网络流选讲

rotcar07

2024 年 8 月 4 日

目录

FAQ

FAQ

FAQ

Q: 你是谁?

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅。**

Q: 我是谁?

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

Q: 我是谁?

A: 你也许是传奇耐假王**诶丝尅**。

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

Q: 我是谁?

A: 你也许是传奇耐假王**诶丝尅**。

Q: 网络流能干什么?

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

Q: 我是谁?

A: 你也许是传奇耐假王**诶丝尅**。

Q: 网络流能干什么?

A: 能从理论上帮助传奇耐表王**诶丝尅**找到嗯批歪。

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

Q: 我是谁?

A: 你也许是传奇耐假王**诶丝尅**。

Q: 网络流能干什么?

A: 能从理论上帮助传奇耐表王**诶丝尅**找到嗯批歪。

Q: 能不能请**诶丝尅**先生锐评一下网络流?

FAQ

Q: 你是谁?

A: 我不是传奇耐罚王**诶丝尅**。

Q: 我是谁?

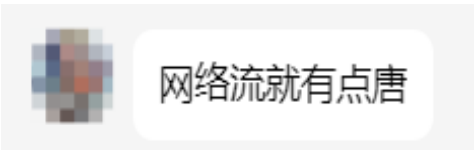
A: 你也许是传奇耐假王**诶丝尅**。

Q: 网络流能干什么?

A: 能从理论上帮助传奇耐表王**诶丝尅**找到嗯批歪。

Q: 能不能请**诶丝尅**先生锐评一下网络流?

A:



网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络上的**流**，可看作是从 E 到实数集或整数集的函数 f，表示边上的流量。

网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络上的**流**，可看作是从 E 到实数集或整数集的函数 f，表示边上的流量。

对于边 $(u, v) \in E$，有 $0 \leq f(u, v) \leq c(u, v)$。

网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络上的**流**，可看作是从 E 到实数集或整数集的函数 f，表示边上的流量。

对于边 $(u, v) \in E$，有 $0 \leq f(u, v) \leq c(u, v)$。

定义点 $x \in V$ 的净流量 $f(x) = \sum_{(x,y)\in E} f(x, y) - \sum_{(y,x)\in E} f(y, x)$。即出的总流量减去入的总流量。

网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络上的**流**，可看作是从 E 到实数集或整数集的函数 f，表示边上的流量。

对于边 $(u, v) \in E$，有 $0 \leq f(u, v) \leq c(u, v)$。

定义点 $x \in V$ 的净流量 $f(x) = \sum_{(x,y)\in E} f(x, y) - \sum_{(y,x)\in E} f(y, x)$。即出的总流量减去入的总流量。

则对点集中除源汇点外任意一点 x 均有 $f(x) = 0$。

网络是特殊的有向图 $G = (V, E)$，边 $(u, v) \in E$ 都有容量 $c(u, v)$。V 中存在源点 s 和汇点 t，$s \neq t$。

网络上的**流**，可看作是从 E 到实数集或整数集的函数 f，表示边上的流量。

对于边 $(u, v) \in E$，有 $0 \leq f(u, v) \leq c(u, v)$。

定义点 $x \in V$ 的净流量 $f(x) = \sum_{(x,y)\in E} f(x, y) - \sum_{(y,x)\in E} f(y, x)$。即出的总流量减去入的总流量。

则对点集中除源汇点外任意一点 x 均有 $f(x) = 0$。

$f(s)$ 为流 f 的**流量** $|f|$。显然有 $f(s) = -f(t)$。

最大流问题要求网络 $G = (V, E)$ 中流量最大的流。

最大流问题要求网络 $G = (V, E)$ 中流量最大的流。

对于网络 $G(V, E)$ 和 G 上的流 f，做出如下定义：

最大流问题要求网络 $G = (V, E)$ 中流量最大的流。

对于网络 $G(V, E)$ 和 G 上的流 f，做出如下定义：

定义边 $(u, v) \in E$ 的**剩余流量** $c_f(u, v) = c(u, v) - f(u, v)$。

最大流问题要求网络 $G=(V,E)$ 中流量最大的流。

对于网络 $G(V,E)$ 和 G 上的流 f，做出如下定义：

定义边 $(u,v)\in E$ 的**剩余流量** $c_f(u,v)=c(u,v)-f(u,v)$。

将所有 $c_f(u,v)>0$ 的边形成的边集 E_f 与 V 结合形成的子图 $G_f(V,E_f)$ 称为**残量网络**。

最大流问题要求网络 $G=(V,E)$ 中流量最大的流。

对于网络 $G(V,E)$ 和 G 上的流 f，做出如下定义：

定义边 $(u,v)\in E$ 的**剩余流量** $c_f(u,v)=c(u,v)-f(u,v)$。

将所有 $c_f(u,v)>0$ 的边形成的边集 E_f 与 V 结合形成的子图 $G_f(V,E_f)$ 称为**残量网络**。

称在 G_f 上的一条从 s 到 t 的路径为一条**增广路**。

最大流问题要求网络 $G=(V,E)$ 中流量最大的流。

对于网络 $G(V,E)$ 和 G 上的流 f，做出如下定义：

定义边 $(u,v)\in E$ 的**剩余流量** $c_f(u,v)=c(u,v)-f(u,v)$。

将所有 $c_f(u,v)>0$ 的边形成的边集 E_f 与 V 结合形成的子图 $G_f(V,E_f)$ 称为**残量网络**。

称在 G_f 上的一条从 s 到 t 的路径为一条**增广路**。

给增广路上的每条边 (u,v) 加上等量的流量的过程称为**增广**。则流可看做若干增广的叠加。

对于每一条边 $(u, v) \in E$，新建一条**反向边** (v, u)，并定义 $f(v, u) = -f(u, v)$，即 $c_f(v, u) = f(u, v)$。

对于每一条边 $(u, v) \in E$，新建一条**反向边** (v, u)，并定义 $f(v, u) = -f(u, v)$，即 $c_f(v, u) = f(u, v)$。

然后将 $c_f > 0$ 的反向边也引入残量网络中。

对于每一条边 $(u, v) \in E$，新建一条**反向边** (v, u)，并定义 $f(v, u) = -f(u, v)$，即 $c_f(v, u) = f(u, v)$。

然后将 $c_f > 0$ 的反向边也引入残量网络中。

考虑此时增广的实际意义。若增广到反向边 (v, u)，实际上意味着 $f(u, v)$ 在减小，是合法的**退流**操作。

对于每一条边 $(u, v) \in E$，新建一条**反向边** (v, u)，并定义 $f(v, u) = -f(u, v)$，即 $c_f(v, u) = f(u, v)$。

然后将 $c_f > 0$ 的反向边也引入残量网络中。

考虑此时增广的实际意义。若增广到反向边 (v, u)，实际上意味着 $f(u, v)$ 在减小，是合法的**退流**操作。

引入反向边后，我们就有了**反悔**的策略。

对于每一条边 $(u, v) \in E$，新建一条**反向边** (v, u)，并定义 $f(v, u) = -f(u, v)$，即 $c_f(v, u) = f(u, v)$。

然后将 $c_f > 0$ 的反向边也引入残量网络中。

考虑此时增广的实际意义。若增广到反向边 (v, u)，实际上意味着 $f(u, v)$ 在减小，是合法的**退流**操作。

引入反向边后，我们就有了**反悔**的策略。

我们不加证明地指出，若持续在当前的 G_f 上增广直到找不到增广路，最终叠加形成的流即为最大流。

问题在于如何快速求出增广路并增广。此处引入 dinic 算法。

问题在于如何快速求出增广路并增广。此处引入 dinic 算法。

对 G_f BFS 求得所有点 $u \in V$ 到 s 的距离 $d(u)$，然后对 V 按照 d 分层。若 $d(t)$ 不存在则不存在增广。

问题在于如何快速求出增广路并增广。此处引入 dinic 算法。

对 G_f BFS 求得所有点 $u \in V$ 到 s 的距离 $d(u)$，然后对 V 按照 d 分层。若 $d(t)$ 不存在则不存在增广。

此时强制令经过每个点的流量只能流向下一层。即对于所有边 $(u, v) \in E_f$，仅保留 $d(v) = d(u) + 1$ 的边。称形成的网络 $G_L(V, E_L)$ 为**层次图**。

问题在于如何快速求出增广路并增广。此处引入 dinic 算法。

对 G_f BFS 求得所有点 $u \in V$ 到 s 的距离 $d(u)$，然后对 V 按照 d 分层。若 $d(t)$ 不存在则不存在增广。

此时强制令经过每个点的流量只能流向下一层。即对于所有边 $(u, v) \in E_f$，仅保留 $d(v) = d(u) + 1$ 的边。称形成的网络 $G_L(V, E_L)$ 为**层次图**。

记 G_L 上流量最大的流为**阻塞流**。

问题在于如何快速求出增广路并增广。此处引入 dinic 算法。

对 G_f BFS 求得所有点 $u \in V$ 到 s 的距离 $d(u)$，然后对 V 按照 d 分层。若 $d(t)$ 不存在则不存在增广。

此时强制令经过每个点的流量只能流向下一层。即对于所有边 $(u, v) \in E_f$，仅保留 $d(v) = d(u) + 1$ 的边。称形成的网络 $G_L(V, E_L)$ 为**层次图**。

记 G_L 上流量最大的流为**阻塞流**。

则 dinic 算法可简单分为三步，BFS 分层，DFS 找阻塞流，更新最大流与残量网络。重复此过程直到不存在增广。

为保证 dinic 复杂度正确，需引入**当前弧优化**。

为保证 dinic 复杂度正确，需引入**当前弧优化**。

注意到若 G_L 中存在点 u，出边和入边数量都很大，那么每次 DFS 在 u 上会产生平方级别的复杂度。

为保证 dinic 复杂度正确，需引入**当前弧优化**。

注意到若 G_L 中存在点 u，出边和入边数量都很大，那么每次 DFS 在 u 上会产生平方级别的复杂度。

有些出边已无剩余流量，或者该出边无法走向 t，这些边是无需尝试的。

为保证 dinic 复杂度正确，需引入**当前弧优化**。

注意到若 G_L 中存在点 u，出边和入边数量都很大，那么每次 DFS 在 u 上会产生平方级别的复杂度。

有些出边已无剩余流量，或者该出边无法走向 t，这些边是无需尝试的。

所以对于每个点维护第一个有必要尝试的出边的指针即可。

为保证 dinic 复杂度正确，需引入**当前弧优化**。

注意到若 G_L 中存在点 u，出边和入边数量都很大，那么每次 DFS 在 u 上会产生平方级别的复杂度。

有些出边已无剩余流量，或者该出边无法走向 t，这些边是无需尝试的。

所以对于每个点维护第一个有必要尝试的出边的指针即可。

记 $n = |V|, m = |E|$，则 dinic 单轮增广的复杂度为 $O(nm)$，增广轮数 $O(n)$，总时间复杂度 $O(n^2m)$。

实际上在特殊图里 dinic 的算法可以被分析到更优。

实际上在特殊图里 dinic 的算法可以被分析到更优。

在单位容量网络里（$\forall (u,v) \in E, c_{(u,v)} \in \{0,1\}$），dinic 增广轮数是 $O(\min(m^{1/2}, n^{2/3}))$，单轮增广复杂度是 $O(m)$。

