图论

EndSaH

2019 年 8 月 12 日

前言

放的题有很多难度偏低，会根据难度调整思考时间。

放的题有很多难度偏低，会根据难度调整思考时间。

不教板子。

最短路

Luogu P3403 跳楼机

有一栋层数为 H 的楼，你初始时位于第一层。

你每次可以选择向上走 x 或 y 或 z 层，走的次数不限，但不能走到 H 层之上。

问能到达的不同的层数的个数。

$1 \leq x, y, z \leq 10^5, 1 \leq H \leq 2^{63} - 1$

Solution

这看起来就很背包...

Solution

这看起来就很背包...

不过背包显然爆炸，所以还是往 x, y, z 的数据范围上想吧。

Solution

这看起来就很背包...

不过背包显然爆炸，所以还是往 x, y, z 的数据范围上想吧。

考虑这样一种方法：

Solution

这看起来就很背包...

不过背包显然爆炸，所以还是往 x, y, z 的数据范围上想吧。

考虑这样一种方法：

先只考虑走 y, z 步，他们一定能到达一些楼层；把这些楼层再用 x 去扩展，即可得到答案。

Solution

这看起来就很背包...

不过背包显然爆炸，所以还是往 x, y, z 的数据范围上想吧。

考虑这样一种方法：

先只考虑走 y, z 步，他们一定能到达一些楼层；把这些楼层再用 x 去扩展，即可得到答案。

于是如下设状态：设 f_i 表示只走 y, z 步，楼层模 x 结果为 i 的所能到达的最小楼层。

Solution

这看起来就很背包...

不过背包显然爆炸，所以还是往 x, y, z 的数据范围上想吧。

考虑这样一种方法：

先只考虑走 y, z 步，他们一定能到达一些楼层；把这些楼层再用 x 去扩展，即可得到答案。

于是如下设状态：设 f_i 表示只走 y, z 步，楼层模 x 结果为 i 的所能到达的最小楼层。

有转移：$f_i + y \to f_{(i+y) \mod x}$，$f_i + z \to f_{(i+z) \mod x}$。

Solution

答案就是 $\sum_{i=0}^{x-1} \left\lfloor \frac{H-f_i}{x} \right\rfloor + 1$。

可以发现这玩意虽然有后效性，但实际上可以建图，转移就是连边，于是令 $f_1 = 1$，每个点连边跑最短路即可。边权非负，Dijkstra 足以胜任。

Solution

答案就是 $\sum_{i=0}^{x-1} \left\lfloor \frac{H-f_i}{x} \right\rfloor + 1$。

可以发现这玩意虽然有后效性，但实际上可以建图，转移就是连边，于是令 $f_1 = 1$，每个点连边跑最短路即可。边权非负，Dijkstra 足以胜任。

另外注意特判 x, y, z 中有一个为 1 的情况。

Solution

答案就是 $\sum_{i=0}^{x-1} \left\lfloor \frac{H-f_i}{x} \right\rfloor + 1$。

可以发现这玩意虽然有后效性，但实际上可以建图，转移就是连边，于是令 $f_1 = 1$，每个点连边跑最短路即可。边权非负，Dijkstra 足以胜任。

另外注意特判 x, y, z 中有一个为 1 的情况。

$\mathcal{O}(x \log x)$

Solution

答案就是 $\sum_{i=0}^{x-1} \lfloor \frac{H-f_i}{x} \rfloor + 1$。

可以发现这玩意虽然有后效性，但实际上可以建图，转移就是连边，于是令 $f_1 = 1$，每个点连边跑最短路即可。边权非负，Dijkstra 足以胜任。

另外注意特判 x, y, z 中有一个为 1 的情况。

$\mathcal{O}(x \log x)$

本质上应该是利用同余和计数优化了空间，类似的题还有“墨墨的等式”，很多人应该都做过。

51nod1693 水群

初始时，你的输入框中有一个滑稽。

现在你可以进行以下三种操作：

1. 全选复制：将所有滑稽复制到剪切板中。
2. 粘贴：把剪切板中的滑稽粘贴到输入框中。
3. 退格：删掉一个滑稽。

问，至少需要多少次操作，才能让输入框中恰好有 n 个表情?

$1 \leq n \leq 10^6$

时限：0.4s

Bonus: 0.1s

Solution

因为有退格无法直接 DP，考虑建图。

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

于是就可以考虑建图跑最短路，$x \to x*k, x-1$。

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

于是就可以考虑建图跑最短路，$x \to x*k, x-1$。

显然爆炸，考虑优化。

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

于是就可以考虑建图跑最短路，$x \to x * k, x - 1$。

显然爆炸，考虑优化。

可以想到 k 为合数时可以分解成几个质因数，相当于将一次复制很多次粘贴变成复制粘贴复制粘贴的几个小块，这样会更优一些（你也可以看成多个数的和会比多个数的积小）

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

于是就可以考虑建图跑最短路，$x \to x * k, x - 1$。

显然爆炸，考虑优化。

可以想到 k 为合数时可以分解成几个质因数，相当于将一次复制很多次粘贴变成复制粘贴复制粘贴的几个小块，这样会更优一些（你也可以看成多个数的和会比多个数的积小）

所以 $x \to x * k$ 中的 k 现在被限制为了质数。

Solution

因为有退格无法直接 DP，考虑建图。

发扬人类智慧可以知道过程一定是：一次复制，多次粘贴，多次退格这样不断重复。

于是就可以考虑建图跑最短路，$x \to x*k, x-1$。

显然爆炸，考虑优化。

可以想到 k 为合数时可以分解成几个质因数，相当于将一次复制很多次粘贴变成复制粘贴复制粘贴的几个小块，这样会更优一些（你也可以看成多个数的和会比多个数的积小）

所以 $x \to x*k$ 中的 k 现在被限制为了质数。

然而只是这个优化距离 0.4s 还有很远。

Solution

通过打表的辅助可以得到两个结论：

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

只用第二个结论就可以卡进 0.4s 了，但还不能进 0.1s。

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

只用第二个结论就可以卡进 0.4s 了，但还不能进 0.1s。

最短路很难继续利用第一个结论了，只能考虑回到 DP:

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

只用第二个结论就可以卡进 0.4s 了，但还不能进 0.1s。

最短路很难继续利用第一个结论了，只能考虑回到 DP:

设 $F_{i,0/1}$ 表示从 1 到 i，上一条边能否是退格边的最小距离。

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

只用第二个结论就可以卡进 0.4s 了，但还不能进 0.1s。

最短路很难继续利用第一个结论了，只能考虑回到 DP:

设 $F_{i,0/1}$ 表示从 1 到 i，上一条边能否是退格边的最小距离。

有转移方程：

$$F_{i,1} = \min(\min_{k=1}^{4}\{F_{i+k,0}\}, \min_{p\in\mathbb{P}}^{11}\{F_{\frac{i}{p},1}\})$$

$F_{i,0}$ 只用后面那块转移就可以了。

Solution

通过打表的辅助可以得到两个结论：

在 $n \leq 10^6$ 时，连续退格操作不会超过四次，$k > 11$ 的边在最短路中不会出现。

只用第二个结论就可以卡进 0.4s 了，但还不能进 0.1s。

最短路很难继续利用第一个结论了，只能考虑回到 DP:

设 $F_{i,0/1}$ 表示从 1 到 i，上一条边能否是退格边的最小距离。

有转移方程：

$$F_{i,1} = \min(\min_{k=1}^{4}\{F_{i+k,0}\}, \min_{p\in\mathbb{P}}^{11}\{F_{\frac{i}{p},1}\})$$

$F_{i,0}$ 只用后面那块转移就可以了。

用记忆化搜索实现，跑的飞快。

$\mathcal{O}(\text{Unknown})$

BZOJ4289 Tax

给出一个 n 个点 m 条边的无向图，经过一个点的代价是进入和离开这个点的两条边的边权的较大值，求从 1 到 n 的最小代价。起点的代价是离开起点的边的边权，终点的代价是进入终点的边的边权。

