T1

直接三重循环模拟是 $O(n^3)$。

双重循环可以做到 $O(n^2)$。枚举了两个数之后，第三个数是可以直接算出来的。

为了方便代码实现，可以选择枚举 2 和 3 的情况。

T2

40 分:

数组模拟。

100 分:

链表模拟。

如何确定当前的最值？可以额外用一个数组进行标记每个数是否被拿走，以及在链表中对应的下标。每次只需要从大往小找到第一个没被拿走的数，并以这个数所在位置开始对链表进行操作就可以了。

T3

30 分：

枚举调整的活动是哪一个。不难发现，如果希望满足条件的活动尽量多，那么把调整的活动挪到无穷远处必然可以得到最大值。

所以，现在只需要考虑剩下的活动中，可以出现多少天有且仅有一个活动在这一天进行即可。对于第 t_i 天进行的活动 i，我们可以在区间 $t_i \sim t_i + m - 1$ 上做加一操作。这样一来，数组值为 1 的地方，就是一个满足条件的日子。用差分和前缀和对除掉调整时间的活动外的 $n-1$ 个活动所对应的区间进行操作即可。

时间复杂度 $O(n \times m)$。

60 分：

还是考虑枚举调整的是哪一个活动。

注意到，在 30 分的做法中，每次判断 $n-1$ 个活动中满足条件的日子有哪些这一过程是进行了大量重复操作的。因此，考虑如何快速判断，调整了某一个活动后，符合条件的天数会如何变化。

假设调整前符合条件的有 d 天。考虑调整后出现的变化：

- 当前活动必然能找到一个位置，使满足条件的天数加上 m；
- 设当前活动调整前，独自进行的天数为 x，则调整后满足条件的天数会减少 x；
- 设当前活动调整前，恰好只和另外的一个活动共同进行的天数为 y，则调整后满足条件的天数会增加 y。

因此，找到 $d+m-x+y$ 的最大值即可。

有没有快速找到 x, y 的方法？前缀和！

用 $s[i]$ 表示第 i 天有几个活动进行。

用 $f[i]$ 表示初始时一天内有且仅有一个活动进行的情况的前缀和，
$f[i] = f[i-1] + (s[i] == 1 ? 1 : 0)$。

用 $g[i]$ 表示初始时一天内恰有两个活动进行的情况的前缀和，$g[i] = g[i-1] + (s[i] == 2 ? 1 : 0)$。

那么，调整第 i 个活动可以得到的最大天数就是 $d + m - (f[r] - f[l-1]) + (g[r] - g[l-1])$，其中 l, r 分别表示这个活动的开始和结束时间。

时间负责度 $O(n+m)$。

另外的 10 分：

当 $m = 1$ 时，每个活动只会在开始的那一天进行。

不妨设调整前符合条件的有 d 天，然后可进行如下分类讨论：

- 是否存在一天恰有两个活动进行？如果有，答案为 $d+2$。
- 如果不满足上一条，继续判断是否存在一天有三个甚至更多活动进行？如果有，答案为 $d+1$。
- 如果不满足上一条，答案为 d。

100 分:

由于 m 很大，前缀和直接不可用。

但回顾前缀和的做法，会发现我们要知道的只是符合要求的天数为多少，而符合条件的天数经常会是连续的一整段。即便不能通过前缀和统计出具体到每一天有多少个活动在进行，只要我们知道符合要求的连续段的起点和终点，也能算出答案。

考虑排序。

排序后，对于第 i 个活动，对应的活动时间是 $t_i \sim t_i + m - 1$ 天。

记 $l = \ max\ (t[i], t[i-1]+m)$，$r = \ min\ (t[i]+m-1, t[i+1]-1)$。

则第 i 个活动独自开展的天数为 $d[i] = \ max\ (0, r-l+1)$。

$ans = \sum d[i]$ 就是调整前满足条件的天数。

接下来，延续之前的思路，考虑调整某一个活动对天数产生的影响。

- 当前活动必然能找到一个位置，使满足条件的天数加上 m；
- 当前活动调整前，独自进行的天数为 $d[i]$，则调整后满足条件的天数会减少 $d[i]$；
- 设当前活动调整前，恰好只和另外的一个活动共同进行的天数为 y，则调整后满足条件的天数会增加 y。

现在只要计算 y 就可以了。容易发现，相邻的两个活动才有可能出现统计 y 所需的情况。但是，对于第 i 个活动和第 $i-1$ 个活动重叠的部分，并不会直接等于 y，因为还得把当天开展 3 个及以上活动的情况排除掉。

记 $l = \ max\ (t[i], t[i-2]+m)$，$r = \ min\ (t[i+1]-1, t[i-1]+m-1)$。

此时，第 $i-1$ 个活动和第 i 个活动共同开展，且没有其他活动开展的天数为 $tmp[i] = \ max\ (0, r-l+1)$。

最终的答案为 $\ max\ \{ans - d[i] + m + tmp[i] + tmp[i+1]\}$。

时间复杂度为 $O(n \log n)$

T4

30 分:

直接模拟。时间复杂度 $O(T \times q \times n^2)$。

$60 \sim 80$ 分:

预处理每个位置的下一个相同值所在位置，记为 nxt_i。

对于右边界 r，当前位置为 i 时，如果能跳就直接跳到 nxt_i+1 位置，否则跳到 $i+1$ 位置同时令答案加一。

右边界处理好了有就是 80，否则就是 60。

时间复杂度 $O(T \times q \times n)$。

100 分:

考虑倍增跳，记 $nxt_{i,j}$ 表示位置 i 往右跳 2^j 次时可以到达的位置。

这样，对于长度为 n 的区间，最多 $\log n$ 次就可以跳完了。

数组的初始值 $nxt_{i,0}$ 就是每个位置的下一个相同值所在位置，再通过 $nxt_{i,j}=nxt_{nxt_{i,j-1}+1,j-1}$ 计算即可。

时间复杂度 $O(T \times q \times \log n)$。