实际上在特殊图里 dinic 的算法可以被分析到更优。

在单位容量网络里（$\forall (u,v) \in E, c_{(u,v)} \in \{0,1\}$），dinic 增广轮数是 $O(\min(m^{1/2}, n^{2/3}))$，单轮增广复杂度是 $O(m)$。

在单位容量网络里，若对于除源汇点外的每个点均满足入度为 1 或出度为 1，则 dinic 增广轮数是 $O(n^{1/2})$。

在网络 $G = (V, E)$ 上，定义**割**为 V 的一种划分方式 (S, T)，满足 $s \in S, t \in T$。

在网络 $G = (V, E)$ 上，定义**割**为 V 的一种划分方式 (S, T)，满足 $s \in S, t \in T$。

定义割的**容量**为 $||S, T|| = \sum_{(u,v)\in E, u\in S, v\in T} c(u, v)$，即所有从 S 中一点出发到 T 中一点的边的容量和。

在网络 $G = (V, E)$ 上，定义**割**为 V 的一种划分方式 (S, T)，满足 $s \in S, t \in T$。

定义割的**容量**为 $||S, T|| = \sum_{(u,v)\in E, u\in S, v\in T} c(u, v)$，即所有从 S 中一点出发到 T 中一点的边的容量和。

最小割问题要求网络 $G = (V, E)$ 容量最小的割。

在网络 $G = (V, E)$ 上，定义**割**为 V 的一种划分方式 (S, T)，满足 $s \in S, t \in T$。

定义割的**容量**为 $||S, T|| = \sum_{(u,v)\in E, u\in S, v\in T} c(u, v)$，即所有从 S 中一点出发到 T 中一点的边的容量和。

最小割问题要求网络 $G = (V, E)$ 容量最小的割。

更好理解的说法是，断掉容量总和最少的边使得 s 无法走到 t。

最大流最小割定理

对于任意网络 $G(V, E)$，最大流 $|f|$ 等于最小割 $||S, T||$。

最大流最小割定理

对于任意网络 $G(V, E)$，最大流 $|f|$ 等于最小割 $||S, T||$。

证明

$$|f| = f(s) = \sum_{u \in S, v \in V} (f(u, v) - f(v, u))$$

$$= \sum_{u \in S, v \in T} \big(f(u, v) - f(v, u)\big) \leq \sum_{u \in S, v \in T} c_f(u, v) = ||S, T||$$

最大流最小割定理

对于任意网络 $G(V, E)$，最大流 $|f|$ 等于最小割 $||S, T||$。

证明

$$|f| = f(s) = \sum_{u \in S, v \in V} (f(u, v) - f(v, u))$$

$$= \sum_{u \in S, v \in T} \big(f(u, v) - f(v, u)\big) \leq \sum_{u \in S, v \in T} c_f(u, v) = ||S, T||$$

等号成立要求对于流 f 找到一个割 (S, T) 满足对所有 $u \in S, v \in T$，(u, v) 满流，(v, u) 空流。将最后的残量网络中 s 所处的连通块的点集当做 S 即可。

网络 $G(V, E)$ 上每条边上都有一个单位流量费用 $w(u, v)$，每经过一单位流量就要花 $w(u, v)$ 的费用。

网络 $G(V, E)$ 上每条边上都有一个单位流量费用 $w(u, v)$，每经过一单位流量就要花 $w(u, v)$ 的费用。

最小费用最大流问题要求此网络费用最小的最大流 f。

网络 $G(V, E)$ 上每条边上都有一个单位流量费用 $w(u, v)$，每经过一单位流量就要花 $w(u, v)$ 的费用。

最小费用最大流问题要求此网络费用最小的最大流 f。

考虑直接套用最大流的算法，改进如下两点：

- 根据**反悔**策略，反向边的费用应改为相反数。
- 由于求的是最小费用，则将 bfs 改成 bellman-ford 算法求单源最短路即可。

网络 $G(V, E)$ 上每条边上都有一个单位流量费用 $w(u, v)$，每经过一单位流量就要花 $w(u, v)$ 的费用。

最小费用最大流问题要求此网络费用最小的最大流 f。

考虑直接套用最大流的算法，改进如下两点：

- 根据**反悔**策略，反向边的费用应改为相反数。
- 由于求的是最小费用，则将 bfs 改成 bellman-ford 算法求单源最短路即可。

费用流的复杂度是 $O(nmf)$ 的，其中 f 是最大流。

网络 $G(V, E)$ 上每条边上都有一个单位流量费用 $w(u, v)$，每经过一单位流量就要花 $w(u, v)$ 的费用。

最小费用最大流问题要求此网络费用最小的最大流 f。

考虑直接套用最大流的算法，改进如下两点：

- 根据**反悔**策略，反向边的费用应改为相反数。
- 由于求的是最小费用，则将 bfs 改成 bellman-ford 算法求单源最短路即可。

费用流的复杂度是 $O(nmf)$ 的，其中 f 是最大流。

最大费用流即把费用取相反数类似地做一遍。

下面通过一些例题来讲如何针对题目要求建立对应的网络模型。

P2763 试题库问题

诶丝尅先生有一个试题库，中有 n 道试题。每道试题都标明了所属类别。同一道题可能有多个类别属性。现要从题库中抽取 m 道题组成试卷，并对于每个属性钦定选出 c_i 题。

判无解，有解输出方案。

$m = \sum c_i, 2 \leq k \leq 20, k \leq n \leq 10^3$。

以下简记在网络中加入一条从 u 到 v、容量为 $c(u,v)$ 的边为 $(u,v,c(u,v))$。

以下简记在网络中加入一条从 u 到 v、容量为 $c(u,v)$ 的边为 $(u,v,c(u,v))$。

- 建立源点 s，汇点 t。令 a_i 表示第 i 个试题的点，b_i 表示第 i 个类型的点。

以下简记在网络中加入一条从 u 到 v、容量为 $c(u,v)$ 的边为 $(u,v,c(u,v))$。

- 建立源点 s，汇点 t。令 a_i 表示第 i 个试题的点，b_i 表示第 i 个类型的点。
- 建边 $(s,a_i,1)$，表示每个题目最多只能选一次。

以下简记在网络中加入一条从 u 到 v、容量为 $c(u,v)$ 的边为 $(u,v,c(u,v))$。

- 建立源点 s，汇点 t。令 a_i 表示第 i 个试题的点，b_i 表示第 i 个类型的点。
- 建边 $(s,a_i,1)$，表示每个题目最多只能选一次。
- 建边 (b_i,t,c_i)，表示每个类型最多选 c_i 次。

以下简记在网络中加入一条从 u 到 v、容量为 $c(u,v)$ 的边为 $(u,v,c(u,v))$。

- 建立源点 s，汇点 t。令 a_i 表示第 i 个试题的点，b_i 表示第 i 个类型的点。
- 建边 $(s,a_i,1)$，表示每个题目最多只能选一次。
- 建边 (b_i,t,c_i)，表示每个类型最多选 c_i 次。
- 对于每个题目，向它能成为的每个类型连一条容量为 1 的边。

建图后求最大流，若最大流 $=m$ 说明所有 t 的入边均满流，即每个限制都达成了。若 $<m$ 则说明无解。

那么方案怎么求呢?

那么方案怎么求呢?

考察残量网络，若从题目向类型连的某条边满流了，说明这个题目被选用了并成为了这个类型。

P1361 小 M 的作物

诶丝尅开辟了两块巨大的耕地 A 和 B，他有 n 个种子，编号为 1 到 n。

现在，第 i 种作物种植在 A, B 分别有 a_i, b_i 的收益，另有 m 种作物组合，第 i 个组合中的作物共同种在 A 或共同种在 B 中可以分别获得 $c_{1,i}, c_{2,i}$ 的额外收益。

求最大收益。

所有数不超过 1000。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

- 建立源点 s，汇点 t，每个作物的点 u_i。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

- 建立源点 s，汇点 t，每个作物的点 u_i。
- 建边 $(s, u_i, a_i), (u_i, t, b_i)$。种哪边就留哪边。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

- 建立源点 s，汇点 t，每个作物的点 u_i。
- 建边 $(s, u_i, a_i), (u_i, t, b_i)$。种哪边就留哪边。
- 对每个组合新建两个辅助点 A, B，连 $(s, A, c_1), (B, t, c_2)$，对于组合里的每个作物连 $(A, u_i, +\infty), (u_i, B, +\infty)$。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

- 建立源点 s，汇点 t，每个作物的点 u_i。
- 建边 $(s, u_i, a_i), (u_i, t, b_i)$。种哪边就留哪边。
- 对每个组合新建两个辅助点 A, B，连 $(s, A, c_1), (B, t, c_2)$，对于组合里的每个作物连 $(A, u_i, +\infty), (u_i, B, +\infty)$。

其中 $+\infty$ 边表示这条边不能被割。若组合中某个作物种在了 B，那么会产生 $s \to A \to u_i \to t$ 的通路，不满足割的定义，所以必须割掉 (s, A)。所以最小割是正确的。

发现最大流模型无法解决形如“若多个条件同时满足则获得收益”的问题。

所以这里考虑最小割。

- 建立源点 s，汇点 t，每个作物的点 u_i。
- 建边 $(s, u_i, a_i), (u_i, t, b_i)$。种哪边就留哪边。
- 对每个组合新建两个辅助点 A, B，连 $(s, A, c_1), (B, t, c_2)$，对于组合里的每个作物连 $(A, u_i, +\infty), (u_i, B, +\infty)$。

其中 $+\infty$ 边表示这条边不能被割。若组合中某个作物种在了 B，那么会产生 $s \to A \to u_i \to t$ 的通路，不满足割的定义，所以必须割掉 (s, A)。所以最小割是正确的。

