网络流入门

daklqw

Zhenhai High School

August 7, 2021

- 由先前的经验，没人问我问题！所以这里不 QA 了。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。
- 如果只是针对做题，那么学习网络流的一个方法就是了解网络流的性质，以及熟悉多种建图技巧的原理。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。
- 如果只是针对做题，那么学习网络流的一个方法就是了解网络流的性质，以及熟悉多种建图技巧的原理。
- 所以我不保证我教的能做出神仙网络流题，因为我自己都可能做不出。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。
- 如果只是针对做题，那么学习网络流的一个方法就是了解网络流的性质，以及熟悉多种建图技巧的原理。
- 所以我不保证我教的能做出神仙网络流题，因为我自己都可能做不出。
- 还有，证明的内容会比较少，因为我较多成分是感性理解的，所以甚至可能会证错。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。
- 如果只是针对做题，那么学习网络流的一个方法就是了解网络流的性质，以及熟悉多种建图技巧的原理。
- 所以我不保证我教的能做出神仙网络流题，因为我自己都可能做不出。
- 还有，证明的内容会比较少，因为我较多成分是感性理解的，所以甚至可能会证错。
- 详尽地学习算法请看算导，这里主要讲模型。

- 由先前的经验，没人问我问题！所以这里不 QA 了。
- 网络流的题目一般是利用网络流的性质建图解决问题，一般建完图之后使用一些经典的网络流算法或者用数据结构模拟来得出解。
- 如果只是针对做题，那么学习网络流的一个方法就是了解网络流的性质，以及熟悉多种建图技巧的原理。
- 所以我不保证我教的能做出神仙网络流题，因为我自己都可能做不出。
- 还有，证明的内容会比较少，因为我较多成分是感性理解的，所以甚至可能会证错。
- 详尽地学习算法请看算导，这里主要讲模型。
- OI 中大部分都写基于增广路的，所以基于 push-relabel 的我也没学。

图的定义不多讲，我们来定义有源汇点的网络流。

图的定义不多讲，我们来定义有源汇点的网络流。给定一张有向简单图 $G=(V,E)$，每条边 $e=(u,v)$ 有容量 c_e，有流量 w_e，当 $e\notin E$ 不存在则记 $c_e=w_e=0$。给定源点 S 和汇点 T，如果：

- 对于任意 e 都有 $0\le w_e\le c_e$。
- 对于 $u\notin\{S,T\}$，都有 $\sum_{v\in V}w_{(v,u)}=\sum_{v\in V}w_{(u,v)}$。
- $\sum_{v\in V}w_{(v,S)}=\sum_{v\in V}w_{(T,v)}=0$。（不重要）

图的定义不多讲，我们来定义有源汇点的网络流。给定一张有向简单图 $G=(V,E)$，每条边 $e=(u,v)$ 有容量 c_e，有流量 w_e，当 $e\notin E$ 不存在则记 $c_e=w_e=0$。给定源点 S 和汇点 T，如果：

- 对于任意 e 都有 $0\le w_e\le c_e$。
- 对于 $u\notin\{S,T\}$，都有 $\sum_{v\in V}w_{(v,u)}=\sum_{v\in V}w_{(u,v)}$。
- $\sum_{v\in V}w_{(v,S)}=\sum_{v\in V}w_{(T,v)}=0$。（不重要）

第一条是容量限制，第二条是流量守恒。可以将边看做水管，源点提供了源源不断的水，汇点则不停接收水，而可行的网络流就是某一时刻水管的流量。

最大流就是可行网络流中从源点流出流量取到最大值。很明显一张图可能会有多种最大流的方案。

图的定义不多讲，我们来定义有源汇点的网络流。给定一张有向简单图 $G=(V,E)$，每条边 $e=(u,v)$ 有容量 c_e，有流量 w_e，当 $e\notin E$ 不存在则记 $c_e=w_e=0$。给定源点 S 和汇点 T，如果：

- 对于任意 e 都有 $0\le w_e\le c_e$。
- 对于 $u\notin\{S,T\}$，都有 $\sum_{v\in V}w_{(v,u)}=\sum_{v\in V}w_{(u,v)}$。
- $\sum_{v\in V}w_{(v,S)}=\sum_{v\in V}w_{(T,v)}=0$。（不重要）

第一条是容量限制，第二条是流量守恒。可以将边看做水管，源点提供了源源不断的水，汇点则不停接收水，而可行的网络流就是某一时刻水管的流量。

最大流就是可行网络流中从源点流出流量取到最大值。很明显一张图可能会有多种最大流的方案。

至于费用流就是给每条边 e 加上费用 w_e，因此会有求费用取到最值时的最大流或可行流的问题。

很多人求最大流的时候使用的是增广路算法。我们可以将容量为 x 的边看做 x 条重边，那么可行流就是源点到汇点若干条边不相交的路径。

于是我们可以使用不断找这样的路径的方法求最大流，但是会有反例（二分图随手构造）。

很多人求最大流的时候使用的是增广路算法。我们可以将容量为 x 的边看做 x 条重边，那么可行流就是源点到汇点若干条边不相交的路径。

于是我们可以使用不断找这样的路径的方法求最大流，但是会有反例（二分图随手构造）。

此时要加入撤销边的概念，当一条边被经过，我们在图中删去它，并加入它的反向边，作为撤销边。这样得到的网络被称作残余网络。

当这条撤销边被经过的时候，我们有两条路径：
$a \to u \to v \to b, c \to v \to u \to d$，那么 $a \to d, c \to b$ 也是行的。

很多人求最大流的时候使用的是增广路算法。我们可以将容量为 x 的边看做 x 条重边，那么可行流就是源点到汇点若干条边不相交的路径。

于是我们可以使用不断找这样的路径的方法求最大流，但是会有反例（二分图随手构造）。

此时要加入撤销边的概念，当一条边被经过，我们在图中删去它，并加入它的反向边，作为撤销边。这样得到的网络被称作残余网络。

当这条撤销边被经过的时候，我们有两条路径：

$a \to u \to v \to b, c \to v \to u \to d$，那么 $a \to d, c \to b$ 也是行的。

于是我们可以在残余网络上不断找 S 到 T 的路径，只要找到就能使流增加 1，同时保证是一个可行的流。这样的路径就叫增广路。

当找不到增广路的时候，就得到了最大流。我们使用反证法证明这个。

首先显然当存在 $x(x>0)$ 的可行流的时候，也存在 $x-1$ 的可行流，因为删去一条路径就好。

当找不到增广路的时候，就得到了最大流。我们使用反证法证明这个。

首先显然当存在 $x(x>0)$ 的可行流的时候，也存在 $x-1$ 的可行流，因为删去一条路径就好。

当我们得到 x 的流时没有增广路，如果最大流 $>x$，记为 x 时的方案为 A，$x+1$ 时为 B，那么在 A 不在 B 中的边记为反向边，在 B 不在 A 中的边记为正向边，拿出这些边，分析度数发现仍然满足流量守恒，取出与 S 弱连通的边，就是一条增广路。矛盾。