$1 \leq n \leq 10^5, 1 \leq m \leq 2 \times 10^5$

Solution

直接做不可做，先考虑怎么建新图。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

对于从起点出发和汇入终点的边，新建源点 S 汇点 T，连接 S 和从起点出发的有向边，T 和汇入终点的有向边，跑最短路就行。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

对于从起点出发和汇入终点的边，新建源点 S 汇点 T，连接 S 和从起点出发的有向边，T 和汇入终点的有向边，跑最短路就行。

然而边数是 $\mathcal{O}(m^2)$ 的，需要优化。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

对于从起点出发和汇入终点的边，新建源点 S 汇点 T，连接 S 和从起点出发的有向边，T 和汇入终点的有向边，跑最短路就行。

然而边数是 $\mathcal{O}(m^2)$ 的，需要优化。

可以想到这样建边很浪费；若是从小往大建，相邻边建差值的话，不仅边数是 $O(m)$ 的，建出来跑最短路还是等价的。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

对于从起点出发和汇入终点的边，新建源点 S 汇点 T，连接 S 和从起点出发的有向边，T 和汇入终点的有向边，跑最短路就行。

然而边数是 $\mathcal{O}(m^2)$ 的，需要优化。

可以想到这样建边很浪费；若是从小往大建，相邻边建差值的话，不仅边数是 $O(m)$ 的，建出来跑最短路还是等价的。

于是考虑如何差分建边。先枚举中转点 b，对其出边入边排序（其实这里出边入边排序结果是一样的，因为原图是无向图），对所有相邻出边，小的往大的连权值之差的边，大的往小的连权值为 0 的边，再对所有入边，往它对应权值一样的出边连其权值。

Solution

直接做不可做，先考虑怎么建新图。

可以把每条边拆成两条有向边，再将每条有向边视为点，对于原图上的两条相接的边 $(a,b),(b,c)$，在新图上便将 (a,b) 表示的这个点连向 (b,c) 表示的点，边权为他们两个的较大值。

对于从起点出发和汇入终点的边，新建源点 S 汇点 T，连接 S 和从起点出发的有向边，T 和汇入终点的有向边，跑最短路就行。

然而边数是 $\mathcal{O}(m^2)$ 的，需要优化。

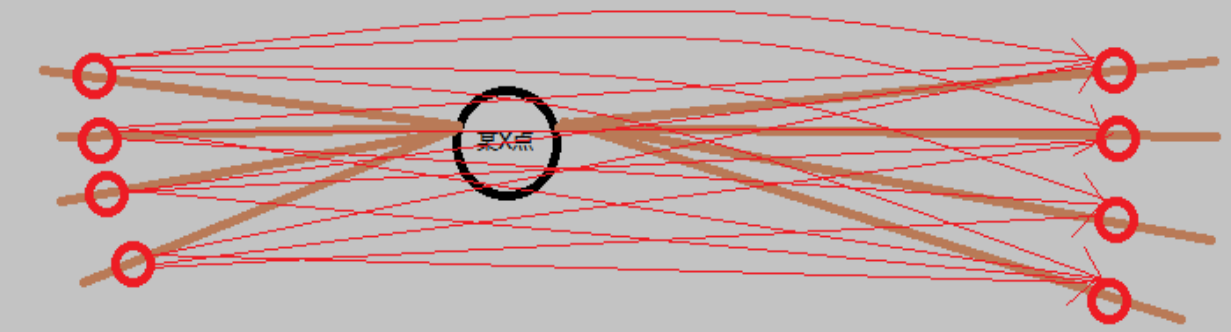
可以想到这样建边很浪费；若是从小往大建，相邻边建差值的话，不仅边数是 $O(m)$ 的，建出来跑最短路还是等价的。

于是考虑如何差分建边。先枚举中转点 b，对其出边入边排序（其实这里出边入边排序结果是一样的，因为原图是无向图），对所有相邻出边，小的往大的连权值之差的边，大的往小的连权值为 0 的边，再对所有入边，往它对应权值一样的出边连其权值。