最后用 $\sum(a_i + b_i) + \sum(c_{1,i} + c_{2,i})$ 减去最小割就行。

P1251 餐巾计划问题

诶丝尅餐厅在 N 天里第 i 天需要 r_i 块餐巾。餐厅可以花 p 购买新的餐巾；或者花 f 洗一块旧餐巾，需 m 天；或者花 s 洗一块，需要 n 天。

每天洗好的餐巾和购买的新餐巾数之和，要满足当天的需求量。

求最小花费。$1 \leq N \leq 2 \times 10^3$。

问题多了一维，涉及到费用了。考虑费用流。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。
- 建边 $(a_i, t, r_i, 0)$ 表示每天需要贡献 r_i 条好餐巾。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。
- 建边 $(a_i, t, r_i, 0)$ 表示每天需要贡献 r_i 条好餐巾。
- $(s, b_i, r_i, 0)$ 表示每天可以获得 r_i 条脏餐巾。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。
- 建边 $(a_i, t, r_i, 0)$ 表示每天需要贡献 r_i 条好餐巾。
- $(s, b_i, r_i, 0)$ 表示每天可以获得 r_i 条脏餐巾。
- $(b_i, b_{i+1}, +\infty, 0)$ 表示可以留餐巾给以后洗。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。
- 建边 $(a_i, t, r_i, 0)$ 表示每天需要贡献 r_i 条好餐巾。
- $(s, b_i, r_i, 0)$ 表示每天可以获得 r_i 条脏餐巾。
- $(b_i, b_{i+1}, +\infty, 0)$ 表示可以留餐巾给以后洗。
- $(s, a_i, +\infty, p), (b_i, a_{i+m}, +\infty, f), (b_i, a_{i+n}, +\infty, s)$ 分别表示三种获取好餐巾的方式。

问题多了一维，涉及到费用了。考虑费用流。

- 建立 s, t， a_i, b_i 分别表示源汇点，每天的好餐巾和每天结束后的脏餐巾。
- 建边 $(a_i, t, r_i, 0)$ 表示每天需要贡献 r_i 条好餐巾。
- $(s, b_i, r_i, 0)$ 表示每天可以获得 r_i 条脏餐巾。
- $(b_i, b_{i+1}, +\infty, 0)$ 表示可以留餐巾给以后洗。
- $(s, a_i, +\infty, p), (b_i, a_{i+m}, +\infty, f), (b_i, a_{i+n}, +\infty, s)$ 分别表示三种获取好餐巾的方式。

这样，求一遍最小费用最大流。由于最大流保证了出边全满流(否则可以通过 $s \to a_i \to t$ 这条增广路继续增广)，所以每天的要求全满足，则求得的费用即为最小费用。

P3355 骑士共存问题

$n \times n$ 的国际象棋棋盘，有些格子有障碍，不能放置棋子。
诶丝尅问你能够在棋盘上放置多少个骑士（走日），使得任两个骑士不能互相攻击。
$1 \leq n \leq 200$。

由于攻击的定义是双向的，如果直接建图会出现一车二元环。所以不能 naive 的建边。

由于攻击的定义是双向的，如果直接建图会出现一车二元环。所以不能 naive 的建边。

考虑棋盘染色！即将 $2|x+y$ 的 (x,y) 染成白色，另外的染成黑色。

由于攻击的定义是双向的，如果直接建图会出现一车二元环。所以不能 naive 的建边。

考虑棋盘染色！即将 $2|x+y$ 的 (x,y) 染成白色，另外的染成黑色。

发现一个点能攻击到的只有异色点。这是一个二分图。

由于攻击的定义是双向的，如果直接建图会出现一车二元环。所以不能 naive 的建边。

考虑棋盘染色！即将 $2|x+y$ 的 (x,y) 染成白色，另外的染成黑色。

发现一个点能攻击到的只有异色点。这是一个二分图。

那么一种颜色连源点，权值为 1；另一种连汇点，权值为 1；可攻击的点中间连 $+\infty$ 的边。有障碍的点直接扔了。

由于攻击的定义是双向的，如果直接建图会出现一车二元环。所以不能 naive 的建边。

考虑棋盘染色！即将 $2|x+y$ 的 (x,y) 染成白色，另外的染成黑色。

发现一个点能攻击到的只有异色点。这是一个二分图。

那么一种颜色连源点，权值为 1；另一种连汇点，权值为 1；可攻击的点中间连 $+\infty$ 的边。有障碍的点直接扔了。

跑一遍最小割即可。

CF786E ALT

诶丝尅给你一棵 n 个节点的树，每个边上有一个守卫。有 m 个居民，每个居民有一个散步路径（两个节点的树上最短路）。一个居民高兴当且仅当他获得了一个宠物或者他散步的路径上所有的守卫都有宠物。宠物可以分配给居民或者守卫者。求最少需要几只宠物才能让所有居民高兴。输出方案。

$n, m \leq 2000$。

考虑朴素的二分图建模，源点到居民连 1 的边，居民到对应的边连 $+\infty$ 的边，边到源点连 1 的边，则答案等于该图最小割。

考虑朴素的二分图建模，源点到居民连 1 的边，居民到对应的边连 $+\infty$ 的边，边到源点连 1 的边，则答案等于该图最小割。

这样建图是 $O(n^2)$ 的。考虑树剖套线段树优化建图，则优化成 $O(n\log^2 n)$。

考虑朴素的二分图建模，源点到居民连 1 的边，居民到对应的边连 $+\infty$ 的边，边到源点连 1 的边，则答案等于该图最小割。

这样建图是 $O(n^2)$ 的。考虑树剖套线段树优化建图，则优化成 $O(n\log^2 n)$。

容易转化为单位容量网络，且所有点满足入度或出度 $=1$，则直接跑 dinic 的复杂度是 $O(n^{1.5}\log^2 n)$，可以通过。

考虑朴素的二分图建模，源点到居民连 1 的边，居民到对应的边连 $+\infty$ 的边，边到源点连 1 的边，则答案等于该图最小割。

这样建图是 $O(n^2)$ 的。考虑树剖套线段树优化建图，则优化成 $O(n\log^2 n)$。

容易转化为单位容量网络，且所有点满足入度或出度 $=1$，则直接跑 dinic 的复杂度是 $O(n^{1.5}\log^2 n)$，可以通过。

方案的输出是容易的，只要看 s 的出边就可以得知哪些居民获得了宠物，从而可以推出哪些守卫获得了宠物。

考虑朴素的二分图建模，源点到居民连 1 的边，居民到对应的边连 $+\infty$ 的边，边到源点连 1 的边，则答案等于该图最小割。

这样建图是 $O(n^2)$ 的。考虑树剖套线段树优化建图，则优化成 $O(n\log^2 n)$。

容易转化为单位容量网络，且所有点满足入度或出度 $=1$，则直接跑 dinic 的复杂度是 $O(n^{1.5}\log^2 n)$，可以通过。

方案的输出是容易的，只要看 s 的出边就可以得知哪些居民获得了宠物，从而可以推出哪些守卫获得了宠物。

实际上，对路径树剖之后，除了最上面的链外，之后的链都是一个重链的前缀。即对重链每个前缀建个结点。这样复杂度少支 $\log$。

SOJ1876 古拉符

在诶丝尅星球上，有 n 个国家，每个国家的综合国力为 a_i。
这个星球上的飞机航线规划非常奇怪，具体来说，对于任意 $i < j$，从 i 国家到 j 国家存在一条单向航线当且仅当这两个国家满足其中至少一条性质：

- $|a_i - a_j| = A$。
- $a_i = a_j \times B$ 或 $a_j = a_i \times B$。
- $a_i \equiv a_j \pmod{C}$。

其中 A, B, C 均为给定的常数，$C \neq 0$。
现在诶丝尅想要进行 k 次环游世界的旅行。每次旅行的起点任意，但是之后只能通过这些航线旅行，且他不愿意经过重复的国家。因此，他想要知道他最多能够访问多少个国家。

考虑使用流量来记录旅行总次数，则限定最大流必须为 k，使用费用来记录国家数。

考虑使用流量来记录旅行总次数，则限定最大流必须为 k，使用费用来记录国家数。

在网络流中一个常用的技巧是拆点。具体的，将每个国家拆成入点和出点，中间连一条容量、费用均为 1 的边。

考虑使用流量来记录旅行总次数，则限定最大流必须为 k，使用费用来记录国家数。

在网络流中一个常用的技巧是拆点。具体的，将每个国家拆成入点和出点，中间连一条容量、费用均为 1 的边。

发现这样做保证了每个国家最多经过一次。之后按照题目中的关系，若 u 可达 v，则用 u 的出边连 v 的入边。

考虑使用流量来记录旅行总次数，则限定最大流必须为 k，使用费用来记录国家数。

在网络流中一个常用的技巧是拆点。具体的，将每个国家拆成入点和出点，中间连一条容量、费用均为 1 的边。

发现这样做保证了每个国家最多经过一次。之后按照题目中的关系，若 u 可达 v，则用 u 的出边连 v 的入边。

最后源点 s 连所有入点，表示可以从此开始；所有出点连汇点，表示可以从此结束。

考虑使用流量来记录旅行总次数，则限定最大流必须为 k，使用费用来记录国家数。

在网络流中一个常用的技巧是拆点。具体的，将每个国家拆成入点和出点，中间连一条容量、费用均为 1 的边。

发现这样做保证了每个国家最多经过一次。之后按照题目中的关系，若 u 可达 v，则用 u 的出边连 v 的入边。

最后源点 s 连所有入点，表示可以从此开始；所有出点连汇点，表示可以从此结束。

注意还需要一个超级源点 S，连一条 $(S, s, k, 0)$，表示最多进行 k 次旅行。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

比较好做的是同余。考虑到一个同余类里所有点都可以互达，对于每个点新建一个 mod_i，建边 $(mod_i, in_i, +\infty, 0)$。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

比较好做的是同余。考虑到一个同余类里所有点都可以互达，对于每个点新建一个 mod_i，建边 $(mod_i, in_i, +\infty, 0)$。

之后每个点的 mod_i 只需要连它之后第一个与它同余的 mod_j 即可。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

比较好做的是同余。考虑到一个同余类里所有点都可以互达，对于每个点新建一个 mod_i，建边 $(mod_i, in_i, +\infty, 0)$。

之后每个点的 mod_i 只需要连它之后第一个与它同余的 mod_j 即可。

一二条件类似，只讲第一个条件怎么做。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

比较好做的是同余。考虑到一个同余类里所有点都可以互达，对于每个点新建一个 mod_i，建边 $(mod_i, in_i, +\infty, 0)$。

之后每个点的 mod_i 只需要连它之后第一个与它同余的 mod_j 即可。

一二条件类似，只讲第一个条件怎么做。

对于某两个权值相同的点 $u < v$，若其他的点能够到 u，那么必然能到 v，所以每个点的入点可以直接连到它之后与它权值相同的最近一个的入点。

发现题目中的可达关系可能会有 $O(n^2)$ 条！考虑优化建图。

比较好做的是同余。考虑到一个同余类里所有点都可以互达，对于每个点新建一个 mod_i，建边 $(mod_i, in_i, +\infty, 0)$。

之后每个点的 mod_i 只需要连它之后第一个与它同余的 mod_j 即可。

一二条件类似，只讲第一个条件怎么做。

对于某两个权值相同的点 $u < v$，若其他的点能够到 u，那么必然能到 v，所以每个点的入点可以直接连到它之后与它权值相同的最近一个的入点。

这样处理之后，条件一就好做多了。每个点可以直接连它之后的第一个 $a_i + A$ 和 $a_i - A$（如果存在的话）。

这样优化之后边数变成了 $O(n)$。

这样优化之后边数变成了 $O(n)$。

来分析一下复杂度：$O(nmf) = O(n^2k)$。

这样优化之后边数变成了 $O(n)$。

来分析一下复杂度：$O(nmf) = O(n^2k)$。

然后这道题 n, k 开了 3000。他是能跑的！绝对~~不是出题人不会卡，嗯嗯。~~

P10541 [THUPC 2024 决赛] 研发计划

诶丝尅有 m 种产品和 n 种技术。第 i 个产品推出会获得 g_i 的收益，但有 p 条技术-产品依赖关系 (u, v)，对于每个产品，必须获得它的全部前置技术之后才能推出。