当找不到增广路的时候，就得到了最大流。我们使用反证法证明这个。

首先显然当存在 $x(x>0)$ 的可行流的时候，也存在 $x-1$ 的可行流，因为删去一条路径就好。

当我们得到 x 的流时没有增广路，如果最大流 $>x$，记为 x 时的方案为 A，$x+1$ 时为 B，那么在 A 不在 B 中的边记为反向边，在 B 不在 A 中的边记为正向边，拿出这些边，分析度数发现仍然满足流量守恒，取出与 S 弱连通的边，就是一条增广路。矛盾。

直接找增广路是和权值有关的，但是如果我们的方案中可能会存在若干条完全重合的增广路，我们取路径上 v 的 min 来增广 min 次，可以加快过程。

如果每次通过 bfs 找到 S 到 T 的最短路增广，那么这样的复杂度是 $O(VE^2)$ 的，也就是最多增广 $O(VE)$ 次。

如果每次通过 bfs 找到 S 到 T 的最短路增广，那么这样的复杂度是 $O(VE^2)$ 的，也就是最多增广 $O(VE)$ 次。

首先证明增广后每个点的最短路是单调不减的。反证法，取减少了的，最短路最小的点 u，它减小的原因只能是新加的边，那么它上一个点 v 是它在增广路中的后继。而 v 一定是没减小的，那么在增广前，增广路中 v 的距离比 u 距离大显然矛盾了。

如果每次通过 bfs 找到 S 到 T 的最短路增广，那么这样的复杂度是 $O(VE^2)$ 的，也就是最多增广 $O(VE)$ 次。

首先证明增广后每个点的最短路是单调不减的。反证法，取减少了的，最短路最小的点 u，它减小的原因只能是新加的边，那么它上一个点 v 是它在增广路中的后继。而 v 一定是没减小的，那么在增广前，增广路中 v 的距离比 u 距离大显然矛盾了。

当一条边 (u, v) 被使用过后，它的反向边若要再被使用，那么此时 u 的距离会至少 $+2$。

最短路长度小于等于 V，于是得到上界 $O(VE)$。这个算法就是 Edmonds-Karp 算法。

我们 bfs 完会得到一棵最短路网，如果沿着这个最短路网 dfs 增广完所有路径，那么复杂度是 $O(V^2E)$ 的，这个算法是 dinic 算法，当然 dfs 的时候要注意实现技巧（比如当前弧优化）。

我们 bfs 完会得到一棵最短路网，如果沿着这个最短路网 dfs 增广完所有路径，那么复杂度是 $O(V^2E)$ 的，这个算法是 dinic 算法，当然 dfs 的时候要注意实现技巧（比如当前弧优化）。

首先关于增广，bfs 完每条边最多被废弃一次，那么这样每次找增广路都是 $O(V)$ 的，也就是这部分复杂度是 $O(V^2E)$ 的。

我们 bfs 完会得到一棵最短路网，如果沿着这个最短路网 dfs 增广完所有路径，那么复杂度是 $O(V^2E)$ 的，这个算法是 dinic 算法，当然 dfs 的时候要注意实现技巧（比如当前弧优化）。

首先关于增广，bfs 完每条边最多被废弃一次，那么这样每次找增广路都是 $O(V)$ 的，也就是这部分复杂度是 $O(V^2E)$ 的。

关于 bfs 的次数，因为这个最短路网只能是分层图，所以每条增广路每层只能选刚好一个点。于是每次增广完所有的，源点到汇点的最短路一定会增加。所以最多 bfs $O(V)$ 次。

我们 bfs 完会得到一棵最短路网，如果沿着这个最短路网 dfs 增广完所有路径，那么复杂度是 $O(V^2E)$ 的，这个算法是 dinic 算法，当然 dfs 的时候要注意实现技巧（比如当前弧优化）。

首先关于增广，bfs 完每条边最多被废弃一次，那么这样每次找增广路都是 $O(V)$ 的，也就是这部分复杂度是 $O(V^2E)$ 的。

关于 bfs 的次数，因为这个最短路网只能是分层图，所以每条增广路每层只能选刚好一个点。于是每次增广完所有的，源点到汇点的最短路一定会增加。所以最多 bfs $O(V)$ 次。

两个部分结合，就证明了上界。

至于费用流，如果考虑最小费用流，那么就用费用网络的最短路来代替 bfs。需要注意的是反向边的费用是正向边的相反数。

至于费用流，如果考虑最小费用流，那么就用费用网络的最短路来代替 bfs。需要注意的是反向边的费用是正向边的相反数。

直接改 EK 算法的话得到的复杂度就是 $O(nmf)$，其中 f 是最大流。由于存在负权，且图在不断变化，所以只能使用 bellman-ford 算法来求最短路。

至于费用流，如果考虑最小费用流，那么就用费用网络的最短路来代替 bfs。需要注意的是反向边的费用是正向边的相反数。

直接改 EK 算法的话得到的复杂度就是 $O(nmf)$，其中 f 是最大流。由于存在负权，且图在不断变化，所以只能使用 bellman-ford 算法来求最短路。

复杂度和最大流有关，所以一般费用流题目的复杂度就比较玄学！虽然是存在关于 f 更优的算法的。

至于费用流，如果考虑最小费用流，那么就用费用网络的最短路来代替 bfs。需要注意的是反向边的费用是正向边的相反数。

直接改 EK 算法的话得到的复杂度就是 $O(nmf)$，其中 f 是最大流。由于存在负权，且图在不断变化，所以只能使用 bellman-ford 算法来求最短路。

复杂度和最大流有关，所以一般费用流题目的复杂度就比较玄学！虽然是存在关于 f 更优的算法的。

通过类似的方法，基于三角形不等式能证明每次得到的最短路是单调不降的。

至于费用流，如果考虑最小费用流，那么就用费用网络的最短路来代替 bfs。需要注意的是反向边的费用是正向边的相反数。

直接改 EK 算法的话得到的复杂度就是 $O(nmf)$，其中 f 是最大流。由于存在负权，且图在不断变化，所以只能使用 bellman-ford 算法来求最短路。

