#0 一些定义

对于带权有向图 G：

- 定义 $(x \to y, w)$ 为一条权值为 w 的从 x 连向 y 的有向边；
- 定义 $V(G)$ 为图 G 的点集，$E(G)$ 为 G 的边集；

#1 什么是网络流

定义 1.1（网络、源点汇点、容量）

> 定义**网络** (G, S, T) 如下：
>
> - G 是一个**边带非负权**的**有向图**，S 和 T 是 G 中的节点；
> - S 被称为**源点**，T 被称为**汇点**；
> - 对于 G 中的边 $(x \to y, w)$，w 被称为改边的**容量**；

定义 1.2（流、流量）

> 对于一个网络 (G, S, T)，定义一组**流** $F = f_{[1,|E(G)|]}$，其中 f_i 为第 i 条边**被使用的流量**，需要满足：
>
> - $\forall 1 \le i \le |E(G)|$，有 $f_i \le w_i$；
> - $\forall u \in V(G), u \notin \{S, T\}$，有 $\sum\limits_{(x_i \to y_i, w_i) \in E(G), y_i = u} f_i = \sum\limits_{(x_i \to y_i, w_i) \in E(G), x_i = u} f_i$，即流入该点的流量和等于流出该点的流量和；
>
> 定义流 F 的**流量** $|F|$ 为 $\sum\limits_{(x_i \to y_i, w_i) \in E(G), y_i = T} f_i$，即流到汇点 T 的流量和。

形象地：

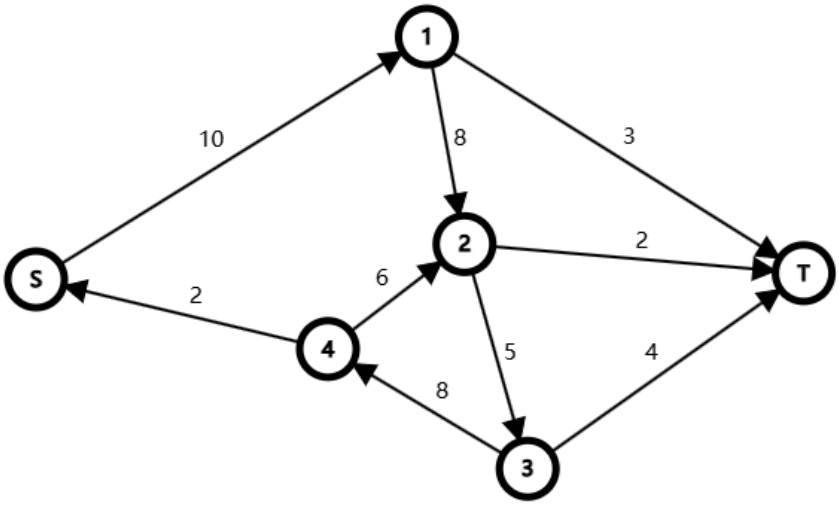
> 对于一个带权有向图 G 及图上的两个点 S（源点）和 T（汇点），可以这样看待这张有向图：
>
> - 将点看作水池，边看作水管，则有向边 $(x \to y, w)$ 表示单位时间内最多 w 单位的水可以通过该水管从水池 x 流向水池 y；
> - 源点 S 是水源，单位时间内能提供无限单位的水；
> - 汇点 T 是排水口，单位时间内能接受无限单位的水；
>
> 流就是让水从水源流到排水口而不爆水管的方案，流量就是方案中从排水口排走的水量。

#2 网络最大流

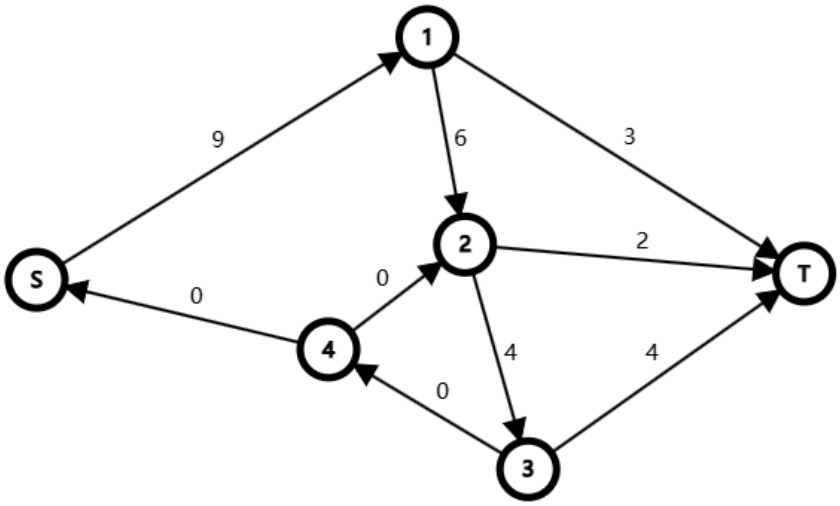
定义 2.1（有源汇最大流）

> 对于一个网络 (G, S, T)，定义其上的**最大流** $\mathrm{MaxFlow}(G, S, T)$ 为**一组满足 $|F|$ 最大的流** F。
>
> 严格的定义理应是满足 $|F|$ 最大的 F 构成的集合，简便起见就不这样记了。

例如对于这个网络：



其一组最大流（流量为 9）如下：（边权为该边使用的流量）



2.1 最大流的求解

2.1.1 反向边的引入

考虑贪心，不断找到从 S 到 T 且容量均 >0 的路径（增广路），即通过这条路径可以让至少 1 的流量从 S 流到 T，那么将答案增加 1，并将路径上边的容量全部减少 1。