对于第 j 个技术，可以选择花费 f_j 的代价直接购买获得，或者花费 h_j 的代价通过研发获得。研发需要一定的条件：给出 q 条技术-技术依赖关系 (a, b)，必须在获得了技术 j 的所有前置技术后才能通过研发获得技术 j。

保证技术-技术依赖关系构成一个有向无环图。

最大化收益。

$n, m \leq 100, p \leq nm, q \leq \frac{n(n-1)}{2}$。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

技术-产品依赖关系是好处理的。重点讲技术-技术依赖。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

技术-产品依赖关系是好处理的。重点讲技术-技术依赖。

考虑技术 i 的两种获取方式，一种是有前置并花费 h_i，一种无前置并花费 f_i。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

技术-产品依赖关系是好处理的。重点讲技术-技术依赖。

考虑技术 i 的两种获取方式，一种是有前置并花费 h_i，一种无前置并花费 f_i。

那么我们把 s 到 i 的路分为两层，加入中间点 i'，第一层 $(s, i', h_i), (j, i', +\infty)$，第二层 (i', i, f_i)。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

技术-产品依赖关系是好处理的。重点讲技术-技术依赖。

考虑技术 i 的两种获取方式，一种是有前置并花费 h_i，一种无前置并花费 f_i。

那么我们把 s 到 i 的路分为两层，加入中间点 i'，第一层 $(s, i', h_i), (j, i', +\infty)$，第二层 (i', i, f_i)。

这两层至少要割一层，否则得割掉所有 i 的后置产品，与题意相符。

最小割。将产品看做先全选后割掉不选的，源点到不了某个技术则算有了这个技术。

技术-产品依赖关系是好处理的。重点讲技术-技术依赖。

考虑技术 i 的两种获取方式，一种是有前置并花费 h_i，一种无前置并花费 f_i。

那么我们把 s 到 i 的路分为两层，加入中间点 i'，第一层 $(s, i', h_i), (j, i', +\infty)$，第二层 (i', i, f_i)。

这两层至少要割一层，否则得割掉所有 i 的后置产品，与题意相符。

最后用 $\sum g_i$ 减去最小割即可。

上下界网络流即在普通网络上为每条边增加流量下界 b，即每条边 $(u, v) \in E$ 必须满足 $b(u, v) \leq f(u, v) \leq c(u, v)$。

无源汇上下界可行流

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f(u)=\sum b(u,v)-\sum b(v,u)$：

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。
若 $f'(u)<0$，说明出流量过大，需要 u 向 T 连 $-f'(u)$ 的边。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。
若 $f'(u)<0$，说明出流量过大，需要 u 向 T 连 $-f'(u)$ 的边。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。
若 $f'(u)<0$，说明出流量过大，需要 u 向 T 连 $-f'(u)$ 的边。

这样，求一遍这个网络的最大流。此时新网络内流量平衡。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。
若 $f'(u)<0$，说明出流量过大，需要 u 向 T 连 $-f'(u)$ 的边。

这样，求一遍这个网络的最大流。此时新网络内流量平衡。若 S 的任意一条边不满流，则说明对应的点无法流量平衡，即可行流不存在。

无源汇上下界可行流

首先让每条边先流 $b(u,v)$，在新网络中加入 $c(u,v)-b(u,v)$ 的边 (u,v)。在新网络中建立超级源点 S 和超级源点 T，想要求这个网络的最大流。但是在此之前先进行调整。考察每个普通点的净流量 $f'(u)=\sum b(u,v)-\sum b(v,u)$：

若 $f'(u)=0$，则不需要调整。
若 $f'(u)>0$，说明入流量过大，需要 S 向 u 连 $f'(u)$ 的边。
若 $f'(u)<0$，说明出流量过大，需要 u 向 T 连 $-f'(u)$ 的边。

这样，求一遍这个网络的最大流。此时新网络内流量平衡。若 S 的任意一条边不满流，则说明对应的点无法流量平衡，即可行流不存在。否则由于 S,T 的所有边都需要撤去，这样就和 b 一抵消，每个点都流量平衡了。

有源汇上下界可行流

这题比上题多了源点和汇点。

有源汇上下界可行流

这题比上题多了源点和汇点。

但实际上只需要连一条 $(t, s, +\infty)$ 就转化为无源汇了。

有源汇上下界最大最小流

在上题基础上求原网络的最大流。

有源汇上下界最大最小流

在上题基础上求原网络的最大流。

考虑在做完可行流之后的残量网络上调整，先删去附加边。

有源汇上下界最大最小流

在上题基础上求原网络的最大流。

考虑在做完可行流之后的残量网络上调整，先删去附加边。

之后在整个残量网络上做一遍 s 到 t 的最大流，加上可行流流量即为总的最大流。

有源汇上下界最大最小流

在上题基础上求原网络的最大流。

考虑在做完可行流之后的残量网络上调整，先删去附加边。

之后在整个残量网络上做一遍 s 到 t 的最大流，加上可行流流量即为总的最大流。

最小流类似。在做完可行流并删去附加边的残量网络上考虑退去不需要的流量。

有源汇上下界最大最小流

在上题基础上求原网络的最大流。

考虑在做完可行流之后的残量网络上调整，先删去附加边。

之后在整个残量网络上做一遍 s 到 t 的最大流，加上可行流流量即为总的最大流。

最小流类似。在做完可行流并删去附加边的残量网络上考虑退去不需要的流量。

那我们对着 t 到 s 做一遍最大流就可以了！可行流减最大流即为最小流。

有个问题。在最小费用流中 bellman-ford 做的时候遇到负圈了怎么办?

有个问题。在最小费用流中 bellman-ford 做的时候遇到负圈了怎么办?

注意流的定义，只包括**流量守恒**和**不超过容量限制**，所以并不需要每条路径都是从 s 到 t，是允许在圈上加一个流量的。而这也正是我们之前增广算法的缺陷。一种暴力的方法是直接在残量网络上消圈。

有个问题。在最小费用流中 bellman-ford 做的时候遇到负圈了怎么办?

注意流的定义，只包括**流量守恒**和**不超过容量限制**，所以并不需要每条路径都是从 s 到 t，是允许在圈上加一个流量的。而这也正是我们之前增广算法的缺陷。一种暴力的方法是直接在残量网络上消圈。

顺便一提，在费用流建模时，如果有负圈的出现，需要保证在圈上加流量是满足建模实际意义的。

有个问题。在最小费用流中 bellman-ford 做的时候遇到负圈了怎么办?

注意流的定义，只包括**流量守恒**和**不超过容量限制**，所以并不需要每条路径都是从 s 到 t，是允许在圈上加一个流量的。而这也正是我们之前增广算法的缺陷。一种暴力的方法是直接在残量网络上消圈。

顺便一提，在费用流建模时，如果有负圈的出现，需要保证在圈上加流量是满足建模实际意义的。

一种更好的做法是对于负边**强制满流**。即限定负边的流量下界等于流量上界，然后将其反向边的下界设为 0。这样，就将负边的流量反映到其反向边的流量上了。

有个问题。在最小费用流中 bellman-ford 做的时候遇到负圈了怎么办?

注意流的定义，只包括**流量守恒**和**不超过容量限制**，所以并不需要每条路径都是从 s 到 t，是允许在圈上加一个流量的。而这也正是我们之前增广算法的缺陷。一种暴力的方法是直接在残量网络上消圈。

顺便一提，在费用流建模时，如果有负圈的出现，需要保证在圈上加流量是满足建模实际意义的。

一种更好的做法是对于负边**强制满流**。即限定负边的流量下界等于流量上界，然后将其反向边的下界设为 0。这样，就将负边的流量反映到其反向边的流量上了。

之后跑有源汇上下界最小费用最大流，易知不会出现负圈。

模拟费用流适用于解决一些最优化问题，针对能够建出费用流建模但是时间复杂度明显不对的题。
它通常是通过分析网络的特殊结构来利用贪心等算法人工模拟网络流。
但是感觉有些时候能被反悔贪心平替掉。（这时候也可以看做用费用流来证明反悔贪心的正确性）

CF865D Buy Low Sell High

已知接下来 N 天的股票价格，每天诶丝尅可以买进一股股票，卖出一股股票，或者什么也不做。N 天之后他拥有的股票应为 0，最大化收益。

$n \leq 3 \times 10^5$。

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

费用流是好建的，$(s, u_i, 1, -c_i), (u_i, t, 1, c_i), (u_i, u_{i+1}, +\infty, 0)$，求最大费用任意流。

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

费用流是好建的，$(s, u_i, 1, -c_i), (u_i, t, 1, c_i), (u_i, u_{i+1}, +\infty, 0)$，求最大费用任意流。

接下来进行模拟。考虑每次往网络里新加一个点会发生什么。

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

费用流是好建的，$(s, u_i, 1, -c_i), (u_i, t, 1, c_i), (u_i, u_{i+1}, +\infty, 0)$，求最大费用任意流。

接下来进行模拟。考虑每次往网络里新加一个点会发生什么。

第一，考虑正费用的增广路。这对应着选之前未选的最便宜的一天买入，今天卖出；

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

费用流是好建的，$(s, u_i, 1, -c_i), (u_i, t, 1, c_i), (u_i, u_{i+1}, +\infty, 0)$，求最大费用任意流。

接下来进行模拟。考虑每次往网络里新加一个点会发生什么。

第一，考虑正费用的增广路。这对应着选之前未选的最便宜的一天买入，今天卖出；

第二，考虑新产生的正费用环。这对应着选之前的较便宜的卖出操作，撤销这次操作，改成今天卖出。

典题，想必大家都做过，但是用它来引进模拟费用流也算不错。

费用流是好建的，$(s, u_i, 1, -c_i), (u_i, t, 1, c_i), (u_i, u_{i+1}, +\infty, 0)$，求最大费用任意流。

接下来进行模拟。考虑每次往网络里新加一个点会发生什么。

第一，考虑正费用的增广路。这对应着选之前未选的最便宜的一天买入，今天卖出；

第二，考虑新产生的正费用环。这对应着选之前的较便宜的卖出操作，撤销这次操作，改成今天卖出。

用堆维护即可。

P5470 [NOI2019] 序列

给定两个长为 n 的正整数序列 a, b，诶丝尅让你在 a, b 中各选 k 个数，且必须要满足存在至少 l 个下标 x_i 使得 a_{x_i}, b_{x_i} 都被选了。最大化选出的数的和。