复杂度和最大流有关，所以一般费用流题目的复杂度就比较玄学！虽然是存在关于 f 更优的算法的。

通过类似的方法，基于三角形不等式能证明每次得到的最短路是单调不降的。

也就是说记 $f(x)$ 为流量为 x 时的最小费用，那么 $f(x)$ 在定义域里是凸的。这点在证明某些凸优化题目的时候尤其有用，直接费用流建图得到权值的凸性。

割是将点集分割成两个集合 (S, T)，其中源点在 S，汇点在 T，则割删除的边就是 $u \in S, v \in T$ 的边 (u, v)。割的容量就是这些边容量的和。最小割就是容量最小的割。显然一个割会使得源汇点不连通。

割是将点集分割成两个集合 (S, T)，其中源点在 S，汇点在 T，则割删除的边就是 $u \in S, v \in T$ 的边 (u, v)。割的容量就是这些边容量的和。最小割就是容量最小的割。显然一个割会使得源汇点不连通。

删去这些边会导致 S 和 T 不连通，于是得到了最大流小于等于最小割。

割是将点集分割成两个集合 (S, T)，其中源点在 S，汇点在 T，则割删除的边就是 $u \in S, v \in T$ 的边 (u, v)。割的容量就是这些边容量的和。最小割就是容量最小的割。显然一个割会使得源汇点不连通。

删去这些边会导致 S 和 T 不连通，于是得到了最大流小于等于最小割。

考虑得到最大流的时候，我们从残余网络中得到割 (S, T)，使得源点能到达所有 S 中的点，T 中所有的点能到达汇点。很明显这样的集合对是唯一的。

割是将点集分割成两个集合 (S, T)，其中源点在 S，汇点在 T，则割删除的边就是 $u \in S, v \in T$ 的边 (u, v)。割的容量就是这些边容量的和。最小割就是容量最小的割。显然一个割会使得源汇点不连通。

删去这些边会导致 S 和 T 不连通，于是得到了最大流小于等于最小割。

考虑得到最大流的时候，我们从残余网络中得到割 (S, T)，使得源点能到达所有 S 中的点，T 中所有的点能到达汇点。很明显这样的集合对是唯一的。

考虑 $u \in S, v \in T, (u, v) \in E$，$(u, v)$ 一定满流，否则残余网络中存在 (u, v)，v 能被源点到达。

割是将点集分割成两个集合 (S, T)，其中源点在 S，汇点在 T，则割删除的边就是 $u \in S, v \in T$ 的边 (u, v)。割的容量就是这些边容量的和。最小割就是容量最小的割。显然一个割会使得源汇点不连通。

删去这些边会导致 S 和 T 不连通，于是得到了最大流小于等于最小割。

考虑得到最大流的时候，我们从残余网络中得到割 (S, T)，使得源点能到达所有 S 中的点，T 中所有的点能到达汇点。很明显这样的集合对是唯一的。

考虑 $u \in S, v \in T, (u, v) \in E$，$(u, v)$ 一定满流，否则残余网络中存在 (u, v)，v 能被源点到达。

考虑 $u \in S, v \in T, (v, u) \in E$，$(u, v)$ 一定没有流，否则残余网络中存在 (u, v)，v 能被源点到达。

一个容易理解的引理就是对于割 (S,T)，有 $f=\sum_{u\in S}\sum_{v\in T}\left(w_{(u,v)}-w_{(v,u)}\right)$。

感性地理解就是把 S 看做一个点 u，T 看做一个点 v，然后把边压成 (u,v) 和 (v,u)，那么这个等式很显然了。

一个容易理解的引理就是对于割 (S, T)，有 $f = \sum_{u \in S} \sum_{v \in T} \left(w_{(u,v)} - w_{(v,u)} \right)$。

感性地理解就是把 S 看做一个点 u，T 看做一个点 v，然后把边压成 (u, v) 和 (v, u)，那么这个等式很显然了。

那么我们的最大流刚好对应着这个割割掉的所有边，所以最大流大于等于最小割。

因此最小割等于最大流。鹌鹑吃鸭子（quail eat duck）。

一个容易理解的引理就是对于割 (S,T)，有 $f=\sum_{u\in S}\sum_{v\in T}\left(w_{(u,v)}-w_{(v,u)}\right)$。

感性地理解就是把 S 看做一个点 u，T 看做一个点 v，然后把边压成 (u,v) 和 (v,u)，那么这个等式很显然了。

那么我们的最大流刚好对应着这个割割掉的所有边，所以最大流大于等于最小割。

因此最小割等于最大流。鹌鹑吃鸭子（quail eat duck）。

最小割看上去就很割！它使源汇点不连通！利用这个定理我们可以建很多图了！!1

你也许听说过 dinic 跑二分图匹配！我们在这里证明它的复杂度！

你也许听说过 dinic 跑二分图匹配！我们在这里证明它的复杂度！

因为 dinic 每次匹配最短路长度会至少增加 1，那么做完 $O(\sqrt{V})$ 轮后，最短路长度也至少会达到 $O(\sqrt{V})$。

你也许听说过 dinic 跑二分图匹配！我们在这里证明它的复杂度！

因为 dinic 每次匹配最短路长度会至少增加 1，那么做完 $O(\sqrt{V})$ 轮后，最短路长度也至少会达到 $O(\sqrt{V})$。

考虑此时匹配和原图最大匹配的对称差（异或），由路径和环组成。环不用管，路径的长度至少是 $O(\sqrt{V})$，因为图中的边数是 $O(V)$ 的，所以这样的路径最多有 $O(\sqrt{V})$ 条。

你也许听说过 dinic 跑二分图匹配！我们在这里证明它的复杂度！

因为 dinic 每次匹配最短路长度会至少增加 1，那么做完 $O(\sqrt{V})$ 轮后，最短路长度也至少会达到 $O(\sqrt{V})$。

考虑此时匹配和原图最大匹配的对称差（异或），由路径和环组成。环不用管，路径的长度至少是 $O(\sqrt{V})$，因为图中的边数是 $O(V)$ 的，所以这样的路径最多有 $O(\sqrt{V})$ 条。