可能用图讲更清楚：

Solution

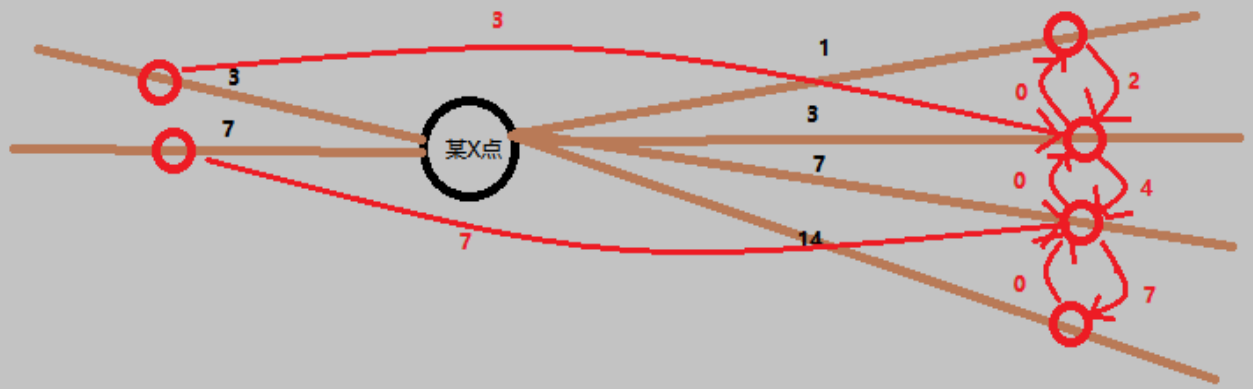
全是个人理解：



按照老办法，每个入边（点）都和每一个出边（点）建边，那边数的数量级为平方级的

入边

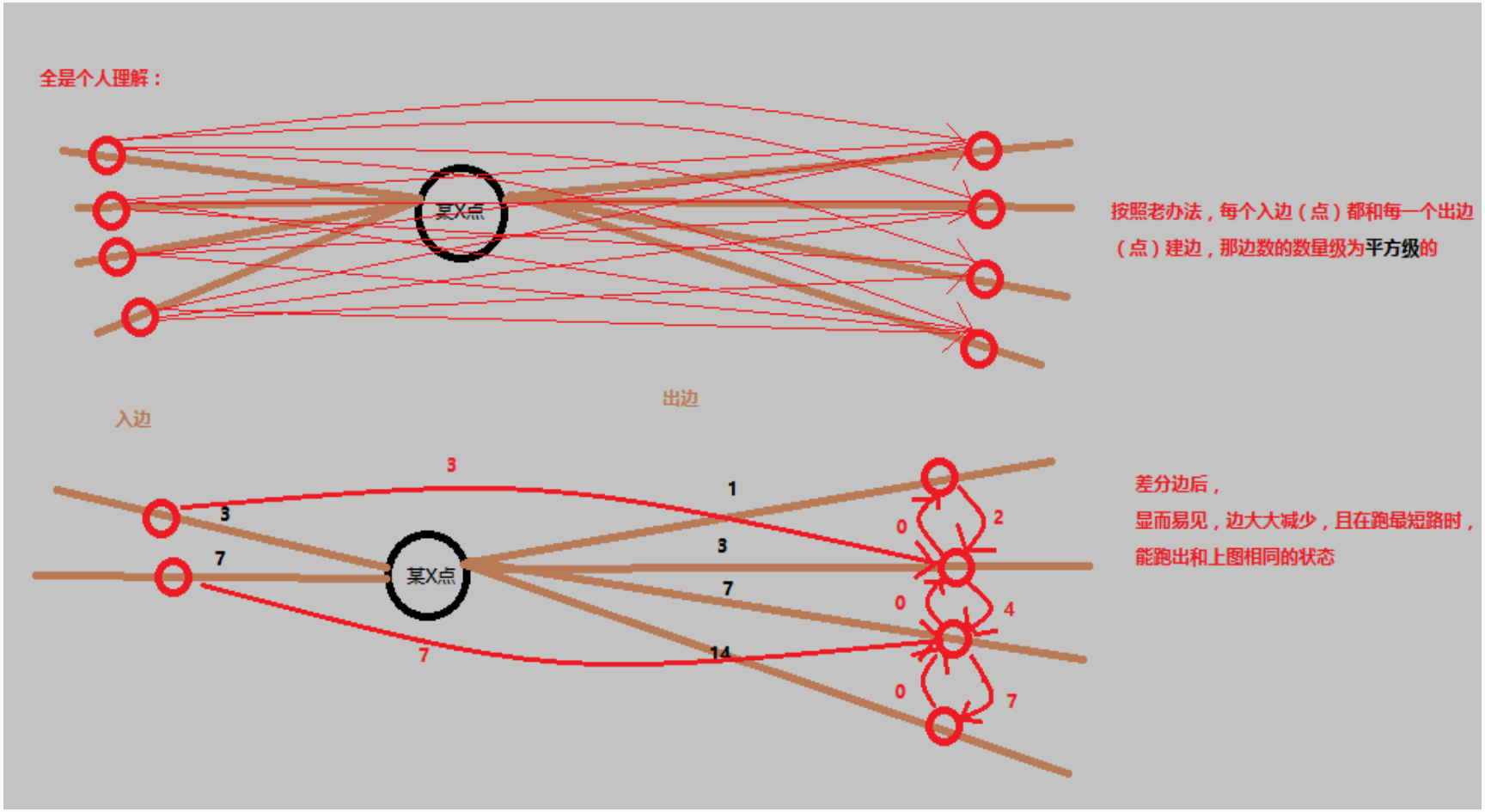
出边



差分边后，

显而易见，边大大减少，且在跑最短路时，能跑出和上图相同的状态

Solution



$\mathcal{O}((n+m)\log m)$

给定一个 n 个点 m 条边的有向图，边有权值，以及 k 个关键点。

求出 k 个点之间两两最短路的最小值。

$1 \leq n \leq 10^5, 1 \leq m \leq 5 \times 10^5, 2 \leq k \leq n$

Solution1

提供两种思路。

Solution1

提供两种思路。

解法一：

Solution1

提供两种思路。

解法一：

假如将特殊点分为 A, B 两个集合，新建源点 S 向 A 中所有点连权值为 0 的边，B 中所有点向新建汇点 T 连权值为 0 的边，S 到 T 之间的最短路即为 A, B 集合点之间的最短路最小值。

Solution1

提供两种思路。

解法一：

假如将特殊点分为 A, B 两个集合，新建源点 S 向 A 中所有点连权值为 0 的边，B 中所有点向新建汇点 T 连权值为 0 的边，S 到 T 之间的最短路即为 A, B 集合点之间的最短路最小值。

不可能枚举划分两个集合。考虑二进制分组，枚举第 p 位，将第 p 位为 0 的关键点划在 A，为 1 的划在 B，每位都建图跑一次最短路，最终答案即这 $\log n$ 次最短路所得答案的最小值。

Solution1

提供两种思路。

解法一：

假如将特殊点分为 A, B 两个集合，新建源点 S 向 A 中所有点连权值为 0 的边，B 中所有点向新建汇点 T 连权值为 0 的边，S 到 T 之间的最短路即为 A, B 集合点之间的最短路最小值。

不可能枚举划分两个集合。考虑二进制分组，枚举第 p 位，将第 p 位为 0 的关键点划在 A，为 1 的划在 B，每位都建图跑一次最短路，最终答案即这 $\log n$ 次最短路所得答案的最小值。

最终答案的两个点肯定至少会有一个二进制位不同，正确性显然。

Solution1

提供两种思路。

解法一：

假如将特殊点分为 A, B 两个集合，新建源点 S 向 A 中所有点连权值为 0 的边，B 中所有点向新建汇点 T 连权值为 0 的边，S 到 T 之间的最短路即为 A, B 集合点之间的最短路最小值。

不可能枚举划分两个集合。考虑二进制分组，枚举第 p 位，将第 p 位为 0 的关键点划在 A，为 1 的划在 B，每位都建图跑一次最短路，最终答案即这 $\log n$ 次最短路所得答案的最小值。

最终答案的两个点肯定至少会有一个二进制位不同，正确性显然。

$\mathcal{O}((n+m+k)\log m\log n)$

Solution2

解法二：

Solution2

解法二：

考虑最终答案的路径，一定可以被划分为：$a \to u \to v \to b$。

其中 u, v 是一条边所连接的两个普通点（关键，非关键都可，也可以与 a, b 重合），a, b 分别是满足**从其出发到达 u 最近的关键点，从 v 出发能到达的最近的关键点**。

Solution2

解法二：

考虑最终答案的路径，一定可以被划分为：$a \to u \to v \to b$。

其中 u, v 是一条边所连接的两个普通点（关键，非关键都可，也可以与 a, b 重合），a, b 分别是满足**从其出发到达 u 最近的关键点，从 v 出发能到达的最近的关键点**。

于是最开始将所有关键点丢进优先队列跑最短路，过程中将其从哪个关键点更新得到记录下来，并且建反图也跑一遍，这样就能得到所有点的 a, b。

Solution2

解法二：

考虑最终答案的路径，一定可以被划分为：$a \to u \to v \to b$。

其中 u, v 是一条边所连接的两个普通点（关键，非关键都可，也可以与 a, b 重合），a, b 分别是满足**从其出发到达 u 最近的关键点，从 v 出发能到达的最近的关键点**。

于是最开始将所有关键点丢进优先队列跑最短路，过程中将其从哪个关键点更新得到记录下来，并且建反图也跑一遍，这样就能得到所有点的 a, b。

最后枚举所有边，设边的两点为 u, v，其权值为 w，那么若 u 在正图上和 v 在反图上不是被同一个关键点更新，则用 $dis_{0,u} + dis_{1,v} + w$（$w$ 为边权，$dis_{0,u}$ 表示从某个关键点出发到达 u 的最短路的最小值，$dis_{1,v}$ 同理）更新答案。（取 min）

Solution2

解法二：

考虑最终答案的路径，一定可以被划分为：$a \to u \to v \to b$。

其中 u, v 是一条边所连接的两个普通点（关键，非关键都可，也可以与 a, b 重合），a, b 分别是满足**从其出发到达 u 最近的关键点，从 v 出发能到达的最近的关键点**。

于是最开始将所有关键点丢进优先队列跑最短路，过程中将其从哪个关键点更新得到记录下来，并且建反图也跑一遍，这样就能得到所有点的 a, b。

最后枚举所有边，设边的两点为 u, v，其权值为 w，那么若 u 在正图上和 v 在反图上不是被同一个关键点更新，则用 $dis_{0,u} + dis_{1,v} + w$（$w$ 为边权，$dis_{0,u}$ 表示从某个关键点出发到达 u 的最短路的最小值，$dis_{1,v}$ 同理）更新答案。（取 min）