$n \leq 2 \times 10^5, T \leq 10, \sum n \leq 10^6$。

模拟费用流你首先得建出网络对吧。

模拟费用流你首先得建出网络对吧。

将这道题刻画成二分图匹配问题，不匹配的走公共边即可。

模拟费用流你首先得建出网络对吧。

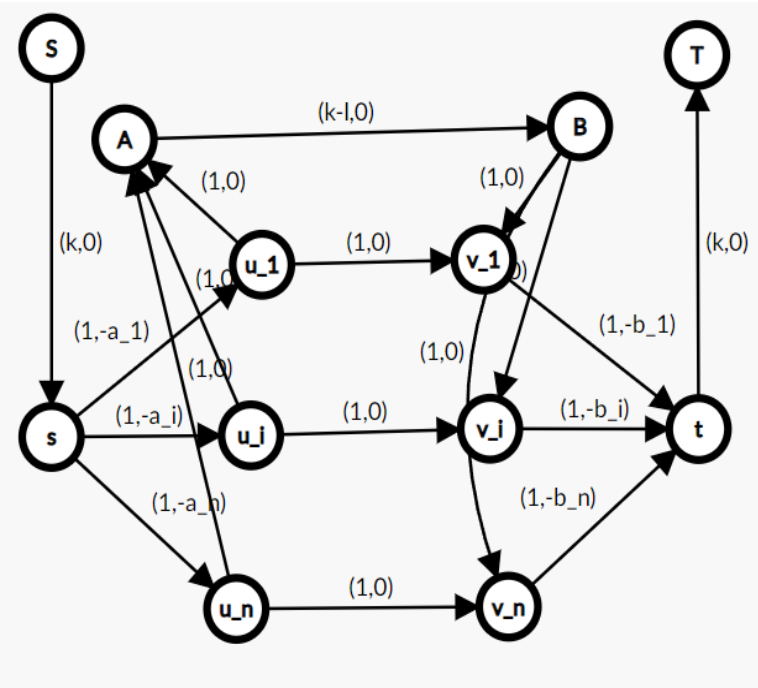
将这道题刻画成二分图匹配问题，不匹配的走公共边即可。

题目要求匹配至少 l 次，即不匹配至多 $k-l$ 次，将公共边的容量设为 $k-l$ 就做完了。

模拟费用流你首先得建出网络对吧。

将这道题刻画成二分图匹配问题，不匹配的走公共边即可。

题目要求匹配至少 l 次，即不匹配至多 $k-l$ 次，将公共边的容量设为 $k-l$ 就做完了。



但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

若 (A, B) 不满流则优先考虑走 AB。即选剩下没选的 a, b 中的最大值。(注意如果 a, b 匹配则优先走匹配边)

但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

若 (A, B) 不满流则优先考虑走 AB。即选剩下没选的 a, b 中的最大值。(注意如果 a, b 匹配则优先走匹配边)

否则，考虑走匹配边。即在剩下未匹配的中选择 $a_i + b_i$ 最大的。

但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

若 (A, B) 不满流则优先考虑走 AB。即选剩下没选的 a, b 中的最大值。(注意如果 a, b 匹配则优先走匹配边)

否则，考虑走匹配边。即在剩下未匹配的中选择 $a_i + b_i$ 最大的。

还有经过反向边的退流操作：
找到一个已选择的 a_i（设它匹配到 b_j），再另选一个未匹配的 a_k，将其改为 a_i, b_i 匹配、b_j, a_k 匹配。权值为 $b_i + a_k$ 且 AB 边流量不变。
a, b 反一下也可以。

但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

若 (A, B) 不满流则优先考虑走 AB。即选剩下没选的 a, b 中的最大值。(注意如果 a, b 匹配则优先走匹配边)

否则，考虑走匹配边。即在剩下未匹配的中选择 $a_i + b_i$ 最大的。

还有经过反向边的退流操作：
找到一个已选择的 a_i（设它匹配到 b_j），再另选一个未匹配的 a_k，将其改为 a_i, b_i 匹配、b_j, a_k 匹配。权值为 $b_i + a_k$ 且 AB 边流量不变。
a, b 反一下也可以。

我们成功地模拟出了网络上可能进行的增广。

但是三方显然过不去 10^6。考虑逐步模拟增广。大致有如下几种增广：

若 (A, B) 不满流则优先考虑走 AB。即选剩下没选的 a, b 中的最大值。(注意如果 a, b 匹配则优先走匹配边)

否则，考虑走匹配边。即在剩下未匹配的中选择 $a_i + b_i$ 最大的。

还有经过反向边的退流操作：
找到一个已选择的 a_i（设它匹配到 b_j），再另选一个未匹配的 a_k，将其改为 a_i, b_i 匹配、b_j, a_k 匹配。权值为 $b_i + a_k$ 且 AB 边流量不变。
a, b 反一下也可以。

我们成功地模拟出了网络上可能进行的增广。

这样，我们拿一车堆来维护一车最大值就可以了。

P6122 [NEERC2016] Mole Tunnels

诶丝尅给你一棵 n 个点的完全二叉树。
每个点上有 c_i 个食物最多供给 c_i 个鼹鼠吃。
一共有 m 个鼹鼠，分别住在第 p_i 个点。对于所有的 $1 \leq k \leq m$，求：
对前 k 个鼹鼠指定觅食点，每个点不能被超过 c_i 个鼹鼠指定；
求最小的前 k 个鼹鼠到其觅食点的距离和。

费用流模型：

$(s, p_i, 1, 0), (i, t, c_i, 0), (\lfloor \frac{i}{2} \rfloor, i, +\infty, 1), (i, \lfloor \frac{i}{2} \rfloor, +\infty, 1)$。

费用流模型：

$(s, p_i, 1, 0), (i, t, c_i, 0), (\lfloor \frac{i}{2} \rfloor, i, +\infty, 1), (i, \lfloor \frac{i}{2} \rfloor, +\infty, 1)$。

考虑模拟费用流，一个一个加入鼹鼠。由于 s 的出边必定满流，所以不会有新的环。只需考虑新的增广路即可。

费用流模型：
$(s, p_i, 1, 0), (i, t, c_i, 0), (\lfloor \frac{i}{2} \rfloor, i, +\infty, 1), (i, \lfloor \frac{i}{2} \rfloor, +\infty, 1)$。

考虑模拟费用流，一个一个加入鼹鼠。由于 s 的出边必定满流，所以不会有新的环。只需考虑新的增广路即可。

新的增广路即找到最近的一个关键点。

费用流模型：
$(s, p_i, 1, 0), (i, t, c_i, 0), (\lfloor \frac{i}{2} \rfloor, i, +\infty, 1), (i, \lfloor \frac{i}{2} \rfloor, +\infty, 1)$。

考虑模拟费用流，一个一个加入鼹鼠。由于 s 的出边必定满流，所以不会有新的环。只需考虑新的增广路即可。

新的增广路即找到最近的一个关键点。

并且考虑完全二叉树的直径 $O(\log n)$，所以一旦找到最近路可以暴力将边取反。

这样，问题就转化为了：

- 查询到点 u 的最近关键点。
- 将一条路上的边的权值取相反数。
- 删除一个关键点。

这样，问题就转化为了：

- 查询到点 u 的最近关键点。
- 将一条路上的边的权值取相反数。
- 删除一个关键点。

设 f_u 表示 u 子树内离 u 最近的关键点。查询只需要向上跳即可。

这样，问题就转化为了：

- 查询到点 u 的最近关键点。
- 将一条路上的边的权值取相反数。
- 删除一个关键点。

设 f_u 表示 u 子树内离 u 最近的关键点。查询只需要向上跳即可。

维护 f_u 也很简单，每次最多更新两条祖孙链，像线段树一样更新就可以了。总复杂度 $O(n \log n)$。

在解决一些问题时，我们能够轻易建出最大流模型，但是复杂度过高。
灵活利用最大流最小割定理，从最小割的角度剖析网络，也许能得到更好的做法。

神秘联考题

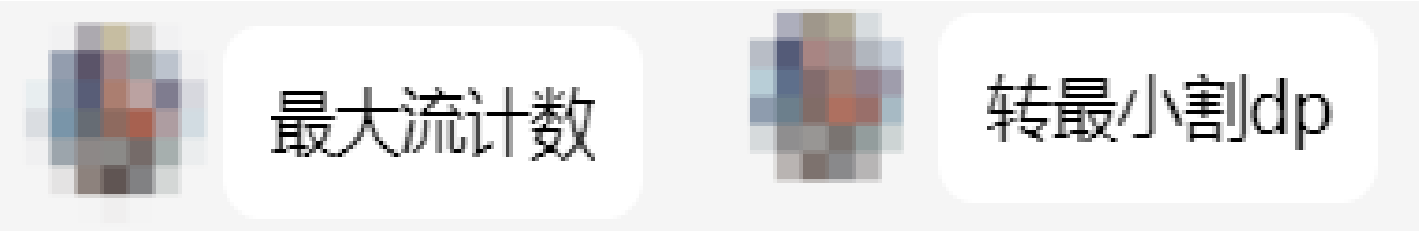
一个 $n \times m$ 的网格，每一格中填 0 或 1。

给定序列 $a_1, a_2, \cdots, a_m$，诶丝尅问有多少可能的序列 $b_1, b_2, \cdots, b_n$，使得存在一种填 01 的方式，第 i 列恰好有 a_i 个 1，第 j 行恰好有 b_j 个 1。取模。