两个部分结合，复杂度变成了 $O(E\sqrt{V})$ ！

到这里我们才开始讲建图，建图是网络流最灵魂的部分。而建图的基础就是网络流的性质。

到这里我们才开始讲建图，建图是网络流最灵魂的部分。而建图的基础就是网络流的性质。

一般见得到两大类的建图，一类是直接用网络流性质，一类是利用二分图。

到这里我们才开始讲建图，建图是网络流最灵魂的部分。而建图的基础就是网络流的性质。

一般见得到两大类的建图，一类是直接用网络流性质，一类是利用二分图。

为什么是二分图呢，因为最大流最小割定理，最大流将图分为两个部分，所以和二分图的性质十分契合。

到这里我们才开始讲建图，建图是网络流最灵魂的部分。而建图的基础就是网络流的性质。

一般见得到两大类的建图，一类是直接用网络流性质，一类是利用二分图。

为什么是二分图呢，因为最大流最小割定理，最大流将图分为两个部分，所以和二分图的性质十分契合。

一般情况下如果数据范围小就不用关心复杂度了，因为 dinic 跑的真的是太快辣。

Example (【CQOI2009】dance)

一次舞会有 n 个男孩和 n 个女孩。每首曲子开始时，所有男孩和女孩恰好配成 n 对跳交谊舞。每个男孩都不会和同一个女孩跳两首（或更多）舞曲。有一些男孩女孩相互喜欢，而其他相互不喜欢（不会“单向喜欢”）。每个男孩最多只愿意和 k 个不喜欢的女孩跳舞，而每个女孩也最多只愿意和 k 个不喜欢的男孩跳舞。给出每对男孩女孩是否相互喜欢的信息，舞会最多能有几首舞曲？

$n \leq 50, k \leq 30$。

观察一下题目的限制，发现是个类似于二分图匹配的问题。

观察一下题目的限制，发现是个类似于二分图匹配的问题。
那么 $(u, v, 1)$ 就是表示 u 可以和 v 匹配一次。
(S, u, c) 表示源点到 u，也就是从 u 可以匹配 c 次。

观察一下题目的限制，发现是个类似于二分图匹配的问题。
那么 $(u, v, 1)$ 就是表示 u 可以和 v 匹配一次。
(S, u, c) 表示源点到 u，也就是从 u 可以匹配 c 次。
只能和 k 个不喜欢的，可以拆点。
所以对于每个男孩就可以
$(S, u, c), (u, u', k), (u, like, 1), (u', dislike, 1)$。

观察一下题目的限制，发现是个类似于二分图匹配的问题。

那么 $(u, v, 1)$ 就是表示 u 可以和 v 匹配一次。

(S, u, c) 表示源点到 u，也就是从 u 可以匹配 c 次。

只能和 k 个不喜欢的，可以拆点。

所以对于每个男孩就可以 $(S, u, c), (u, u', k), (u, like, 1), (u', dislike, 1)$。

为了保证每个点都匹配 c 次，我们可以求最大流看看最大流是不是 nc。

观察一下题目的限制，发现是个类似于二分图匹配的问题。

那么 $(u, v, 1)$ 就是表示 u 可以和 v 匹配一次。

(S, u, c) 表示源点到 u，也就是从 u 可以匹配 c 次。

只能和 k 个不喜欢的，可以拆点。

所以对于每个男孩就可以 $(S, u, c), (u, u', k), (u, like, 1), (u', dislike, 1)$。

为了保证每个点都匹配 c 次，我们可以求最大流看看最大流是不是 nc。

当然这个信息是可二分的，于是我们直接二分答案，得到最大的 c 即可。

Example (狼抓兔子)

$n \times m$ 网格图，再加上每个网格的主对角线连边，也就是 $(u, v) \to (u+1, v)$，$(u, v) \to (u, v+1)$，$(u, v) \to (u+1, v+1)$。

每条边有正边权，此时你要删去一些边使得左上角不能到右下角，最小化边权和。

$n, m \leq 1000$。

这道题求的是最小割！由最大流最小割定理，直接求最大流就好了！

这道题求的是最小割！由最大流最小割定理，直接求最大流就好了！

??? 点数非常的多！直接 TLE 了！

这道题求的是最小割！由最大流最小割定理，直接求最大流就好了！

??? 点数非常的多！直接 TLE 了！

考虑到最大流的残余网络将图分为两个点集。因为我们得到的图是平面图，因此看起来就可以用一条线把图分成两半！

这道题求的是最小割！由最大流最小割定理，直接求最大流就好了！

??? 点数非常的多！直接 TLE 了！

考虑到最大流的残余网络将图分为两个点集。因为我们得到的图是平面图，因此看起来就可以用一条线把图分成两半！

具体地，求出图的对偶图，那么割就是新的源点到新的汇点的一条路径。

那么我们求最短路就得到了最小割。

这道题求的是最小割！由最大流最小割定理，直接求最大流就好了！

??? 点数非常的多！直接 TLE 了！

考虑到最大流的残余网络将图分为两个点集。因为我们得到的图是平面图，因此看起来就可以用一条线把图分成两半！

具体地，求出图的对偶图，那么割就是新的源点到新的汇点的一条路径。

那么我们求最短路就得到了最小割。

这道题作为网络流经典题竟然不用网络流！

Example (【TJOI2015】线性代数)

给定一个 $n \times n$ 的矩阵 B 和一个 $1 \times n$ 的矩阵 C。求出一个 $1 \times n$ 的 01 矩阵 A。使得 $D = (A*B-C)*A^T$ 最大, 其中 A^T 为 A 的转置。输出 D。矩阵 B, C 权值非负。

$n \leq 500$。

这道题意思就是有 n 个物品，每个物品有代价，但是如果同时选 i,j 就有 $B_{i,j}$ 的奖励。

这道题意思就是有 n 个物品，每个物品有代价，但是如果同时选 i,j 就有 $B_{i,j}$ 的奖励。

由于是 01 变量，我们假设一个点 u 和源点连通是 0，和汇点连通是 1，同时尽量使用最小割模型解决。最大化权值就是要减去的最少。

这道题意思就是有 n 个物品，每个物品有代价，但是如果同时选 i,j 就有 $B_{i,j}$ 的奖励。

由于是 01 变量，我们假设一个点 u 和源点连通是 0，和汇点连通是 1，同时尽量使用最小割模型解决。最大化权值就是要减去的最少。

由于 C 在答案中是负的，所以对于 i，要和源点连容量为 C_i 的边。

这道题意思就是有 n 个物品，每个物品有代价，但是如果同时选 i,j 就有 $B_{i,j}$ 的奖励。

由于是 01 变量，我们假设一个点 u 和源点连通是 0，和汇点连通是 1，同时尽量使用最小割模型解决。最大化权值就是要减去的最少。

由于 C 在答案中是负的，所以对于 i，要和源点连容量为 C_i 的边。

由于 B 在答案中是正的，对于 i,j 只要任意一者和 S 连通它就要被割掉，也就是说要么 i,j 被割，要么自己被割。

这道题意思就是有 n 个物品，每个物品有代价，但是如果同时选 i,j 就有 $B_{i,j}$ 的奖励。

由于是 01 变量，我们假设一个点 u 和源点连通是 0，和汇点连通是 1，同时尽量使用最小割模型解决。最大化权值就是要减去的最少。

由于 C 在答案中是负的，所以对于 i，要和源点连容量为 C_i 的边。

由于 B 在答案中是正的，对于 i,j 只要任意一者和 S 连通它就要被割掉，也就是说要么 i,j 被割，要么自己被割。

那么就分别从 i 和 j 向 (i,j) 连一条容量为正无穷大的边，那么这条边不可能出现在最小割的方案中。然后 (i,j) 向汇点连容量 $B_{i,j}$ 的边。

