两个 1, 可以像之前一样考虑, 这样异或和仍是 0.
      2. $n = 3$, 这样无法调整 k 的值, 结论是 $k = 4, 8$ 无解, $k \geqslant 16$ 可以填写 $\frac{3k}{16}, \frac{6k}{16}, \frac{7k}{16}$, 这样异或和仍是 0.

接着就是 m 为奇数的情形, 这样异或和至少是 1, 意味着填写的数至少是 2, 这也要求 $m \geqslant 2n$.

3. n 为偶数, m 为奇数.
   只需要填 $2, 2, \cdots, 2, 3, \frac{m-2n+3}{2}, \frac{m-2n+2}{2}$, 这时异或和为 1, 最小值为 2.
4. n 为奇数, m 为奇数.
   先填写 $n-3$ (偶数) 个 2, 再考虑后面三个数怎么填.
   1. 如果 $k = m - 2n + 6$ (奇数) 并不形如 $2^p + 1$ 或者 $2^p + 3$.

   如果设 $k' = k - 1$, 可以考虑填写 $\frac{k'}{2}, \frac{k'}{2} - \operatorname{lowbit}\left(\frac{k'}{2}\right), \operatorname{lowbit}\left(\frac{k'}{2}\right)$, 接着只需要考虑把相差的 1 加给谁.

   首先并不能填写 1, 如果 $\operatorname{lowbit}\left(\frac{k'}{2}\right) = 1$, 则需要进行一些调整, 例如:

```
          11001    11001
110011 -> 11000 -> 10001
              1     1001
```

   这时异或和为 1.

   否则只需要在后面两个数字选择一个偶数加 1 即可, 例如:

```
          11000    11000
110001 -> 10000 -> 10001
           1000     1000
```

   这时异或和为 1.

   2. 如果 $n \geqslant 5$, 可以像之前一样调整 k, 如果 k 形如 $2^p + 1$, 只需要将前面的两个 2 变成 3, 如果 k 形如 $2^p + 3$, 只需要将前面的两个 2 变成 4.
   3. 如果 $n = 3, k$ 形如 $2^p + 1$, 结论是 $k = 9, 17$ 无解, 其中情形可以填写 $\frac{3(k-1)}{16}, \frac{6(k-1)}{16} + 1, \frac{7(k-1)}{16}$.
   4. 如果 $n = 3, k$ 形如 $2^p + 3$, 结论是 $k = 7, 11, 19$ 无解, 其中情形可以填写 $7, \frac{m-7}{2} + 1, \frac{m-7}{2} - 1$.

L. Largest Unique Wins

离谱构造, 但没有 k 题 离谱. 构造纳什均衡点.

(结论题?)

结论就是 m 由 $1, 2$ 以概率 1 选择, $m-1$ 由 $3, 4$ 以概率 1 选择, $\cdots$

如果 $m > 2n$, 就空出前面的选项, 如果 $m < 2n$, 则多出的人随意选择.

从 n, m 较小的情形开始很容易得出结论.

M. Youth Finale

找规律 + 分治. 动态计算逆序数.

每次的操作有两种:

1. 将第一个数移动到末尾,
2. 将序列翻转.

首先利用归并排序计算整个序列的逆序数. 接着考虑每一种操作.

实际上只需要记录序列的起始位置以及方向即可, 对于第二种操作而言, 逆序数进行了翻转, 同时方向进行了反正, 对于第一种操作而言, 起始位置沿着方向移动一位, 逆序数加上了 $n+1-2p_{start}$.