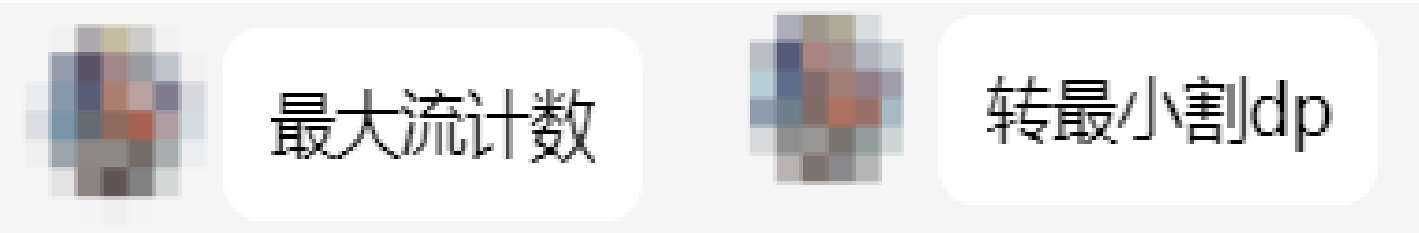
$n, m \leq 80$。

这题诶丝尅先生秒切。

这题诶丝尅先生秒切。

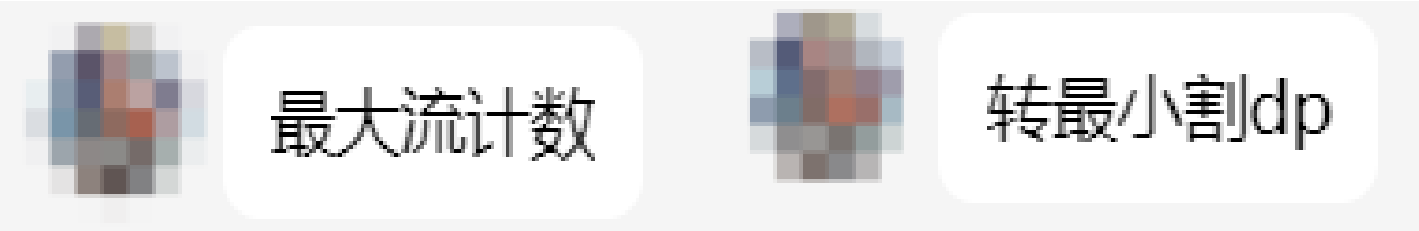


这题诶丝尅先生秒切。

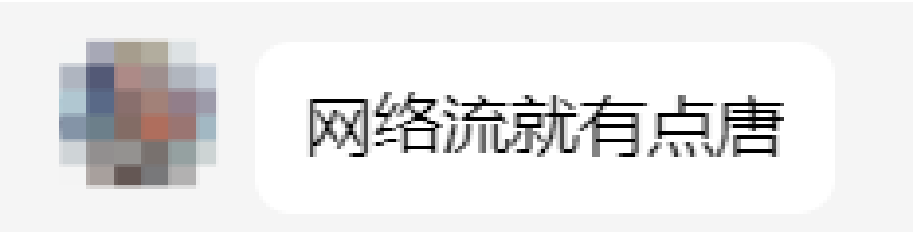


他还锐评了。

这题诶丝尅先生秒切。



他还锐评了。



这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

建出二分图模型：为每行每列建点，那么行列之间两两连边，源点连行点，汇点连列点。

这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

建出二分图模型：为每行每列建点，那么行列之间两两连边，源点连行点，汇点连列点。

那么 b_i 合法当且仅当 $\sum a_i = \sum b_i$（记为 S）且上图最大流 $= S$。

这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

建出二分图模型：为每行每列建点，那么行列之间两两连边，源点连行点，汇点连列点。

那么 b_i 合法当且仅当 $\sum a_i = \sum b_i$（记为 S）且上图最大流 $= S$。

转化为最小割是 S，即这个图的任意一个割 $\geq S$。

这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

建出二分图模型：为每行每列建点，那么行列之间两两连边，源点连行点，汇点连列点。

那么 b_i 合法当且仅当 $\sum a_i = \sum b_i$（记为 S）且上图最大流 $= S$。

转化为最小割是 S，即这个图的任意一个割 $\geq S$。

考虑左边割了最小的 x 个 a_i，右边割了最小的 y 个 b_j，那么中间就必须割 $(n-x)(m-y)$ 条边。根据割 $\geq S$，就得到了 b 序列前 y 小的数的和的下界。

这题确实如诶丝尅先生所说啊。首先考虑如何判定 b_i 合法。

建出二分图模型：为每行每列建点，那么行列之间两两连边，源点连行点，汇点连列点。

那么 b_i 合法当且仅当 $\sum a_i = \sum b_i$（记为 S）且上图最大流 $= S$。

转化为最小割是 S，即这个图的任意一个割 $\geq S$。

考虑左边割了最小的 x 个 a_i，右边割了最小的 y 个 b_j，那么中间就必须割 $(n-x)(m-y)$ 条边。根据割 $\geq S$，就得到了 b 序列前 y 小的数的和的下界。

对于所有的 x, y 计算一遍，就得到了序列 l_i 表示 b 序列前 i 小的数的和要不小于 l_i。易知这是充要条件。

之后就根据这个条件进行 dp，容易设计一个 $O(n^5)$ dp（视 n, m 同阶）：

之后就根据这个条件进行 dp，容易设计一个 $O(n^5)$ dp（视 n, m 同阶）：

设 $dp_{i,j,k}$ 表示钦定前 j 小的 b 的值域是 $[0, i]$ 且它们的总和是 k 的方案数。

之后就根据这个条件进行 dp，容易设计一个 $O(n^5)$ dp（视 n, m 同阶）：

设 $dp_{i,j,k}$ 表示钦定前 j 小的 b 的值域是 $[0, i]$ 且它们的总和是 k 的方案数。

转移的时候就是枚举新产生了 x 个 i，然后组合数转移即可。

CF724E Goods transportation

诶丝尅升任了 CF 国的大总管，他管辖的 n 个城市，编号为 $1, 2, \ldots, n$。每个城市生产了 p_i 个货物，限制最多可以卖掉 s_i 个货物。对于每两个城市 i, j，如果 $i < j$，则可以最多从 i 运送 c 个货物到 j。注意不能反向运送，却可以在多个城市之间送来送去。现在诶丝尅想知道，经过运输后，最多能卖掉多少个货物。$n \leq 10^4$。

Bonus: $n \leq 10^6$。

首先可以建出边数 $O(n^2)$ 的最大流模型：
$(s, i, p_i), (i, j, c)\{i < j\}, (i, t, s_i)$。

首先可以建出边数 $O(n^2)$ 的最大流模型：
$(s, i, p_i), (i, j, c)\{i < j\}, (i, t, s_i)$。

考虑最大流转最小割 dp。记 $dp_{i,j}$ 表示考虑到第 i 个点，$1 \sim i$ 点中有 j 个点可以从 s 到达的局部最小割。

首先可以建出边数 $O(n^2)$ 的最大流模型：
$(s, i, p_i), (i, j, c)\{i < j\}, (i, t, s_i)$。

考虑最大流转最小割 dp。记 $dp_{i,j}$ 表示考虑到第 i 个点，$1 \sim i$ 点中有 j 个点可以从 s 到达的局部最小割。

转移就是分讨 s 到 i 的可到达性，若 s 不可到达 i，则 $dp_{i,j} \leftarrow dp_{i-1,j} + p_i + j \times c$；
否则需要割掉到 t 的边，即 $dp_{i,j} \leftarrow dp_{i-1,j-1} + s_i$。

首先可以建出边数 $O(n^2)$ 的最大流模型：
$(s, i, p_i), (i, j, c)\{i < j\}, (i, t, s_i)$。

考虑最大流转最小割 dp。记 $dp_{i,j}$ 表示考虑到第 i 个点，$1 \sim i$ 点中有 j 个点可以从 s 到达的局部最小割。

转移就是分讨 s 到 i 的可到达性，若 s 不可到达 i，则 $dp_{i,j} \leftarrow dp_{i-1,j} + p_i + j \times c$；
否则需要割掉到 t 的边，即 $dp_{i,j} \leftarrow dp_{i-1,j-1} + s_i$。

这样是 $O(n^2)$ 的 dp，足以通过本题。

实际上还有更优秀的 $O(n \log n)$ 贪心做法。

实际上还有更优秀的 $O(n\log n)$ 贪心做法。

我们先假设所有 (s,i,p_i) 都割掉了，然后依次更新成割掉到 t 的边并恢复 s 边。假设第 k 次更新的是点 i，记 cnt_L, cnt_R 分别表示 i 左边右边分别更新了多少点（$cnt_L+cnt_R=k-1$），对割的大小产生的贡献是：

$$s_i-p_i-c\times cnt_L+c\times(n-i-cnt_R)=s_i-p_i+c\times(n-i)+c\times(k-1)$$

实际上还有更优秀的 $O(n\log n)$ 贪心做法。

我们先假设所有 (s,i,p_i) 都割掉了，然后依次更新成割掉到 t 的边并恢复 s 边。假设第 k 次更新的是点 i，记 cnt_L, cnt_R 分别表示 i 左边右边分别更新了多少点（$cnt_L+cnt_R=k-1$），对割的大小产生的贡献是：

$$s_i-p_i-c\times cnt_L+c\times(n-i-cnt_R)=s_i-p_i+c\times(n-i)+c\times(k-1)$$

记 $a_i=s_i-p_i+c\times(n-i)$，那么按 a_i 从小到大排序，对每个前缀计算割即可。

CF1861F Four Suits

有 n 个人，4 种不同的卡牌，初始第 i 个人有 $a_{i,j}$ 张第 j 种卡牌。诶丝尅是局外人，手里第 j 种卡牌有 b_j 个，他现在要把他的卡牌分给这 n 个人，使得分完之后每个人手里的卡牌总数相等，保证有解。

第 i 个人最终的分数是手里每种卡牌的数量的最大值。分数最高的那个人如果分数为 x，除了他之外分数最高的人的分数为 y，那么他会获得 $x-y$ 的喜悦程度（显然如果存在两个分数最高的，那么他们的喜悦度都是 0），其他人会获得 0 的喜悦程度。

对于每个人，算出他最大的可能的喜悦程度。

$n \leq 5 \times 10^4$。

先算出每个人最终的卡牌数，得出每个人需要新增 c_i 张牌。然后依次对每个人求答案。

先算出每个人最终的卡牌数，得出每个人需要新增 c_i 张牌。然后依次对每个人求答案。

假设第 x 个人最终最多的卡牌是第 j 种，则最多可能是 $a_{x,j} + \min(b_j, c_x)$，然后可以得知最大可能的分数，直接将最大值对应种卡牌给到最大肯定是优的。（如有多种可能则依次考虑）

先算出每个人最终的卡牌数，得出每个人需要新增 c_i 张牌。然后依次对每个人求答案。

假设第 x 个人最终最多的卡牌是第 j 种，则最多可能是 $a_{x,j} + \min(b_j, c_x)$，然后可以得知最大可能的分数，直接将最大值对应种卡牌给到最大肯定是优的。(如有多种可能则依次考虑)

之后考虑如何使其他人的分数最小，使用二分答案 $mid(mid \geq \max_{i \neq x}\{a_{i,j}\})$，思考如何判定。

先算出每个人最终的卡牌数，得出每个人需要新增 c_i 张牌。然后依次对每个人求答案。

假设第 x 个人最终最多的卡牌是第 j 种，则最多可能是 $a_{x,j} + \min(b_j, c_x)$，然后可以得知最大可能的分数，直接将最大值对应种卡牌给到最大肯定是优的。(如有多种可能则依次考虑)

之后考虑如何使其他人的分数最小，使用二分答案 $mid(mid \geq \max_{i \neq x}\{a_{i,j}\})$，思考如何判定。

考虑网络流建模，建点 $s, t, u_1 \sim u_4, y_1 \sim y_n$，然后建图：

先算出每个人最终的卡牌数，得出每个人需要新增 c_i 张牌。然后依次对每个人求答案。

假设第 x 个人最终最多的卡牌是第 j 种，则最多可能是 $a_{x,j} + \min(b_j, c_x)$，然后可以得知最大可能的分数，直接将最大值对应种卡牌给到最大肯定是优的。(如有多种可能则依次考虑)

