T1

注意到值域很小，我们考虑与值域有关的做法。

设 f_i 表示 $a_{1\dots i}$ 的逆序对个数。

询问时我们只需要先计算 $f_r - f_{l-1}$ 然后消除 $[1,l)$ 与 $[l,r]$ 之间的贡献即可。

只需要对于每个 x 求出 $[l,r]$ 中 x 的个数以及 $[1,l)$ 中 $>x$ 的个数。

这可以通过预处理二维前缀和得到。时间复杂度 $O((n+m)V)$，其中 V 为值域。

T2

假设当前 b 中已经有 $1\sim k-1$，考虑如何加入 k。

如果 k 已经在 $a_{1\dots m}$ 中出现，那么直接执行操作三即可。

否则需要不断进行操作一或操作二直到 k 在 $a_{1\dots m}$ 出现为止。在这种情况下操作后一定有 $a_1=k$ 或 $a_m=k$。

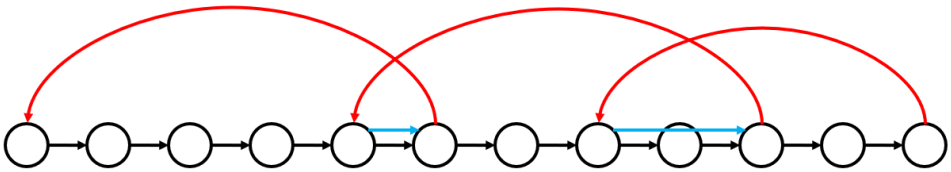
因此我们可以进行 dp：设 $dp_{k,0/1}$ 表示当前 b 中已经有 $1\sim k$，第二维为 0 时要求 $a_1=k$，否则满足 $a_m=k$，在这种情况下已进行的最小操作步数。

转移时我们只需要求出一个 k' 满足 $[k,k')$ 中的数全部都在 $a_{1\dots m}$ 中出现，并且 k' 不在 $a_{1\dots m}$ 中出现。转移到 $dp_{k',0/1}$ 即可。

求 k' 相当于是对于环上一个长度为 m 的区间求不在这个区间中且 $>k$ 的最小值。将所有询问离线下来后可以用支持插入删除前驱后继得数据结构解决。时间复杂度 $O(n\log n)$。

T3

可以发现最终经过的路径一定长这样：



其中左边是 1 号点，右边是 n 号点。黑色部分表示从 1 走到 n 经过的路径，红色的箭头表示以一条最短路径从起点走到终点，蓝色的箭头表示沿着黑色部分已经过的路径从起点走到终点。

我们考虑进行 dp。设 $dp_{u,v}$ 表示当前最后一个蓝色箭头为 $u\to v$ 的答案。

转移时我们枚举下一个蓝色箭头为 $u'\to v'$，那么有：

$$dp_{u',v'}\leftarrow dp_{u,v}+dis_{v',u}+dis_{v,u'}+dis_{u',v'}$$

但这些转移实际上没有固定顺序，我们需要使用类似于 Dijkstra 的方法，每次选择值最小的状态进行转移。

直接用这种方式转移的时间复杂度为 $O(n^4)$。

观察转移方程可以发现，我们可以将 u', v' 分开，先转移 u' 再转移 v'，增加一个中间状态即可。

这样转移数量就降到了 $O(n^3)$，使用堆维护最小的状态即可做到 $O(n^3 \log n)$。

T4

考虑同时钦定一组大小为 k 的匹配和一组大小为 k 的点覆盖，这样就能使得最大匹配 $= k$。

显然对于 k 条匹配边，一定是每条边两个端点中恰好有一个在点覆盖中，而不在匹配中的点一定都不在点覆盖中。

分别考虑两边的点，假设我们已经确定了：

- 共有 x_1 条匹配边，其中有 x_2 条左端点在点覆盖中；共有 x_3 条非匹配边，其中有 x_4 条左端点在点覆盖中。
- 共有 y_1 条匹配边，其中有 y_2 条右端点在点覆盖中；共有 y_3 条非匹配边，其中有 y_4 条右端点在点覆盖中。

只需要满足 $x_1 = y_1 = k, x_2 + y_2 = k, x_3 = y_3 \le \min\{x_4 + y_4, m - k\}$，那么一定存在一个满足限制，并且 $match(G) = k$ 的 G。

因此我们只需要对于左右两边分别 dp，状态中分别需要记录 $x_{1\dots4}, y_{1\dots4}$ 的值。

通过这两个 dp 数组即可快速得到答案。总时间复杂度为 $O(n^3 m^3)$，常数极小。