$\mathcal{O}((n + m + k) \log m)$

~~Solution3~~

由于原题数据暴力直接过了，所以这里也说说暴力。

~~Solution3~~

由于原题数据暴力直接过了，所以这里也说说暴力。

对每个关键点都跑一次 Dijkstra。由于 Dijkstra 的性质，最短路最小的点一定会先被取出来，所以只要到取出的点是关键点时就可以跑下一个点了。

由于原题数据暴力直接过了，所以这里也说说暴力。

对每个关键点都跑一次 Dijkstra。由于 Dijkstra 的性质，最短路最小的点一定会先被取出来，所以只要到取出的点是关键点时就可以跑下一个点了。

虽然最坏下依然 $\mathcal{O}(k(n+m)\log m)$，但可以预见这样的数据很难构造，而且时限不知道为什么开了 5s，所以说理论上能骗到很多分。

由于原题数据暴力直接过了，所以这里也说说暴力。

对每个关键点都跑一次 Dijkstra。由于 Dijkstra 的性质，最短路最小的点一定会先被取出来，所以只要到取出的点是关键点时就可以跑下一个点了。

虽然最坏下依然 $\mathcal{O}(k(n+m)\log m)$，但可以预见这样的数据很难构造，而且时限不知道为什么开了 5s，所以说理论上能骗到很多分。

实际上，这个解法在洛咕上最优解 rk1。

给定一个 n 点 m 边的无向图。现要新加入一个点，这个点与 n 点分别连了一条边。需要设定这 n 条边的边权（可为实数），满足 n 个点到 $1,2$ 号节点的最短路不一定经过这个新点，并且使这 n 条边的边权平均值最小。

$1 \leq n, m \leq 10^5$

Solution

这题直接构造貌似非常不可做... 边权之间会互相影响。

Solution

这题直接构造貌似非常不可做... 边权之间会互相影响。

于是只能~~去看题解~~考虑抽象上的意义。首先设加边前 a_i 表示 1 号点到 i 号点的最短路，b_i 表示 2 号点到 i 号点的最短路，t 为新加点，d_i 为连接 i 和 t 的那条边的边权，加边后 $dis_{i,j}$ 表 i, j 间的最短路。

Solution

这题直接构造貌似非常不可做... 边权之间会互相影响。

于是只能~~去看题解~~考虑抽象上的意义。首先设加边前 a_i 表示 1 号点到 i 号点的最短路，b_i 表示 2 号点到 i 号点的最短路，t 为新加点，d_i 为连接 i 和 t 的那条边的边权，加边后 $dis_{i,j}$ 表 i,j 间的最短路。

由于 1 到 i 最短路不一定经过 t，故加完边后一定满足：$dis_{1,t} + dis_{t,i} \geq dis_{1,i}$，对 2 同理。

Solution

这题直接构造貌似非常不可做... 边权之间会互相影响。

于是只能~~去看题解~~考虑抽象上的意义。首先设加边前 a_i 表示 1 号点到 i 号点的最短路，b_i 表示 2 号点到 i 号点的最短路，t 为新加点，d_i 为连接 i 和 t 的那条边的边权，加边后 $dis_{i,j}$ 表 i, j 间的最短路。

由于 1 到 i 最短路不一定经过 t，故加完边后一定满足：$dis_{1,t} + dis_{t,i} \geq dis_{1,i}$，对 2 同理。

而且显然有 $d_1 \geq dis_{1,t}, d_i \geq dis_{t,i}, a_i \leq dis_{1,i}$，所以

$$d_1 + d_i \geq a_i \tag{1}$$

$$d_2 + d_i \geq b_i \tag{2}$$

Solution

这题直接构造貌似非常不可做... 边权之间会互相影响。

于是只能~~去看题解~~考虑抽象上的意义。首先设加边前 a_i 表示 1 号点到 i 号点的最短路，b_i 表示 2 号点到 i 号点的最短路，t 为新加点，d_i 为连接 i 和 t 的那条边的边权，加边后 $dis_{i,j}$ 表 i, j 间的最短路。

由于 1 到 i 最短路不一定经过 t，故加完边后一定满足：$dis_{1,t} + dis_{t,i} \geq dis_{1,i}$，对 2 同理。

而且显然有 $d_1 \geq dis_{1,t}, d_i \geq dis_{t,i}, a_i \leq dis_{1,i}$，所以

$$d_1 + d_i \geq a_i \tag{1}$$

$$d_2 + d_i \geq b_i \tag{2}$$

故

$$d_i \geq \max(a_i - d_1, b_i - d_2)$$

注意，这里写的看起来各个点的 d_i 无关，是因为这个界取的不紧，若考虑另一个点 j，在 $d_1 + d_j \geq a_j$ 这个式子中，d_1 这一块是可以被替换成一个更紧的界的（比如当 $d_1 \geq d_i + a_i$ 时，被替换为 $d_i + a_i$），所以所有的 d 之间仍互相关联。

Solution

注意，这里写的看起来各个点的 d_i 无关，是因为这个界取的不紧，若考虑另一个点 j，在 $d_1 + d_j \geq a_j$ 这个式子中，d_1 这一块是可以被替换成一个更紧的界的（比如当 $d_1 \geq d_i + a_i$ 时，被替换为 $d_i + a_i$），所以所有的 d 之间仍互相关联。

考虑当 d_i 确定时，d_1 的取值。

注意，这里写的看起来各个点的 d_i 无关，是因为这个界取的不紧，若考虑另一个点 j，在 $d_1 + d_j \geq a_j$ 这个式子中，d_1 这一块是可以被替换成一个更紧的界的（比如当 $d_1 \geq d_i + a_i$ 时，被替换为 $d_i + a_i$），所以所有的 d 之间仍互相关联。

考虑当 d_i 确定时，d_1 的取值。

可以发现，$d_1 > d_i + a_i$ 时，它对其他点的限制会被定值 $d_i + a_i$ 代替，答案又要求尽量小，所以 d_1 取大了是没有意义的。

注意，这里写的看起来各个点的 d_i 无关，是因为这个界取的不紧，若考虑另一个点 j，在 $d_1 + d_j \geq a_j$ 这个式子中，d_1 这一块是可以被替换成一个更紧的界的（比如当 $d_1 \geq d_i + a_i$ 时，被替换为 $d_i + a_i$），所以所有的 d 之间仍互相关联。

考虑当 d_i 确定时，d_1 的取值。

可以发现，$d_1 > d_i + a_i$ 时，它对其他点的限制会被定值 $d_i + a_i$ 代替，答案又要求尽量小，所以 d_1 取大了是没有意义的。

所以

$$d_i + a_i \geq d_1 \tag{3}$$

$$d_i + b_i \geq d_2 \tag{4}$$

故

$$d_i \geq \max(d_1 - a_i, d_2 - b_i)$$

故

$$d_i \geq \max(d_1 - a_i, d_2 - b_i)$$

发现这就是上面式子的相反数。于是有

$$d_i \geq \max(|a_i - d_1|, |b_i - d_2|)$$

因为要求 d_i 尽量小，所以当 d_1, d_2 确定时取等号最优。

故

$$d_i \geq \max(d_1 - a_i, d_2 - b_i)$$

发现这就是上面式子的相反数。于是有

$$d_i \geq \max(|a_i - d_1|, |b_i - d_2|)$$

因为要求 d_i 尽量小，所以当 d_1, d_2 确定时取等号最优。

现在问题变为：确定 d_1, d_2 使 $\sum_{i=1}^{n} \max(|a_i - d_1|, |b_i - d_2|)$ 最小。

Solution

这个式子看起来就很切比雪夫... 相当于是平面上 n 个点 (a_i, b_i)，求一个点 (d_1, d_2)，使其到所有点切比雪夫距离和最小。

Solution

这个式子看起来就很切比雪夫... 相当于是平面上 n 个点 (a_i, b_i)，求一个点 (d_1, d_2)，使其到所有点切比雪夫距离和最小。

于是有一个经典的切比雪夫哈密顿距离互转——切比雪夫转哈密顿，只需要将坐标缩放为 $(\frac{x+y}{2}, \frac{x-y}{2})$，然后变成了曼哈顿距离和最小。

Solution

这个式子看起来就很切比雪夫... 相当于是平面上 n 个点 (a_i, b_i)，求一个点 (d_1, d_2)，使其到所有点切比雪夫距离和最小。

于是有一个经典的切比雪夫哈密顿距离互转——切比雪夫转哈密顿，只需要将坐标缩放为 $(\frac{x+y}{2}, \frac{x-y}{2})$，然后变成了曼哈顿距离和最小。