之后考虑如何使其他人的分数最小，使用二分答案 $mid(mid \geq \max_{i \neq x}\{a_{i,j}\})$，思考如何判定。

考虑网络流建模，建点 $s, t, u_1 \sim u_4, y_1 \sim y_n$，然后建图：

$(s, u_j, b'_j), (u_j, v_i, mid - a_{i,j}), (v_i, t, c_i), i \neq x$，$b'_j$ 表示剩余牌数。

上图中最大流 $= \sum_{i \neq x} c_i$ 则 mid 合法。但是每次跑一遍网络流是不能接受的。

上图中最大流 $= \sum_{i \neq x} c_i$ 则 mid 合法。但是每次跑一遍网络流是不能接受的。

考虑最大流转最小割！枚举 (s, u_j, b'_j) 这类边割了哪些（2^4 个状态），然后对于每个 v_i 分讨其连通性可得局部最小割是 $\min c_i, \sum mid - a_{i,j}$。

上图中最大流 $= \sum_{i \neq x} c_i$ 则 mid 合法。但是每次跑一遍网络流是不能接受的。

考虑最大流转最小割！枚举 (s, u_j, b'_j) 这类边割了哪些（2^4 个状态），然后对于每个 v_i 分讨其连通性可得局部最小割是 $\min c_i, \sum mid - a_{i,j}$。

然后按照 $c_i + \sum a_{i,j}$ 从小到大排序，则 $\min$ 取前值的是一段前缀，预处理后二分就做完了。

上图中最大流 $= \sum_{i \neq x} c_i$ 则 mid 合法。但是每次跑一遍网络流是不能接受的。

考虑最大流转最小割！枚举 (s, u_j, b'_j) 这类边割了哪些（2^4 个状态），然后对于每个 v_i 分讨其连通性可得局部最小割是 $\min c_i, \sum mid - a_{i,j}$。

然后按照 $c_i + \sum a_{i,j}$ 从小到大排序，则 $\min$ 取前值的是一段前缀，预处理后二分就做完了。

至此我们可以 $O(\log n)$ 算最小割，总复杂度 $O(n \log n \log V)$，可以通过。

P4897 【模板】最小割树（Gomory-Hu Tree）

诶丝尅有个网络 $G(V,E)$，q 次询问两点之间的最小割。

记 $n=|V|, m=|E|$，$n \leq 500, m \leq 1500, Q \leq 10^5$。

P4897 【模板】最小割树（Gomory-Hu Tree）

诶丝尅有个网络 $G(V,E)$，q 次询问两点之间的最小割。

记 $n=|V|, m=|E|$，$n\leq 500, m\leq 1500, Q\leq 10^5$。

每次跑最小割的复杂度是 $O(Qn^2m)$，是不可接受的。

P4897 【模板】最小割树（Gomory-Hu Tree）

诶丝尅有个网络 $G(V,E)$，q 次询问两点之间的最小割。
记 $n=|V|, m=|E|$，$n\le 500, m\le 1500, Q\le 10^5$。

每次跑最小割的复杂度是 $O(Qn^2m)$，是不可接受的。

接下来介绍最小割树算法，适用于多点对间最小割相关问题。

考虑对任意点 s, t 做一次最小割之后，点集分成 S, T，$s \in S, t \in T$。那么这个最小割也是任意 $u \in S, v \in T$ 之间的一个割。

考虑对任意点 s, t 做一次最小割之后，点集分成 S, T，$s \in S, t \in T$。那么这个最小割也是任意 $u \in S, v \in T$ 之间的一个割。

接着试图将网络转化成树形结构，新建图 $T(\varnothing, E)$，在 T 上将刚刚做出来的 s, t 中间连上边权为其最小割的边。

考虑对任意点 s, t 做一次最小割之后，点集分成 S, T，$s \in S, t \in T$。那么这个最小割也是任意 $u \in S, v \in T$ 之间的一个割。

接着试图将网络转化成树形结构，新建图 $T(\varnothing, E)$，在 T 上将刚刚做出来的 s, t 中间连上边权为其最小割的边。

之后对于 S, T 分治，分别以 S 或 T 为点集做（注意最小割要在原图上跑），分治至点集只剩一个点，就形成了一棵原网络的最小割树。

最小割树的性质是对于任两点 $u, v \in E, u \neq v$，网络中 u 到 v 的最小割是树上 u 到 v 路径上每条边边权的最小值。感性理解一下，每个边对应一种割的方式，且每个割都能够将 u, v 分开。

最小割树的性质是对于任两点 $u, v \in E, u \neq v$，网络中 u 到 v 的最小割是树上 u 到 v 路径上每条边边权的最小值。感性理解一下，每个边对应一种割的方式，且每个割都能够将 u, v 分开。

预处理是 $O(n^3 m)$ 的，但是跑不满。处理询问可以用倍增做到 $O(\log n)$，没必要。

CF1682F MCMF?

诶丝尅给你两个整数序列 a 和 $b(b_i \neq 0)$。数组 a 保证非降。

q 次查询 l, r，求 $a[l:r]$ 的贡献。定义如下：

- 建一个有 $r-l+1$ 个顶点的二分图，从 l 到 r 编号，$b_i < 0$ 的顶点在左边，$b_i > 0$ 的顶点在右边。对于每个 $i, j \in [l, r]$，若 $l \leq i, j \leq r$，$b_i < 0$ 且 $b_j > 0$，则建一条从 i 到 j 的边，容量无限，费用为 $|a_i - a_j|$。
- 再添加源 S 和汇 T，对于每个 $i \in [l, r]$，若 $b_i < 0$，则建边 $(S, i, -b_i, 0)$。否则建边 $(i, T, b_i, 0)$。
- $a[l:r]$ 的贡献就是从 S 到 T 的最大流的最小费用。

对于所有询问，保证 $sum_{j=l}^{r} b_j = 0$。

$n, q \leq 2 \times 10^5$。

容易发现题目的 $O(n^2)$ 建图十分愚蠢。由于 a 不降，则显然可以转化成相邻两点连双向边，费用为 $a_{i+1} - a_i$。

容易发现题目的 $O(n^2)$ 建图十分愚蠢。由于 a 不降，则显然可以转化成相邻两点连双向边，费用为 $a_{i+1} - a_i$。

由于 $\sum b_i = 0$，说明网络必定满流。

容易发现题目的 $O(n^2)$ 建图十分愚蠢。由于 a 不降，则显然可以转化成相邻两点连双向边，费用为 $a_{i+1} - a_i$。

由于 $\sum b_i = 0$，说明网络必定满流。

此时考察每对双向边的总流量。对于每条双向边，其左右的 b 显然是各自先抵消完，再左右抵消的，即贡献为 $|\sum_{j=l}^{i} b_j|$。

容易发现题目的 $O(n^2)$ 建图十分愚蠢。由于 a 不降，则显然可以转化成相邻两点连双向边，费用为 $a_{i+1}-a_i$。

由于 $\sum b_i=0$，说明网络必定满流。

此时考察每对双向边的总流量。对于每条双向边，其左右的 b 显然是各自先抵消完，再左右抵消的，即贡献为 $|\sum_{j=l}^{i} b_j|$。

$$\text{ans}=\sum_{i=l}^{r-1}(a_{i+1}-a_i)\left|\sum_{j=l}^{i} b_j\right|=\sum_{i=l}^{r-1}(a_{i+1}-a_i)|s_i-s_{l-1}|,\ s_i=\sum_{j=1}^{i} b_j$$

容易发现题目的 $O(n^2)$ 建图十分愚蠢。由于 a 不降，则显然可以转化成相邻两点连双向边，费用为 $a_{i+1}-a_i$。

由于 $\sum b_i=0$，说明网络必定满流。

此时考察每对双向边的总流量。对于每条双向边，其左右的 b 显然是各自先抵消完，再左右抵消的，即贡献为 $|\sum_{j=l}^{i} b_j|$。

$$\text{ans}=\sum_{i=l}^{r-1}(a_{i+1}-a_i)\left|\sum_{j=l}^{i} b_j\right|=\sum_{i=l}^{r-1}(a_{i+1}-a_i)|s_i-s_{l-1}|,\ s_i=\sum_{j=1}^{i} b_j$$

然后对着 s_{l-1} 从小到大扫描线，分讨绝对值内部的正负就做完了。$O(n\log n)$。

CF1054F Electric Scheme

平面上诶丝尅画了若干条线段，有红蓝两种颜色。红色线段与 X 轴平行，蓝色线段与 Y 轴平行，同色线段均不相交。

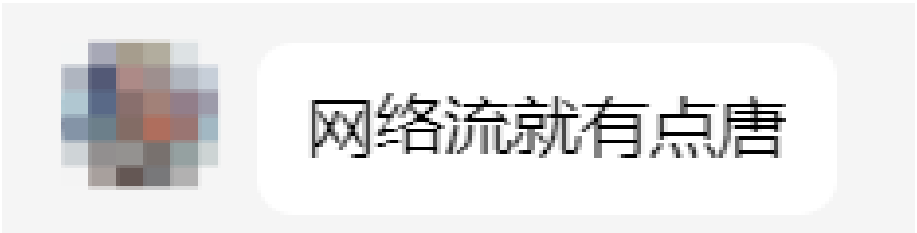
已知这些线段之间形成了 n 个交点 (在端点相交也算相交)，给出这些交点，求平面上至少有多少条线段，并输出任意一种方案。线段长度允许为 0。