Example (【HNOI2013】切糕)

给一个长 P、宽 Q、高 R 的长方体点阵。我们将位于第 z 层中第 x 行、第 y 列上 $(1 \le x \le P, 1 \le y \le Q, 1 \le z \le R)$ 的点称为 (x, y, z)，它有一个非负的不和谐值 $v(x, y, z)$。一个合法的切面满足以下两个条件：

- 与每个纵轴 (一共有 $P \times Q$ 个纵轴) 有且仅有一个交点。即切面是一个函数 $f(x, y)$，对于所有 $1 \le x \le P, 1 \le y \le Q$，我们需指定一个切割点 $f(x, y)$，且 $1 \le f(x, y) \le R$。
- 切面需要满足一定的光滑性要求，即相邻纵轴上的切割点不能相距太远。对于所有的 $1 \le x, x' \le P$ 和 $1 \le y, y' \le Q$，若 $|x - x'| + |y - y'| = 1$，则 $|f(x, y) - f(x', y')| \le D$，其中 D 是给定的一个非负整数。

目标是找出总的切割点上的不和谐值最小的那个，即 $\sum_{x,y} v(x, y, f(x, y))$ 最小。$P, Q, R \le 40$

题面很明显地写出了最小割！我们将每一列 i 与 $i+1$ 连 v_i，然后源点到 1，$R+1$ 到汇点都连无穷大。这样每一列都只割掉一个。

题面很明显地写出了最小割！我们将每一列 i 与 $i+1$ 连 v_i，然后源点到 1，$R+1$ 到汇点都连无穷大。这样每一列都只割掉一个。

由最小割模型，当 $1 \sim i$ 和源点连通，那么相邻的列中 $i-D \sim R$ 也必须存在点和汇点连通。

题面很明显地写出了最小割！我们将每一列 i 与 $i+1$ 连 v_i，然后源点到 1，$R+1$ 到汇点都连无穷大。这样每一列都只割掉一个。

由最小割模型，当 $1 \sim i$ 和源点连通，那么相邻的列中 $i-D \sim R$ 也必须存在点和汇点连通。

于是就是这些点中必须割一条边，那么 i 到相邻列 $i-D$ 连一条正无穷大边就可以解决光滑性要求了。

Example (【NOI2008】志愿者招募)

申奥成功后，布布经过不懈努力，终于成为奥组委下属公司人力资源部门的主管。布布刚上任就遇到了一个难题：为即将启动的奥运新项目招募一批短期志愿者。经过估算，这个项目需要 n 天才能完成，其中第 i 天至少需要 a_i 个人。布布通过了解得知，一共有 m 类志愿者可以招募。其中第 i 类可以从第 s_i 天工作到第 t_i 天，招募费用是每人 c_i 元。新官上任三把火，为了出色地完成自己的工作，布布希望用尽量少的费用招募足够的志愿者，但这并不是他的特长！于是布布找到了你，希望你帮他设计一种最优的招募方案。

$1 \leq n \leq 1000$ $1 \leq m \leq 10000$

这道题是时间轴的网络流。

我们给时间轴创一条路径，那么容量就是最多空闲 $M-a_i$ 人。其中 M 是一个足够大的数。

这道题是时间轴的网络流。

我们给时间轴创一条路径，那么容量就是最多空闲 $M-a_i$ 人。其中 M 是一个足够大的数。

然后 s_i 到 t_i 连一条边，容量为 M 费用为 c_i 就是表示用人。

跑费用流即可.

Example (【UOJ77】A+B Problem)

从前有个 n 个方格排成一行，从左至右依此编号为 $1, 2, \ldots, n$。有一天思考熊想给这 n 个方格染上黑白两色。

第 i 个方格上有 6 个属性：$a_i, b_i, w_i, l_i, r_i, p_i$。如果方格 i 染成黑色就会获得 b_i 的好看度。如果方格 i 染成白色就会获得 w_i 的好看度。

但是太多了黑色就不好看了。如果方格 i 是黑色，并且存在一个 j 使得 $1 \le j < i$ 且 $l_i \le a_j \le r_i$ 且方格 j 为白色，那么方格 i 就被称为奇怪的方格。如果方格 i 是奇怪的方格，就会使总好看度减少 p_i。

也就是说对于一个染色方案，好看度为：

$$\sum_{\text{方格}i\text{为黑色}} b_i + \sum_{\text{方格}i\text{为白色}} w_i - \sum_{\text{方格}i\text{为奇怪的方格}} p_i$$

现在给你 n, a, b, w, l, r, p，问所有染色方案中最大的好看度是多少。$n \le 5000$。

典型的最小鸽，套路就是将加的先加上，减的放入鸽里。

典型的最小鸽，套路就是将加的先加上，减的放入鸽里。
这里还是将点分成两个部分，那么建图很简单。

典型的最小鸽，套路就是将加的先加上，减的放入鸽里。
这里还是将点分成两个部分，那么建图很简单。
假设和源点连通是黑，那么源点连它 w_i，它连汇点 b_i。

典型的最小鸽，套路就是将加的先加上，减的放入鸽里。

这里还是将点分成两个部分，那么建图很简单。

假设和源点连通是黑，那么源点连它 w_i，它连汇点 b_i。

当 $l_i \le a_j \le r_i$，那么当 i 与源点连通，j 与汇点连通，需要割掉的就是 p_i。

也就是 i 连向 j 一条 p_i 的边。

典型的最小鸽，套路就是将加的先加上，减的放入鸽里。

这里还是将点分成两个部分，那么建图很简单。

假设和源点连通是黑，那么源点连它 w_i，它连汇点 b_i。

当 $l_i \le a_j \le r_i$，那么当 i 与源点连通，j 与汇点连通，需要割掉的就是 p_i。

也就是 i 连向 j 一条 p_i 的边。

然而点数很大，边数更大！这里使用区间数据结构优化。

类似于图，区间数据结构只是为了快速地解决点到区间的连边。那么数据结构上的辅助点之间连容量为正无穷大的边就好。

Example (【清华集训 2017】无限之环)