这个问题也是个经典问题，一维时，是坐标的中位数；两维时就是 x, y 坐标分别取中位数。

Solution

这个式子看起来就很切比雪夫... 相当于是平面上 n 个点 (a_i, b_i)，求一个点 (d_1, d_2)，使其到所有点切比雪夫距离和最小。

于是有一个经典的切比雪夫哈密顿距离互转——切比雪夫转哈密顿，只需要将坐标缩放为 $(\frac{x+y}{2}, \frac{x-y}{2})$，然后变成了曼哈顿距离和最小。

这个问题也是个经典问题，一维时，是坐标的中位数；两维时就是 x, y 坐标分别取中位数。

求得这个中位数坐标算答案就行。为了避免精度误差，一开始转坐标系的时候可以先不除以 2，最终答案再除以 2。

Solution

这个式子看起来就很切比雪夫... 相当于是平面上 n 个点 (a_i, b_i)，求一个点 (d_1, d_2)，使其到所有点切比雪夫距离和最小。

于是有一个经典的切比雪夫哈密顿距离互转——切比雪夫转哈密顿，只需要将坐标缩放为 $(\frac{x+y}{2}, \frac{x-y}{2})$，然后变成了曼哈顿距离和最小。

这个问题也是个经典问题，一维时，是坐标的中位数；两维时就是 x, y 坐标分别取中位数。

求得这个中位数坐标算答案就行。为了避免精度误差，一开始转坐标系的时候可以先不除以 2，最终答案再除以 2。

$\mathcal{O}((n+m)\log m)$

生成树

CF125E MST Company

给定一张 n 点 m 边的图，求满足节点 1 度数恰好 k 时的最小生成树。

无解输出 -1，否则输出方案。

$1 \leq n \leq 5000, 0 \leq m \leq 10^5, 0 \leq k < 5000, 1 \leq w_i \leq 10^5$

Bonus: $1 \leq k \leq n \leq 10^5$

Solution1

我们称满足条件的最小生成树为强制 k 度生成树。

还是有两种解法。

Solution1

我们称满足条件的最小生成树为强制 k 度生成树。

还是有两种解法。

解法一：

Solution1

我们称满足条件的最小生成树为强制 k 度生成树。

还是有两种解法。

解法一：

首先去掉 1 号点跑最小生成森林。

Solution1

我们称满足条件的最小生成树为强制 k 度生成树。

还是有两种解法。

解法一：

首先去掉 1 号点跑最小生成森林。

现在加入 1 号点。设此时除 1 号点外的联通块个数为 x，若 $k < x$，必然无解。

Solution1

我们称满足条件的最小生成树为强制 k 度生成树。

还是有两种解法。

解法一：

首先去掉 1 号点跑最小生成森林。

现在加入 1 号点。设此时除 1 号点外的联通块个数为 x，若 $k < x$，必然无解。

否则对于每个联通块加入能联通这个联通块的最小的边，这样就得到了一棵强制 x 度生成树。

Solution1

现在依然考虑经典的破圈算法：

对于一条不在生成树上的与 1 相连的边，若将其加到最小生成树中，其可以替换所产生的环上的一条最大的边，会使权值变小 $maxEdge - w$，其中 $maxEdge$ 是最大边边权，w 是这条边的边权。

Solution1

现在依然考虑经典的破圈算法：

对于一条不在生成树上的与 1 相连的边，若将其加到最小生成树中，其可以替换所产生的环上的一条最大的边，会使权值变小 $maxEdge - w$，其中 $maxEdge$ 是最大边边权，w 是这条边的边权。

在所有 $maxEdge - w$ 找到其最大值并替换，在所替换的最大边不与 1 相连的前提下，我们就能得到一棵强制 $x + 1$ 度生成树。

Solution1

现在依然考虑经典的破圈算法：

对于一条不在生成树上的与 1 相连的边，若将其加到最小生成树中，其可以替换所产生的环上的一条最大的边，会使权值变小 $maxEdge - w$，其中 $maxEdge$ 是最大边边权，w 是这条边的边权。

在所有 $maxEdge - w$ 找到其最大值并替换，在所替换的最大边不与 1 相连的前提下，我们就能得到一棵强制 $x+1$ 度生成树。

将这个过程进行 $k-x$ 次，就能得到题目所要求的强制 k 度生成树。

Solution1

现在依然考虑经典的破圈算法：

对于一条不在生成树上的与 1 相连的边，若将其加到最小生成树中，其可以替换所产生的环上的一条最大的边，会使权值变小 $maxEdge - w$，其中 $maxEdge$ 是最大边边权，w 是这条边的边权。

在所有 $maxEdge - w$ 找到其最大值并替换，在所替换的最大边不与 1 相连的前提下，我们就能得到一棵强制 $x+1$ 度生成树。

将这个过程进行 $k-x$ 次，就能得到题目所要求的强制 k 度生成树。

找到不与 1 相连的最大边由于数据范围原因可以暴力算。设 $maxE_i$ 表示在当前生成树上从 1 出发到 i 节点的路径上**忽略与 1 相连的边**的最大边，每次 DFS 一遍 $O(n)$ 算就行。

Solution1

现在依然考虑经典的破圈算法：

对于一条不在生成树上的与 1 相连的边，若将其加到最小生成树中，其可以替换所产生的环上的一条最大的边，会使权值变小 $maxEdge - w$，其中 $maxEdge$ 是最大边边权，w 是这条边的边权。

在所有 $maxEdge - w$ 找到其最大值并替换，在所替换的最大边不与 1 相连的前提下，我们就能得到一棵强制 $x+1$ 度生成树。

将这个过程进行 $k-x$ 次，就能得到题目所要求的强制 k 度生成树。

找到不与 1 相连的最大边由于数据范围原因可以暴力算。设 $maxE_i$ 表示在当前生成树上从 1 出发到 i 节点的路径上**忽略与 1 相连的边**的最大边，每次 DFS 一遍 $O(n)$ 算就行。

由于与 1 相连的边数量级是 n 而不是 m，所以复杂度：

$\mathcal{O}(m \log m + nk)$

Solution2

解法二：

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

可以发现随着 p 的增加，1 号点在最终的最小生成树里的度数会逐渐减少（与 1 相连的边在排序后的数组里在往后靠），所以这个结果是具有单调性的。

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

可以发现随着 p 的增加，1 号点在最终的最小生成树里的度数会逐渐减少（与 1 相连的边在排序后的数组里在往后靠），所以这个结果是具有单调性的。

所以每次都跑一遍 Kruskal，再根据跑出来最小生成树中 1 的度数确定二分范围。

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

可以发现随着 p 的增加，1 号点在最终的最小生成树里的度数会逐渐减少（与 1 相连的边在排序后的数组里在往后靠），所以这个结果是具有单调性的。

所以每次都跑一遍 Kruskal，再根据跑出来最小生成树中 1 的度数确定二分范围。

另外需要优先保证有解，所以即使二分出来的结果大于 k，也只能加入前 k 条边。（但这不影响二分范围的确定）

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

可以发现随着 p 的增加，1 号点在最终的最小生成树里的度数会逐渐减少（与 1 相连的边在排序后的数组里在往后靠），所以这个结果是具有单调性的。

所以每次都跑一遍 Kruskal，再根据跑出来最小生成树中 1 的度数确定二分范围。

另外需要优先保证有解，所以即使二分出来的结果大于 k，也只能加入前 k 条边。（但这不影响二分范围的确定）

$\mathcal{O}(m \log m \log w)$

Solution2

解法二：

考虑使用 Kruskal 算法作为基础。

因为有 1 号点强制连 k 条边的条件，所以我们可以在排序过程中对与 1 号点相连的边动一些手脚。

二分一个权值 $p(p \in \mathbb{Z})$，将所有与 1 相连的边的权值都加上 p 再进行 Kruskal 的排序过程。

可以发现随着 p 的增加，1 号点在最终的最小生成树里的度数会逐渐减少（与 1 相连的边在排序后的数组里在往后靠），所以这个结果是具有单调性的。

所以每次都跑一遍 Kruskal，再根据跑出来最小生成树中 1 的度数确定二分范围。

另外需要优先保证有解，所以即使二分出来的结果大于 k，也只能加入前 k 条边。（但这不影响二分范围的确定）

$\mathcal{O}(m \log m \log w)$

这个算法貌似有个单独的名字叫 wqs 二分，后两天 DP 优化的专题会讲。

魔术师的桌子上有 n 个杯子排成一行，编号为 $1, 2, \ldots, n$，每个杯子有其价值 a_i。

其中某些杯子底下藏有一个小球，如果你准确地猜出是哪些杯子，你就可以获得奖品。花费 $a_i \oplus a_j$ 元，魔术师就会告诉你杯子 $i, i+1, \ldots, j$ 底下藏有球的总数的奇偶性。

采取最优的询问策略，你至少需要花费多少元，才能保证猜出哪些杯子底下藏着球?

