a

经典时光倒流，考虑把操作倒过来，就变成每次加一个点，加点会导致一些连通块连在一起。然后询问图中所有连通块中，连通块权值和的最大值。

用并查集维护连通块以及每个连通块权值和即可，因为每个点权值是正数，所以合并过程中连通块权值和会越来越大，只需要拿当前合并的几个连通块去更新max即可。

注意开 long long，否则只会获得 50 分。

b

首先$f(n,0)=1$，接下来

$$f(n,k)=\sum_{x_1=0}^{n}\binom{n}{x_1}\sum_{x_2=0}^{x_1}\binom{x_1}{x_2}\sum_{x_3=0}^{x_2}\binom{x_2}{x_3}\cdots\sum_{x_k=0}^{x_{k-1}}\binom{x_{k-1}}{x_k}$$

对每一层都使用二项式定理剥壳：

$$\begin{aligned}f(n,k)&=\sum_{x_1=0}^{n}\binom{n}{x_1}\sum_{x_2=0}^{x_1}\binom{x_1}{x_2}\sum_{x_3=0}^{x_2}\binom{x_2}{x_3}\cdots\sum_{x_{k-1}=0}^{x_{k-2}}\binom{x_{k-2}}{x_{k-1}}\sum_{x_k=0}^{x_{k-1}}\binom{x_{k-1}}{x_k}\\&=\sum_{x_1=0}^{n}\binom{n}{x_1}\sum_{x_2=0}^{x_1}\binom{x_1}{x_2}\sum_{x_3=0}^{x_2}\binom{x_2}{x_3}\cdots\sum_{x_{k-1}=0}^{x_{k-2}}\binom{x_{k-2}}{x_{k-1}}2^{x_{k-1}}\end{aligned}$$

然后$\sum_{x_{k-1}=0}^{x_{k-2}}\binom{x_{k-2}}{x_{k-1}}2^{x_{k-1}}=\sum_{x_{k-1}=0}^{x_{k-2}}\binom{x_{k-2}}{x_{k-1}}1^{x_{k-2}-x_{k-1}}2^{x_{k-1}}=3^{x_{k-2}}$

所以

$$f(n,k)=(k+1)^n$$
$$f(f(n,i),i)=(i+1)^{(i+1)^n}$$

直接循环0~n，然后快速幂计算即可。注意运用欧拉定理计算快速幂时应该模 mod - 1 （mod为素数的情况——费马小定理）

c

分成“存在交为空的集合”与“不存在交为空的集合”来讨论。

如果存在交为空的集合，显然只能有一个（不然可以合并起来）。然后其余的集合大小一定为 1，否则可以只保留里面长度最长的线段。因此这种情况只要按长度排序，贪心选择最长的那些。

如果不存在交为空的集合，对于线段 i 和 j，如果 i 完全包含了 j，那么 i 只有两种选择：跟 j 放在同一个集合，或者自己独占一个集合。其他选择可以用调整证明不会更优。因此对于完全包含了某个其他线段的线段，我们要么忽略它，要么让它独占一个集合。对于剩下的线段，排好序后显然左右端点都是单调的，调整可以证明每个划分的集合都是占据了一个连续段。考虑 $f(i,j)$ 表示用 i 个集合划分前 j 条线段的最大权值，转移形如：

$$f(i,j)=\max_{k<j}(f(i-1,k)+r_{k+1}-l_j)$$

可以使用前缀和优化转移。

最后与完全包含了某个其他线段的线段的贡献合并一下就可以得到答案了。

时间复杂度 $O(nk)$。

d

题意：
n 个长度相同的字符串，每个字符串仅由小写字母和 `?` 组成， `?` 是通配符，可以替换成任意字符。

如果两个替换之后的字符串可以相等，那么就说这两个字符串相似，求相似字符串的对数。

题解：
从前往后扫描到一个字符串时，例如 $a?b?c$，我们可以着重考虑 $a\ b\ c$ 这三个位置能不能和之前的字符串匹配上。用状压来存每个位置是字母还是通配符，因为 $M <= 6$，只有 $1 << 6$ 种可能，所以开 $1 << 6$ 个`unordered_map`，来存每个状态对应的1位置的字符串集合，每个 `unordered_map` 的 `key` 和 `value` 分别是 1 位置的字符串和数量。比如第 00001 个 `unordered_map`，里面可以存 a，可以存 c，可以存 $?$。

那么对于上面 $a?b?c$ 的情况，我们要查询 10101 的子集，也就是说之前的字符串长什么样才能匹配上现在的 $a?b?c$。比如查到 10000 这个子集，对应的字符串可能是 $x????$，那么 x 必须等于 a，才能匹配上。

换句话说，能够匹配上 $a?b?c$ 的，其实两个 $?$ 的位置不用管，就看 abc 的位置，那之前匹配的字符串有很多对应的可能: $abc, ?bc, a?c, ab?, ??c, ?b?, a??, ???$，对应我要搜索的二进制串就是 10101 的子集：$10101, 00101, 10001, 10100, 00001....$

- 查到 10000 这个子集的时候，我们看一下第 10101 个 `unordered_map` 中 `key=a??` 对应的 `value` 有多少个，加到答案里就行。
- 查到 00101 这个子集的时候，我们看一下第 10101 个 `unordered_map` 中 `key=?bc` 对应的 `value` 有多少个，加到答案里就行。
- 查到 10101 这个子集的时候，我们看一下第 10101 个 `unordered_map` 中 `key=abc` 对应的 `value` 有多少个，加到答案里就行。
- 10101 其实就是表示，我们现在只关注 3 个位置，这 3 个位置要么和 abc 对应，要么里面有 $?$，看看前面有多少个字符串满足要求。剩余 2 个位置因为本来就有 $?$，已经搞定了。

修改 `unordered_map` 的的时候，将原字符串的所有可能的情况加入 `map`：比如 10000的位置加入 a，01000 的位置加入 $?$，11000 的位置加入 $a?$，以此类推，注意 `string` 是会 `mle` 的，需要用 `hash`。