$n \times m$ 网格中，每个格子可以有四个方向的接口，那么一共可以得到十五种水管。

可以任意次顺或逆时针旋转 90 度任意一个非直线型的水管，要求最后每个接口都对着接口。

求最少旋转步数。$nm \leq 2000$。

网格图，考虑黑白染色，把黑色放左边，白色放右边做二分图问题。

网格图，考虑黑白染色，把黑色放左边，白色放右边做二分图问题。

把网格的边（也就是接口连接处）当做一个点，然后接口向这个点连边。如果最后满流就说明有方案。

网格图，考虑黑白染色，把黑色放左边，白色放右边做二分图问题。

把网格的边（也就是接口连接处）当做一个点，然后接口向这个点连边。如果最后满流就说明有方案。

考虑水管的旋转，如果只有一个接口，那么只要向四个方向连边即可。

网格图，考虑黑白染色，把黑色放左边，白色放右边做二分图问题。

把网格的边（也就是接口连接处）当做一个点，然后接口向这个点连边。如果最后满流就说明有方案。

考虑水管的旋转，如果只有一个接口，那么只要向四个方向连边即可。

如果有三个接口，解一下方程，表示每个方向的权值：

$$a+b+c=0, a+b+d=a+c+d=1, b+c+d=2$$

解得 $a=-\frac{2}{3}, b=c=\frac{1}{3}, d=\frac{4}{3}$。然后有手就行。

网格图，考虑黑白染色，把黑色放左边，白色放右边做二分图问题。

把网格的边（也就是接口连接处）当做一个点，然后接口向这个点连边。如果最后满流就说明有方案。

考虑水管的旋转，如果只有一个接口，那么只要向四个方向连边即可。

如果有三个接口，解一下方程，表示每个方向的权值：

$$a+b+c=0, a+b+d=a+c+d=1, b+c+d=2$$

解得 $a=-\frac{2}{3}, b=c=\frac{1}{3}, d=\frac{4}{3}$。然后有手就行。

问题在于接口数等于 2，旋转起来十分麻烦。

其实发现任何旋转都可以用水平或垂直翻转来表达，也就是横纵独立了，这个时候有手就行。

最后费用流即可。

Example (DAG 最小路径覆盖)

给定一个有向无环图，选出最少的路径，使得这些路径两两点不相交，且这些路径包含了图上所有的点。

Example (DAG 最小路径覆盖)

给定一个有向无环图，选出最少的路径，使得这些路径两两点不相交，且这些路径包含了图上所有的点。

Example (DAG 最小链覆盖)

给定一个有向无环图，选出最少的路径，且这些路径包含了图上所有的点。路径可以相交。

Example (DAG 最小路径覆盖)

给定一个有向无环图，选出最少的路径，使得这些路径两两点不相交，且这些路径包含了图上所有的点。

Example (DAG 最小链覆盖)

给定一个有向无环图，选出最少的路径，且这些路径包含了图上所有的点。路径可以相交。

Example (DAG 最长反链)

给定一个有向无环图，选出最多的点，使得这些点两两不可达。

对于最小路径覆盖，给每个点选一个出边，由于图没有环，那么就对应着图上的路径。没有出边的点就是路径的终点。于是跑二分图最大匹配就好了。

对于最小路径覆盖，给每个点选一个出边，由于图没有环，那么就对应着图上的路径。没有出边的点就是路径的终点。于是跑二分图最大匹配就好了。

对于最小链覆盖，由于我们还是可以整成路径不相交，而此时所谓的路径也就是偏序上的链。做一个传递闭包就能转化成最小路径覆盖问题。

对于最小路径覆盖，给每个点选一个出边，由于图没有环，那么就对应着图上的路径。没有出边的点就是路径的终点。于是跑二分图最大匹配就好了。

对于最小链覆盖，由于我们还是可以整成路径不相交，而此时所谓的路径也就是偏序上的链。做一个传递闭包就能转化成最小路径覆盖问题。

对于最长反链，由 Dilworth 定理，可以转化为最小链覆盖的问题。

Example (【JSOI2009】球队收益)

在一个篮球联赛里，有 n 支球队，球队的支出是和他们的胜负场次有关系的，具体来说，第 i 支球队的赛季总支出是 $C_i \times x^2 + D_i \times y^2, D_i \leq C_i$。(赢得多，给球员的奖金就多嘛) 其中 x, y 分别表示这只球队本赛季的胜负场次。现在赛季进行到了一半，每只球队分别取得了 a_i 场胜利和 b_i 场失利。而接下来还有 m 场比赛要进行。问联盟球队的最小总支出是多少。

注：每场比赛是已经钦定好谁打谁的。

$2 \leq n \leq 5000, 0 \leq m \leq 1000, 0 \leq D_i \leq C_i \leq 10, 0 \leq a_i, b_i \leq 50$

如果先给每场比赛钦定谁胜利，那么要后面比赛会要么增加胜利，要么减少胜利，对我们的计算影响很大。

如果先给每场比赛钦定谁胜利，那么要后面比赛会要么增加胜利，要么减少胜利，对我们的计算影响很大。

因此给两个人都是失败！然后选择哪个人是胜利。此时，是胜利加一失败减一。

如果先给每场比赛钦定谁胜利，那么要后面比赛会要么增加胜利，要么减少胜利，对我们的计算影响很大。

因此给两个人都是失败！然后选择哪个人是胜利。此时，是胜利加一失败减一。

对于每只队伍记 $f(x)$ 为增加 x 次胜利带来的支出变化。

这个函数是凸的，所以直接差分网络流即可。

Example (混合图欧拉回路)

给定一个有向边无向边混合的图，给无向边定向，并得到一个欧拉回路。

判断是否有解以及输出解。

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。
首先先给每条无向边随便定向。此时我们只需要修正度数！