$1 \leq n \leq 10^5$

Solution

要知道每个位置下球的状况，就需要知道每个位置的 $sum_i - sum_{i-1}$（sum_i 表示前缀球个数）。

Solution

要知道每个位置下球的状况，就需要知道每个位置的 $sum_i - sum_{i-1}$（sum_i 表示前缀球个数）。

实质上，一个询问 (l, r) 就是告诉你 $sum_r - sum_{l-1}$ 的奇偶性，此时，sum_r 与 sum_{l-1} 便关联了起来。

Solution

要知道每个位置下球的状况，就需要知道每个位置的 $sum_i - sum_{i-1}$（sum_i 表示前缀球个数）。

实质上，一个询问 (l, r) 就是告诉你 $sum_r - sum_{l-1}$ 的奇偶性，此时，sum_r 与 sum_{l-1} 便关联了起来。

于是原问题变为以最小代价将 $n+1$ 个变量全部关联起来。若是将关联关系抽象为边，可以发现答案就是**最小生成树**。

Solution

要知道每个位置下球的状况，就需要知道每个位置的 $sum_i - sum_{i-1}$（sum_i 表示前缀球个数）。

实质上，一个询问 (l, r) 就是告诉你 $sum_r - sum_{l-1}$ 的奇偶性，此时，sum_r 与 sum_{l-1} 便关联了起来。

于是原问题变为以最小代价将 $n+1$ 个变量全部关联起来。若是将关联关系抽象为边，可以发现答案就是**最小生成树**。

于是问题变为求解最小异或生成树。

在此之前先要了解一个最小生成树算法：Boruvka 算法。

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

简易来说就是一个多路增广的 Kruskal。

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

简易来说就是一个多路增广的 Kruskal。

执行以下过程，直到图完全联通为止：

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

简易来说就是一个多路增广的 Kruskal。

执行以下过程，直到图完全联通为止：

对于每个联通块，找到其与其他联通块相连的最小边，再将这些最小边加入到生成树中，并将两个联通块并为一个。

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

简易来说就是一个多路增广的 Kruskal。

执行以下过程，直到图完全联通为止：

对于每个联通块，找到其与其他联通块相连的最小边，再将这些最小边加入到生成树中，并将两个联通块并为一个。

最初始时每个点都是一个联通块。

Boruvka 算法

这是个老牌算法，可以看作是 Kruskal 与 Prim 的结合体。

简易来说就是一个多路增广的 Kruskal。

执行以下过程，直到图完全联通为止：

对于每个联通块，找到其与其他联通块相连的最小边，再将这些最小边加入到生成树中，并将两个联通块并为一个。

最初始时每个点都是一个联通块。

时间复杂度 $\mathcal{O}((m+n)\log n)$。但是复杂度瓶颈在于找到那些联通块的最小边。

最小异或生成树

一开始将所有点的点权加入 01Trie 树中，考虑加最小出边的过程。

最小异或生成树

一开始将所有点的点权加入 01Trie 树中，考虑加最小出边的过程。

可以发现，对于每个 01Trie 上的非叶子节点，其两个 0, 1 儿子子树内的点所形成的联通块一定还没有联通。

最小异或生成树

一开始将所有点的点权加入 01Trie 树中，考虑加最小出边的过程。

可以发现，对于每个 01Trie 上的非叶子节点，其两个 0, 1 儿子子树内的点所形成的联通块一定还没有联通。

可以说 01Trie 变为了 Boruvka 算法的分治树。

最小异或生成树

一开始将所有点的点权加入 01Trie 树中，考虑加最小出边的过程。

可以发现，对于每个 01Trie 上的非叶子节点，其两个 0, 1 儿子子树内的点所形成的联通块一定还没有联通。

可以说 01Trie 变为了 Boruvka 算法的分治树。

于是深搜 01Trie，找到每个节点 0, 1 儿子之间的最小异或值然后连接 0, 1 儿子联通块，答案即连边之和。

最小异或生成树

一开始将所有点的点权加入 01Trie 树中，考虑加最小出边的过程。

可以发现，对于每个 01Trie 上的非叶子节点，其两个 0, 1 儿子子树内的点所形成的联通块一定还没有联通。

可以说 01Trie 变为了 Boruvka 算法的分治树。

于是深搜 01Trie，找到每个节点 0, 1 儿子之间的最小异或值然后连接 0, 1 儿子联通块，答案即连边之和。

$\mathcal{O}(n \log n \log SIZE)$，其中 $SIZE$ 为值域大小。

分层图

BJWC2012 冻结

一张 n 个点，m 条边的图，经过一条边需要时间。

有 k 次机会使经过一条边的时间减半，问从 1 走到 n 的最短时间。

$1 \leq n, k \leq 50, 1 \leq m \leq 1000$

Solution

入门题，大家应该都做过。

Solution

入门题，大家应该都做过。

将 n 个点拆成 $n(k+1)$ 个点，每个点 $(i,j)(1 \le i \le n, 0 \le j \le k)$ 表示走到编号为 i 的点时用了 j 次减半机会。

Solution

入门题，大家应该都做过。

将 n 个点拆成 $n(k+1)$ 个点，每个点 $(i,j)(1 \le i \le n, 0 \le j \le k)$ 表示走到编号为 i 的点时用了 j 次减半机会。

实际上就是一张 n 个点的图复制成了 $(k+1)$ 层，这也是分层图名字的由来。

Solution

入门题，大家应该都做过。

将 n 个点拆成 $n(k+1)$ 个点，每个点 $(i,j)(1 \le i \le n, 0 \le j \le k)$ 表示走到编号为 i 的点时用了 j 次减半机会。

实际上就是一张 n 个点的图复制成了 $(k+1)$ 层，这也是分层图名字的由来。

每一层之间都需连上所有 m 条边，相邻层之间连上减半的边。

Solution

入门题，大家应该都做过。

将 n 个点拆成 $n(k+1)$ 个点，每个点 $(i,j)(1 \le i \le n, 0 \le j \le k)$ 表示走到编号为 i 的点时用了 j 次减半机会。

实际上就是一张 n 个点的图复制成了 $(k+1)$ 层，这也是分层图名字的由来。

每一层之间都需连上所有 m 条边，相邻层之间连上减半的边。

直接 Dijkstra 即可。

Solution

入门题，大家应该都做过。

将 n 个点拆成 $n(k+1)$ 个点，每个点 $(i,j)(1 \le i \le n, 0 \le j \le k)$ 表示走到编号为 i 的点时用了 j 次减半机会。