$n \leq 1000$。

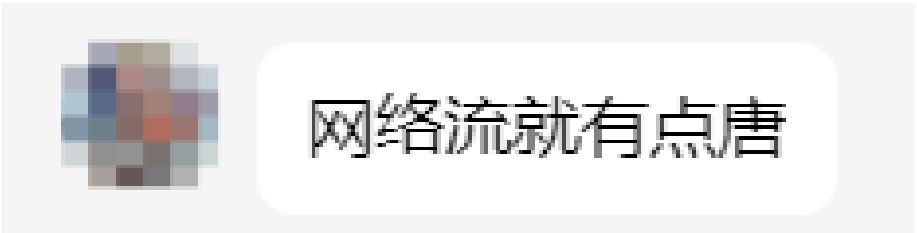
搞笑 *2700。

搞笑 *2700。著名评论家诶丝尅先生曾这样评价这道题：

搞笑 *2700。著名评论家诶丝尅先生曾这样评价这道题：

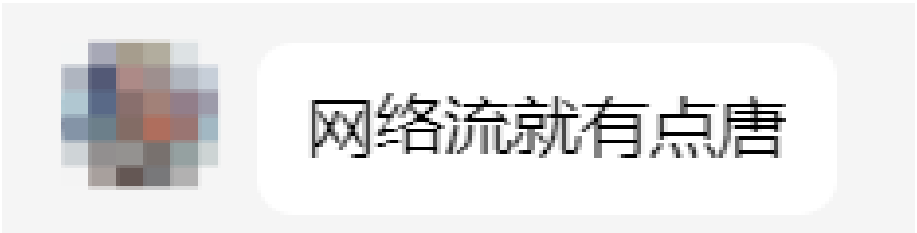


搞笑 *2700。著名评论家诶丝尅先生曾这样评价这道题：



考虑每行、每列上的相邻点对之间的线段组成的集合，那么总线段数最少即等价于在此集合里选取最多使得不存在相交。

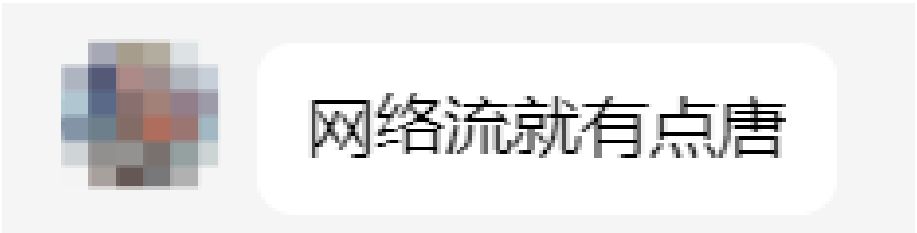
搞笑 *2700。著名评论家诶丝尅先生曾这样评价这道题：



考虑每行、每列上的相邻点对之间的线段组成的集合，那么总线段数最少即等价于在此集合里选取最多使得不存在相交。

建图，将相交的线段对连边，形成二分图，即求该二分图的最大独立集。

搞笑 *2700。著名评论家诶丝尅先生曾这样评价这道题：



考虑每行、每列上的相邻点对之间的线段组成的集合，那么总线段数最少即等价于在此集合里选取最多使得不存在相交。

建图，将相交的线段对连边，形成二分图，即求该二分图的最大独立集。

拿个网络流跑个最大匹配，之后 $O(V \times E)$ 转化为最大独立集的方案就可以了。

P2053 [SCOI2007] 修车

同一时刻 n 个诶丝尅带着他的车来到汽车维修中心。
有 m 个维修人员，第 i 个维修人员维修第 j 辆车耗费 $T_{i,j}$ 的时间。
最小化诶丝尅平均等待时间。等待时间计算至维修完毕的时刻。
$2 \leq m \leq 9, 1 \leq n \leq 60$。

等待时间的计算：对于维修人员独立，设维修人员 x 依次修了 $b_1, b_2, \cdots, b_k$，则 time $= \sum_i \sum_{j \le i} T_{x,b_j} = \sum (k-i+1) T_{x,b_i}$。

等待时间的计算：对于维修人员独立，设维修人员 x 依次修了 $b_1, b_2, \cdots, b_k$，则 time $= \sum_i \sum_{j \le i} T_{x,b_j} = \sum (k - i + 1) T_{x,b_i}$。

将 b 序列翻转，则可以写成 $\sum i T_{x,b_i}$。

等待时间的计算：对于维修人员独立，设维修人员 x 依次修了 $b_1, b_2, \cdots, b_k$，则 time $= \sum_i \sum_{j \le i} T_{x,b_j} = \sum (k-i+1) T_{x,b_i}$。

将 b 序列翻转，则可以写成 $\sum i T_{x,b_i}$。

这样，可以将维修不同车辆的同一维修员拆开来看，对一个维修员建 n 个点，分别表示维修第 i 辆车的维修员。

等待时间的计算：对于维修人员独立，设维修人员 x 依次修了 $b_1, b_2, \cdots, b_k$，则 time $= \sum_i \sum_{j \le i} T_{x,b_j} = \sum (k-i+1) T_{x,b_i}$。

将 b 序列翻转，则可以写成 $\sum i T_{x,b_i}$。

这样，可以将维修不同车辆的同一维修员拆开来看，对一个维修员建 n 个点，分别表示维修第 i 辆车的维修员。

之后二分图建图，第 i 辆车的维修员的费用乘 i，跑最小费用流就做完了。

CF103E Buying Sets

有一个大小为 n 的全集，每个元素是一个数，有 n 个子集。题目保证任意 k 个子集的并的大小 $\geq k$。
每个子集有一个可正可负的权值，你需要选出一些子集使得这些子集并的大小等于子集个数，且所选子集的权值和最小。可以为空集。
$n \leq 300$。

首先二分图建模，中间的边权值设 $+\infty$，然后考虑这个图的任意一个割。

首先二分图建模，中间的边权值设 $+\infty$，然后考虑这个图的任意一个割。

假设左边割了 k 个，那么右边就需要割 $\geq n-k$ 个，即总和 $\geq n$。

首先二分图建模，中间的边权值设 $+\infty$，然后考虑这个图的任意一个割。

假设左边割了 k 个，那么右边就需要割 $\geq n-k$ 个，即总和 $\geq n$。

然而题目要求右端割恰好 $n-k$ 个，即总共恰好割 n 个。所以实际上要求的是割的边数最小化。

首先二分图建模，中间的边权值设 $+\infty$，然后考虑这个图的任意一个割。

假设左边割了 k 个，那么右边就需要割 $\geq n-k$ 个，即总和 $\geq n$。

然而题目要求右端割恰好 $n-k$ 个，即总共恰好割 n 个。所以实际上要求的是割的边数最小化。

那么我们对于左右端边权值均加上一个大数 N，这样若一旦割 $> n$ 个边，割的权值就会加上若干个 N，且显然存在仅割 n 条边的方式，前者就不可能成为最小割。

首先二分图建模，中间的边权值设 $+\infty$，然后考虑这个图的任意一个割。

假设左边割了 k 个，那么右边就需要割 $\geq n-k$ 个，即总和 $\geq n$。

然而题目要求右端割恰好 $n-k$ 个，即总共恰好割 n 个。所以实际上要求的是割的边数最小化。

那么我们对于左右端边权值均加上一个大数 N，这样若一旦割 $> n$ 个边，割的权值就会加上若干个 N，且显然存在仅割 n 条边的方式，前者就不可能成为最小割。

这样，这个图的最小割割出来的即为满足题意的且权值最小的方案。

CF1416F Showing Off

有一个 $n \times m$ 的矩阵，其中每个格子 (i, j) 中包含两个值：

- 这个格子的权值 $a_{i,j}$，其中 $a_{i,j}$ 是一个**正整数**。
- 这个格子的方向 $d_{i,j} \in \{\mathtt{L}, \mathtt{R}, \mathtt{D}, \mathtt{U}\}$，分别表示 (i, j) 指向的格子为 $(i, j-1), (i, j+1), (i+1, j), (i-1, j)$，保证指向的格子仍在矩阵中。

对于每个格子，我们连接一条从它到它指向的格子的有向边，得到一张有向图 G。用 $a \rightsquigarrow b$ 表示 a 可以通过 G 中的边到达 b，特别地，有 $a \rightsquigarrow a$。

定义 $S_{i,j} = \sum_{(i,j) \rightsquigarrow (k,l)} a_{k,l}$，现在给出所有的 $S_{i,j}$，你需要还原出一组可行的 $a_{i,j}$ 及 $d_{i,j}$，或报告无解。

$1 \le nm \le 10^5$。

每个连通块的点数和边数相同，即图 G 是内向基环树森林。

每个连通块的点数和边数相同，即图 G 是内向基环树森林。

考察每棵基环树的圈，圈上的 S 相同且为圈上的 a 之和。

每个连通块的点数和边数相同，即图 G 是内向基环树森林。

考察每棵基环树的圈，圈上的 S 相同且为圈上的 a 之和。

对于圈外的树上的点，有 $S_u = S_v + a_u$，即 $S_u > S_v$。如果某个格子周围有 S 比它小的点，那么这个格子就可以直接指向比它小的格子。我们将这种格子染成黑色。

每个连通块的点数和边数相同，即图 G 是内向基环树森林。

考察每棵基环树的圈，圈上的 S 相同且为圈上的 a 之和。

对于圈外的树上的点，有 $S_u = S_v + a_u$，即 $S_u > S_v$。如果某个格子周围有 S 比它小的点，那么这个格子就可以直接指向比它小的格子。我们将这种格子染成黑色。

接下来只需考虑白格，而白格就不可能成为圈外树上的点，只能考虑匹配圈。由于网格图是二分图，则形成的圈只能是偶圈。由于可以构造任意一个方案，则可以将大圈分解成若干个二元圈。因此接下来只考虑二元圈。

二元圈，即为相邻两个 S 相同的格子匹配，转化为匹配问题。

二元圈，即为相邻两个 S 相同的格子匹配，转化为匹配问题。

对网格棋盘染色（与上面的黑白独立），二分图建模，相邻 S 相同的连边。

二元圈，即为相邻两个 S 相同的格子匹配，转化为匹配问题。

对网格棋盘染色（与上面的黑白独立），二分图建模，相邻 S 相同的连边。

然后对于白色格子，令其与源点（或汇点）的边的流量下界为 1。跑有源汇上下界可行流即可。

二元圈，即为相邻两个 S 相同的格子匹配，转化为匹配问题。

对网格棋盘染色（与上面的黑白独立），二分图建模，相邻 S 相同的连边。

然后对于白色格子，令其与源点（或汇点）的边的流量下界为 1。跑有源汇上下界可行流即可。

时间复杂度 $O((nm)^{1.5})$。

qoj7737 Extending Distance

有一个 n 行 m 列的网格图，相邻格点之间有边，边有边权。
你可以进行任意次操作，每次操作为使某条边的边权增加 1。
求一种操作次数最小的方案使得从第一列**任意**一个点出发到最后一列**任意**一个点的**最短路的最小值**恰好增加了 K，输出方案。
$1 \le n \times m \le 500, 1 \le k \le 100, 1 \le$ 边权 $\le 10^9$。

考虑网格图对偶。建立网络，源点连第一行的点，汇点连第 n 行的点，则原图的左端到右端任意一条路径恰对应了这个网络上的一个割。原图最短路的最小值恰对应这个网络的最小割。

考虑网格图对偶。建立网络，源点连第一行的点，汇点连第 n 行的点，则原图的左端到右端任意一条路径恰对应了这个网络上的一个割。原图最短路的最小值恰对应这个网络的最小割。

将原图中的边费用设为 0，容量设为边权；再对每条边另建费用为 1 容量为 $+\infty$ 的边，则转为找网络上流量为 $D+K$ 的最小费用流，其中 D 是最短路。

但是 D 可能很大，$O(nmf)$ 跑不动。由于原图的边费用均为 0，所以可以先把 0 费的边拎出来先跑最大流，再用费用流对整个网络继续增广。

但是 D 可能很大，$O(nmf)$ 跑不动。由于原图的边费用均为 0，所以可以先把 0 费的边拎出来先跑最大流，再用费用流对整个网络继续增广。

输出方案是容易的，考察所有 1 费的边即可。时间复杂度 $O(n^2m^2k)$。