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

首先先给每条无向边随便定向。此时我们只需要修正度数！

我们计算每个点的入度减出度，会得到正的点和负的点。每条无向边反向会让一个点给另一个点送 2 的度数。

于是我们事先判断度数的奇偶性，解决无解的情况，就可以把所有值先除以 2。

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

首先先给每条无向边随便定向。此时我们只需要修正度数！

我们计算每个点的入度减出度，会得到正的点和负的点。每条无向边反向会让一个点给另一个点送 2 的度数。

于是我们事先判断度数的奇偶性，解决无解的情况，就可以把所有值先除以 2。

因为所有负点需要增加的值之和等于所有正点需要减少的值之和，我们把它建成二分图，每个点向源点或汇点连它需要的变化量！

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

首先先给每条无向边随便定向。此时我们只需要修正度数！

我们计算每个点的入度减出度，会得到正的点和负的点。每条无向边反向会让一个点给另一个点送 2 的度数。

于是我们事先判断度数的奇偶性，解决无解的情况，就可以把所有值先除以 2。

因为所有负点需要增加的值之和等于所有正点需要减少的值之和，我们把它建成二分图，每个点向源点或汇点连它需要的变化量！

然后用我们根据定好向的无向边在图上连，表示这条边反向会让 u 的值 -1，v 的值 $+1$。

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

首先先给每条无向边随便定向。此时我们只需要修正度数！

我们计算每个点的入度减出度，会得到正的点和负的点。每条无向边反向会让一个点给另一个点送 2 的度数。

于是我们事先判断度数的奇偶性，解决无解的情况，就可以把所有值先除以 2。

因为所有负点需要增加的值之和等于所有正点需要减少的值之和，我们把它建成二分图，每个点向源点或汇点连它需要的变化量！

然后用我们根据定好向的无向边在图上连，表示这条边反向会让 u 的值 -1，v 的值 $+1$。

当我们跑出满流之后，在残余网络上就可得到每条无向边的方向了。

考虑有向图欧拉回路充要条件：弱连通，入度出度相等。

首先先给每条无向边随便定向。此时我们只需要修正度数！

我们计算每个点的入度减出度，会得到正的点和负的点。每条无向边反向会让一个点给另一个点送 2 的度数。

于是我们事先判断度数的奇偶性，解决无解的情况，就可以把所有值先除以 2。

因为所有负点需要增加的值之和等于所有正点需要减少的值之和，我们把它建成二分图，每个点向源点或汇点连它需要的变化量！

然后用我们根据定好向的无向边在图上连，表示这条边反向会让 u 的值 -1，v 的值 $+1$。

当我们跑出满流之后，在残余网络上就可得到每条无向边的方向了。

实际上我们建的图的意义就是事先给一些点加上入度或出度，然后再让网络流使图流量平衡。

Example (Delight for a Cat)

一只猫猫在连续的 n 个小时中可以进行睡觉或进食两种动作。一个小时内只能选择其中一种进行。

现在你知道这只猫在接下来的这 n 个小时中每第 i 个小时睡觉或进食分别获得的快乐值 s_i 和 e_i。

但是对于每一个连续的 k 个小时，这只猫必须满足在这 k 个小时内至少有 m_e 个小时的进食时间和 m_s 个小时的睡觉时间。也就是说在这 n 个小时中的 $n-k+1$ 个 k 长连续区间必须满足睡觉时间 $\geq m_s$，进食时间 $\geq m_e$。

现在小猫想知道自己这 n 个小时最多能获得多少快乐值以及相对应的方案。

可以把条件变为区间中睡觉的时间在一个区间 $[m_s, k - m_e]$ 中。

可以把条件变为区间中睡觉的时间在一个区间 $[m_s, k - m_e]$ 中。

此时假设都在睡觉，如果此时进食的话，睡觉的时间会一直 -1 k 轮，就像志愿者招募那道题一样，相当于征用了一个睡觉的人。

连边注意一下边界情况就好了。

Example (Son of Pipe Stream)

给一张无向图，每条边有容量。给两个源点一个汇点，一共两种液体分别从两个源点出发。

现在要求一对可行流的方案，要求每条边两种液体流的方向相同，且两种液体流量和不能超过边的容量，流量可以是实数。

给定一个常数 $a \in (0,1)$，记第一种液体流为 F，第二种为 G，求 $F^a G^{1-a}$ 的最大值，同时输出方案。

点数 ≤ 300，保证不会有重边。

注：这里题意经过简化，原题经过简单的处理即本题意。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

由于 a 是实数，所以 F 也是实数，这要求我们用两个方案组合出答案的方案。因此我们要考虑流的凸集，显然流的组合是满足流量守恒的。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

由于 a 是实数，所以 F 也是实数，这要求我们用两个方案组合出答案的方案。因此我们要考虑流的凸集，显然流的组合是满足流量守恒的。

最大化 F 我们就要求最大化 S_1 流出的流量。S_2 能不能做到 $M-F$ 呢？因为我们先流 S_1 这部分不会影响最大流，而且 S 到 S_1 的反向边不会再被使用，所以一定能达到。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

由于 a 是实数，所以 F 也是实数，这要求我们用两个方案组合出答案的方案。因此我们要考虑流的凸集，显然流的组合是满足流量守恒的。

最大化 F 我们就要求最大化 S_1 流出的流量。S_2 能不能做到 $M-F$ 呢？因为我们先流 S_1 这部分不会影响最大流，而且 S 到 S_1 的反向边不会再被使用，所以一定能达到。

同理最大化 G 也是一样，这样子我们得到两个流。那么我们就知道 F 应该取什么值。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

由于 a 是实数，所以 F 也是实数，这要求我们用两个方案组合出答案的方案。因此我们要考虑流的凸集，显然流的组合是满足流量守恒的。

最大化 F 我们就要求最大化 S_1 流出的流量。S_2 能不能做到 $M-F$ 呢？因为我们先流 S_1 这部分不会影响最大流，而且 S 到 S_1 的反向边不会再被使用，所以一定能达到。

同理最大化 G 也是一样，这样子我们得到两个流。那么我们就知道 F 应该取什么值。

我们同样可以类似地通过跑两遍求出方案，我们随便网络流跑的并不能满足流向相同，因为存在反向边抵消路径的问题。

首先流量一定是会跑满的，也就是说建一个超级源点连向两个源点，需要满流，得到流 M，这个方案一定满足题目所有条件。

由于 a 是实数，所以 F 也是实数，这要求我们用两个方案组合出答案的方案。因此我们要考虑流的凸集，显然流的组合是满足流量守恒的。

最大化 F 我们就要求最大化 S_1 流出的流量。S_2 能不能做到 $M-F$ 呢？因为我们先流 S_1 这部分不会影响最大流，而且 S 到 S_1 的反向边不会再被使用，所以一定能达到。

同理最大化 G 也是一样，这样子我们得到两个流。那么我们就知道 F 应该取什么值。

我们同样可以类似地通过跑两遍求出方案，我们随便网络流跑的并不能满足流向相同，因为存在反向边抵消路径的问题。

但是跑两遍只是我们的可行性证明。我们直接超级源点给两个源点限定流量，从而确定边的方向，然后直接对两个源点跑两遍即可。

Example (Mole Tunnels)