实际上就是一张 n 个点的图复制成了 $(k+1)$ 层，这也是分层图名字的由来。

每一层之间都需连上所有 m 条边，相邻层之间连上减半的边。

直接 Dijkstra 即可。

时间复杂度 $\mathcal{O}(k(n+m)\log(nk))$，空间复杂度 $\mathcal{O}(k(n+m))$。

CF1137C Museums Tour

给定一个 n 个点 m 条边的**有向**图（无重边自环），定义一次周期为 d 天。

现在你会在一个周期的第一天从 1 号点开始行走，每经过一条边，就会花费一天时间；你**不能**在一个点逗留不走，但是所有点你都可以重复经过。

每个点在一个周期中，会有部分时间开放，而其他时间不开放；注意，**不开放不意味着你不能走到这个点**。

求你能访问到的最多的位于开放时间的点（同样的点只算一次）。

$1 \leq n, m \leq 10^5, 1 \leq d \leq 50$

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

由于本人比较菜，所以直接舍掉了直接 DP 的做法，往建模想去了。这里有一份直接 DP 做法（可以点）。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

由于本人比较菜，所以直接舍掉了直接 DP 的做法，往建模想去了。这里有一份直接 DP 做法（可以点）。

这个 d 的范围启发了一个算法：分层图。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

由于本人比较菜，所以直接舍掉了直接 DP 的做法，往建模想去了。这里有一份直接 DP 做法（可以点）。

这个 d 的范围启发了一个算法：分层图。

于是拆为 nd 个点，一个点 $(i,j)(1 \le i \le n, 1 \le j \le d)$ 表示在 i 点，当前是一个周期的第 j 天，且若第 i 个点在第 j 天开放，(i,j) 这个点点权为 1；否则为 0。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

由于本人比较菜，所以直接舍掉了直接 DP 的做法，往建模想去了。这里有一份直接 DP 做法（可以点）。

这个 d 的范围启发了一个算法：分层图。

于是拆为 nd 个点，一个点 $(i,j)(1 \le i \le n, 1 \le j \le d)$ 表示在 i 点，当前是一个周期的第 j 天，且若第 i 个点在第 j 天开放，(i,j) 这个点点权为 1；否则为 0。

于是将图复制 d 层，相邻层间连边（特别地，将第 d 层与第 1 层视作相邻，因为周期会循环）。

Solution

很容易猜到正解复杂度应当为 $\mathcal{O}(nd)$，看起来就很像缩点 $+$ DP。

但是这个天数很不好处理，以某个天数进入一个强连通分量，再乱走走出来的天数都不一样，答案也不一样。

由于本人比较菜，所以直接舍掉了直接 DP 的做法，往建模想去了。这里有一份直接 DP 做法（可以点）。

这个 d 的范围启发了一个算法：分层图。

于是拆为 nd 个点，一个点 $(i,j)(1 \le i \le n, 1 \le j \le d)$ 表示在 i 点，当前是一个周期的第 j 天，且若第 i 个点在第 j 天开放，(i,j) 这个点点权为 1；否则为 0。

于是将图复制 d 层，相邻层间连边（特别地，将第 d 层与第 1 层视作相邻，因为周期会循环）。

缩点后直接求最大权值路径。

Solution

为什么不会算重?

Solution

为什么不会算重?

考虑重复统计的情况，即存在一条路径使得 (i, j_1) 可以到达 (i, j_2) 但不在同一个强连通分量中，这是不可能的，因为单独看这条路径经过的顶点编号序列，沿着这个序列再重复走 $d-1$ 遍一定能回到 (i, j_1)。

为什么不会算重?

考虑重复统计的情况，即存在一条路径使得 (i, j_1) 可以到达 (i, j_2) 但不在同一个强连通分量中，这是不可能的，因为单独看这条路径经过的顶点编号序列，沿着这个序列再重复走 $d-1$ 遍一定能回到 (i, j_1)。

这意味着如果同一顶点拆出的两点弱连通，它们必然也强连通，所以不需要担心重复统计的情况。

为什么不会算重?

考虑重复统计的情况，即存在一条路径使得 (i, j_1) 可以到达 (i, j_2) 但不在同一个强连通分量中，这是不可能的，因为单独看这条路径经过的顶点编号序列，沿着这个序列再重复走 $d-1$ 遍一定能回到 (i, j_1)。

这意味着如果同一顶点拆出的两点弱连通，它们必然也强连通，所以不需要担心重复统计的情况。

（实际上上面这两句话是从博客那搬过来的，讲的比我讲的好一些，~~我只会显然~~）

Solution

为什么不会算重?

考虑重复统计的情况，即存在一条路径使得 (i, j_1) 可以到达 (i, j_2) 但不在同一个强连通分量中，这是不可能的，因为单独看这条路径经过的顶点编号序列，沿着这个序列再重复走 $d-1$ 遍一定能回到 (i, j_1)。

这意味着如果同一顶点拆出的两点弱连通，它们必然也强连通，所以不需要担心重复统计的情况。

（实际上上面这两句话是从博客那搬过来的，讲的比我讲的好一些，~~我只会显然~~）

另外注意每个点 i 可能有几天 $j_{1\dots k}$ 在同一个强连通分量中，此时他们只能算一次点权，所以缩点时不能直接以内部所有点的点权和来作为这个强连通分量的点权，还需要略微处理一下。

Solution

为什么不会算重?

考虑重复统计的情况，即存在一条路径使得 (i, j_1) 可以到达 (i, j_2) 但不在同一个强连通分量中，这是不可能的，因为单独看这条路径经过的顶点编号序列，沿着这个序列再重复走 $d-1$ 遍一定能回到 (i, j_1)。

这意味着如果同一顶点拆出的两点弱连通，它们必然也强连通，所以不需要担心重复统计的情况。

（实际上上面这两句话是从博客那搬过来的，讲的比我讲的好一些，~~我只会显然~~）

另外注意每个点 i 可能有几天 $j_{1\dots k}$ 在同一个强连通分量中，此时他们只能算一次点权，所以缩点时不能直接以内部所有点的点权和来作为这个强连通分量的点权，还需要略微处理一下。

$\mathcal{O}((n+m)d)$

强连通分量

BZOJ2438 杀人游戏

一位冷血的杀手潜入 Na-wiat，并假装成平民。警察希望能在 n 个人里面，查出谁是杀手。

警察能够对每一个人进行查证，假如查证的对象是平民，他会告诉警察，他认识的人，谁是杀手，谁是平民。假如查证的对象是杀手，杀手将会把警察干掉。

现在警察掌握了每一个人认识谁，有 m 条认识关系。每一个人都有可能是杀手，可看作他们是杀手的概率是相同的。问：根据最优的情况，保证警察自身安全并知道谁是杀手的概率最大是多少？

$1 \leq n, m \leq 3 \times 10^5$

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

仔细思考可以发现：在一个强连通分量中的点，只需要查一个人，之后便可以安全地把这个强连通分量查遍。

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

仔细思考可以发现：在一个强连通分量中的点，只需要查一个人，之后便可以安全地把这个强连通分量查遍。

所以我们缩完点变成几个分离的 DAG, 在这些互不联通的 DAG 中，只需要查询入度为零的那些点，入度不为零的都可以通过入度为 0 的点查过来。

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

仔细思考可以发现：在一个强连通分量中的点，只需要查一个人，之后便可以安全地把这个强连通分量查遍。

所以我们缩完点变成几个分离的 DAG, 在这些互不联通的 DAG 中，只需要查询入度为零的那些点，入度不为零的都可以通过入度为 0 的点查过来。

但还有细节：题目确保有且只有一个杀手，所以如果把除一个人之外的人都查遍了，那么这个人显然就是杀手。

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

仔细思考可以发现：在一个强连通分量中的点，只需要查一个人，之后便可以安全地把这个强连通分量查遍。

所以我们缩完点变成几个分离的 DAG, 在这些互不联通的 DAG 中，只需要查询入度为零的那些点，入度不为零的都可以通过入度为 0 的点查过来。

但还有细节：题目确保有且只有一个杀手，所以如果把除一个人之外的人都查遍了，那么这个人显然就是杀手。

所以说，如果缩完点后，有一个入度为零的点的大小为 1，且所有它连向的点不是只有它连过去（相当于它连向的点入度全部大于 1），那么就可以不用查这个点，把其他的查完了之后他的结果自然就出来了，所以这时查的人可以少一个，但最多也只能少一个（如果有两个人，还是无法确定）