鼹鼠们在底下开凿了 n 个洞，由 $n-1$ 条隧道连接，对于任意的 $i>1$，第 i 个洞都会和第 $\frac{i}{2}$（取下整）个洞间有一条隧道，第 i 个洞内还有 c_i 个食物能供最多 c_i 只鼹鼠吃。一共有 m 只鼹鼠，第 i 只鼹鼠住在第 p_i 个洞内，一天早晨，前 k 只鼹鼠醒来了，而后 $n-k$ 只鼹鼠均在睡觉，前 k 只鼹鼠就开始觅食，最终他们都会到达某一个洞，使得所有洞的 c_i 均大于等于该洞内醒着的鼹鼠个数，而且要求鼹鼠行动路径总长度最小。现对于所有的 $1 \leq k \leq m$，输出最小的鼹鼠行动路径的总长度，保证一定存在某种合法方案。

$n, m \leq 10^5$。

很容易可以建出费用流模型。

很容易可以建出费用流模型。

很容易发现，由于树的深度不大，我们可以动态维护树上 min 最大的路径。

想方设法维护一下即可。

Example (Conquer the World)

给一棵带权的树，每个点上有 x_i 个军队，每个点需要至少 y_i 个军队。

最小化调动军队的代价来满足每个点的需求，代价为每个军队走的路径长度之和。

节点数 $\leq 2.5 \times 10^5$，军队数量之和 $\leq 10^6$。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

记 $LCA = L$，考虑权值的计算为 $d_S + d_T - 2d_L$，考虑我们在树上进行子树的合并，那么此时 L 是确定的，可以当做常数。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

记 $LCA = L$，考虑权值的计算为 $d_S + d_T - 2d_L$，考虑我们在树上进行子树的合并，那么此时 L 是确定的，可以当做常数。

当一个目的地更换起点的时候，权值变化为 $2d_{L'} - 2d_L - d_S$，其中 d_L 是常量。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

记 $LCA = L$，考虑权值的计算为 $d_S + d_T - 2d_L$，考虑我们在树上进行子树的合并，那么此时 L 是确定的，可以当做常数。

当一个目的地更换起点的时候，权值变化为 $2d_{L'} - 2d_L - d_S$，其中 d_L 是常量。

于是我们在匹配完后，为了做到可撤销，把目的地权值加上 $2d_L - d_S$，起点同理。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

记 $LCA = L$，考虑权值的计算为 $d_S + d_T - 2d_L$，考虑我们在树上进行子树的合并，那么此时 L 是确定的，可以当做常数。

当一个目的地更换起点的时候，权值变化为 $2d_{L'} - 2d_L - d_S$，其中 d_L 是常量。

于是我们在匹配完后，为了做到可撤销，把目的地权值加上 $2d_L - d_S$，起点同理。

这样子每次寻找起点和目的地的最小值，看看变化量是否小于零即可。

问题就是军队和目的地的匹配。这里再维护 min 最大的路径不太合适。我们需要考虑的只是可撤销的问题。

记 $LCA = L$，考虑权值的计算为 $d_S + d_T - 2d_L$，考虑我们在树上进行子树的合并，那么此时 L 是确定的，可以当做常数。

当一个目的地更换起点的时候，权值变化为 $2d_{L'} - 2d_L - d_S$，其中 d_L 是常量。

于是我们在匹配完后，为了做到可撤销，把目的地权值加上 $2d_L - d_S$，起点同理。

这样子每次寻找起点和目的地的最小值，看看变化量是否小于零即可。

需要注意的是我们要匹配完所有目的地，所以强制给目的地一个 $-\infty$ 的权值。剩下的用可并堆维护。

与网络流不同，每条边加上了流量下界！这个问题里没有源汇点，也就是对所有点都要满足流量守恒。

与网络流不同，每条边加上了流量下界！这个问题里没有源汇点，也就是对所有点都要满足流量守恒。

我们把下界流满，必然会得到流量不平衡。我们使用类似混合图欧拉回路的方法，解出应有的流量。

与网络流不同，每条边加上了流量下界！这个问题里没有源汇点，也就是对所有点都要满足流量守恒。

我们把下界流满，必然会得到流量不平衡。我们使用类似混合图欧拉回路的方法，解出应有的流量。

建立源点 S，比如连边 $(u, v, [l, r])$，那就 $(S, v, l), (u, T, l), (u, v, r - l)$，先给点 v 预支一部分入度，此时求解最大流，v 出度会平白无故多 l 以弥补我们删去的下界带来的入度。

如果满流就说明有解，是一个可行流。

有源汇的网络流源汇点不满足流量守恒？

有源汇的网络流源汇点不满足流量守恒？

我们从汇点向源点连一条容量无穷大的辅助边，然后就变成无源汇的问题了！

如果要最大流？

如果要最大流？我们先求可行流。流量平衡后我们删去求解无源汇时的源汇点和辅助边，但是残余网络不要动！

如果要最大流？我们先求可行流。流量平衡后我们删去求解无源汇时的源汇点和辅助边，但是残余网络不要动！

这个时候再直接求最大流，怎么流都是流量守恒的了！同时因为有反向边同样能求到最大流。

二分辅助边容量，求可行流。

二分辅助边容量，求可行流。
太慢了？

二分辅助边容量，求可行流。

太慢了？

求可行流时，我们最小化的是辅助边的流量，也就是要最大化其他部分的流量。

二分辅助边容量，求可行流。

太慢了？

求可行流时，我们最小化的是辅助边的流量，也就是要最大化其他部分的流量。

不管流量守恒，不管辅助边直接开始最大流！当然还是要判一下有解情况的。

由于还是搞流量守恒，所以直接将最大流换成费用流就好。

Warning：以下内容很可能不靠谱。
有些时候费用流建出图就是有负环的！那就跑不了最短路了！

Warning：以下内容很可能不靠谱。

有些时候费用流建出图就是有负环的！那就跑不了最短路了！

那么先要消负环。怎么消呢？我们先假设用掉所有的负边，然后建立其反向边。

Warning：以下内容很可能不靠谱。

有些时候费用流建出图就是有负环的！那就跑不了最短路了！

那么先要消负环。怎么消呢？我们先假设用掉所有的负边，然后建立其反向边。

这个时候会流量不平衡，套用之前调整流量守恒一套理论即可。跑最小费用最大流，就能同时做到流量守恒和删去所有负环。

真的完结了。
谢谢大家。