Solution

概率问题很好解决：假设最优解中最少要查询 k 个人，答案即 $\frac{n-k}{n}$。

仔细思考可以发现：在一个强连通分量中的点，只需要查一个人，之后便可以安全地把这个强连通分量查遍。

所以我们缩完点变成几个分离的 DAG, 在这些互不联通的 DAG 中，只需要查询入度为零的那些点，入度不为零的都可以通过入度为 0 的点查过来。

但还有细节：题目确保有且只有一个杀手，所以如果把除一个人之外的人都查遍了，那么这个人显然就是杀手。

所以说，如果缩完点后，有一个入度为零的点的大小为 1，且所有它连向的点不是只有它连过去（相当于它连向的点入度全部大于 1），那么就可以不用查这个点，把其他的查完了之后他的结果自然就出来了，所以这时查的人可以少一个，但最多也只能少一个（如果有两个人，还是无法确定）

$\mathcal{O}(n+m)$

二分图

CF1139E Maximize Mex

在一所学校里有 n 名学生和 m 个社团，社团被编号为 $1 \ldots m$ 。第 i 个学生有一个能力值 p_i，且属于社团 c_i（每个学生恰好属于一个社团）。

学校将要举行一个为期 d 天的活动，每天学校要举行一场程序设计比赛——校长将会从每个社团中**各**选出一个人（如果某个社团没有人，就不选）组成一个队，令队里的学生的能力值的集合为 S，则该队的总能力值为 $\operatorname{mex}(S)$。

$\operatorname{mex}(S)$ 表示 S 中没有出现的最小的非负整数。例如：

$\operatorname{mex}(\{0,1,3,4\}) = 2; \operatorname{mex}(\{1,2,3\}) = 0; \operatorname{mex}(\{0,1,3,4\}) = 2; \operatorname{mex}(\{1,2,3\}) = 0$

但是由于学业繁忙，第 i 天时，第 k_i 个学生会离开社团（在校长选这一天参加比赛的学生之前）。

校长希望知道对于活动的每一天，他可能选出的队伍的总能力值最大是多少。

$1 \leq n, m \leq 5000, 0 \leq p_i < 5000$。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

于是要求 mex，只需要每次从小到大枚举能力值，跑匈牙利算法的匹配，只要匹配不了就停下，匹配不了的数值就是最大的 mex 值。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

于是要求 mex，只需要每次从小到大枚举能力值，跑匈牙利算法的匹配，只要匹配不了就停下，匹配不了的数值就是最大的 mex 值。

但这个删边操作并不好搞，如果每次删都重新从小到大匹配，复杂度 $O(nmd)$，无法承受。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

于是要求 mex，只需要每次从小到大枚举能力值，跑匈牙利算法的匹配，只要匹配不了就停下，匹配不了的数值就是最大的 mex 值。

但这个删边操作并不好搞，如果每次删都重新从小到大匹配，复杂度 $O(nmd)$，无法承受。

正难则反，观察样例并思考很容易发现答案非严格单调递减，于是考虑倒过来加边。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

于是要求 mex，只需要每次从小到大枚举能力值，跑匈牙利算法的匹配，只要匹配不了就停下，匹配不了的数值就是最大的 mex 值。

但这个删边操作并不好搞，如果每次删都重新从小到大匹配，复杂度 $O(nmd)$，无法承受。

正难则反，观察样例并思考很容易发现答案非严格单调递减，于是考虑倒过来加边。

每加入一条边便从刚刚匹配失败的地方继续重复匹配过程，直到再次失败为止，于是又得到了这一次的答案。

Solution

每个能力值至少被一个社团提供，每一个社团至多提供一个能力值，再加上 mex，可以启发我们想到二分图。

于是由此构造二分图，左侧是能力值，右侧是社团。每有一个能力值 p_i，属于社团 c_i 的人，便从左侧 p_i 到右侧 c_i 连一条边。

由于匈牙利算法每次只会找增广路，对于当前能力值 t，无论匹配成功还是失败，t 以下的能力值都不会由可以匹配被修改为不能匹配。

于是要求 mex，只需要每次从小到大枚举能力值，跑匈牙利算法的匹配，只要匹配不了就停下，匹配不了的数值就是最大的 mex 值。

但这个删边操作并不好搞，如果每次删都重新从小到大匹配，复杂度 $O(nmd)$，无法承受。

正难则反，观察样例并思考很容易发现答案非严格单调递减，于是考虑倒过来加边。

每加入一条边便从刚刚匹配失败的地方继续重复匹配过程，直到再次失败为止，于是又得到了这一次的答案。

$\mathcal{O}((n+q)m)$

一些杂题

平面图

老虎和蒜头是好朋友。

这一天，老虎和蒜头又聚集到一起讨论有趣的问题了。与往常类似地，蒜头又不知道从哪里摸出了一个 n 个点，m 条边的无向平面图。讨论完毕后，蒜头提议进行一个游戏：轮流给该图加上一条边，直到无论增加上哪一条边图都会变得不是平面图，无法操作者败。

游戏结束。毫无疑问，老虎又输了，然而老虎现在关心的并不是这样的问题。老虎最近刚刚学习过三分图理论，因此他想要知道，有多少种加边的方案（不能加重边和自环），使得加边后的图仍然是一个三分图。

一个图被称为三分图，当且仅当存在一个方案，可以将图划分成三个点集，使得不存在任意一条边在一个点集内部。

保证输入给出的平面图再任意加一条边后都不再是平面图。

$5 \leq n \leq 10^5$

Bonus: $5 \leq n \leq 10^6$

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

而且继续思考容易发现，这个染色方案是唯一的。

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

而且继续思考容易发现，这个染色方案是唯一的。

于是设 a, b, c 分别表示三种颜色点的个数，答案即 $ab + bc + ac - m$。

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

而且继续思考容易发现，这个染色方案是唯一的。

于是设 a, b, c 分别表示三种颜色点的个数，答案即 $ab + bc + ac - m$。

注意虽然 $k(k > 2)$ 分图染色是 NP 问题，但是在这个特殊的图上可以做到 $\mathcal{O}(n)$ 染色。

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

而且继续思考容易发现，这个染色方案是唯一的。

于是设 a, b, c 分别表示三种颜色点的个数，答案即 $ab + bc + ac - m$。

注意虽然 $k(k > 2)$ 分图染色是 NP 问题，但是在这个特殊的图上可以做到 $\mathcal{O}(n)$ 染色。

原数据范围为 10^5 是因为标算貌似用了一个启发式的 $\log^2$ 的染色方法，实际上~~有人爆碾标算~~可以做到更好：

Solution

有一个小 trick ：极大平面图一定构成三角剖分。

又因为两边相连的点不能分在同一组，所以只要确定了这个图中任意一条边相连的两个点的颜色，那么就能唯一确定其他点的颜色。

而且继续思考容易发现，这个染色方案是唯一的。

于是设 a, b, c 分别表示三种颜色点的个数，答案即 $ab + bc + ac - m$。

注意虽然 $k(k > 2)$ 分图染色是 NP 问题，但是在这个特殊的图上可以做到 $\mathcal{O}(n)$ 染色。

原数据范围为 10^5 是因为标算貌似用了一个启发式的 $\log^2$ 的染色方法，实际上~~有人爆碾标算~~可以做到更好：

首先随意标记两个点的颜色，然后进行 BFS，每次都将这个点相邻的节点标记为**不是**它这个颜色；如果被标记的点被标记了不是两种颜色，那么它一定是剩下的那种，于是入队。

The End

Thanks!