定义 2.1.1（增广路）

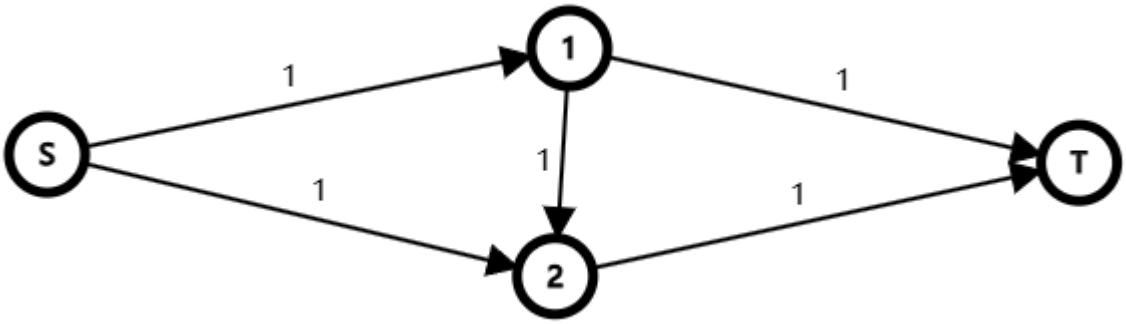
> 定义一条增广路为一条从 S 到 T 路径，满足路径上边的容量均 >0。

定义 2.1.2 pre（增广）

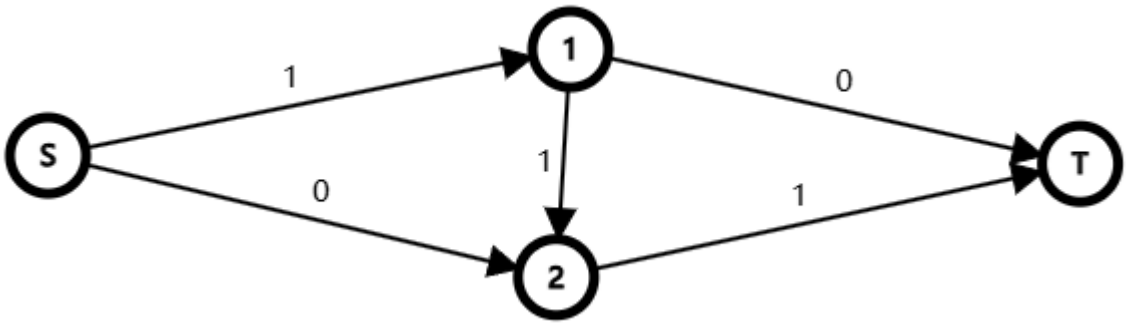
> 对于一条增广路 p，定义增广操作：
>
> - 令 $w(p)=\min\limits_{(x_i\to y_i,w_i)\in p}\{w_i\}$，即 p 中的边的最小容量；
> - 令 p 中边的容量均减少 $w(p)$；
> - 令答案（最大流大小）增加 $w(p)$；

那么上述贪心实际上就是不断找增广路进行增广的过程。

但这个贪心是错的，对于一个这样的最大流大小为 2 的网络：

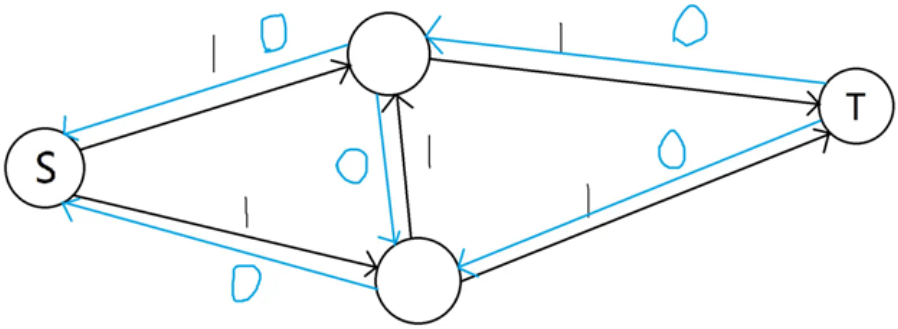


贪心会求出这样的最大流，大小为 1：



因此参考反悔贪心，引入反悔（退流）思想。

具体的，对于每条边 $(x \to y, w)$，建立反向边 $(y \to x, 0)$：

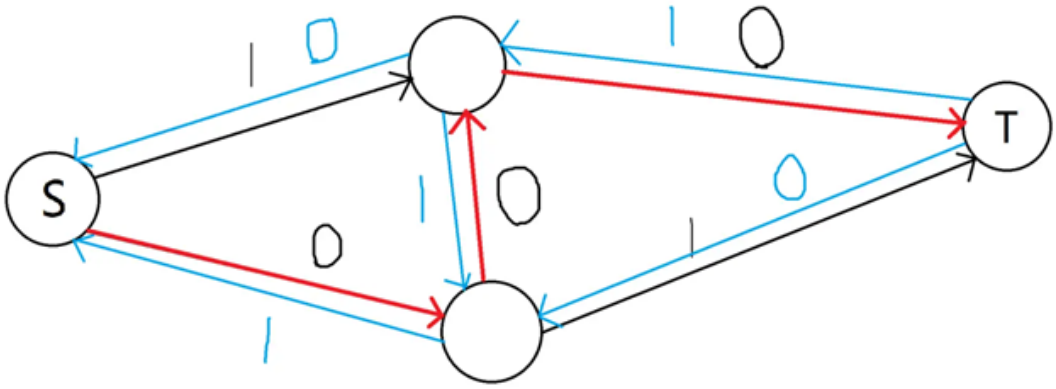


并修改增广的定义：

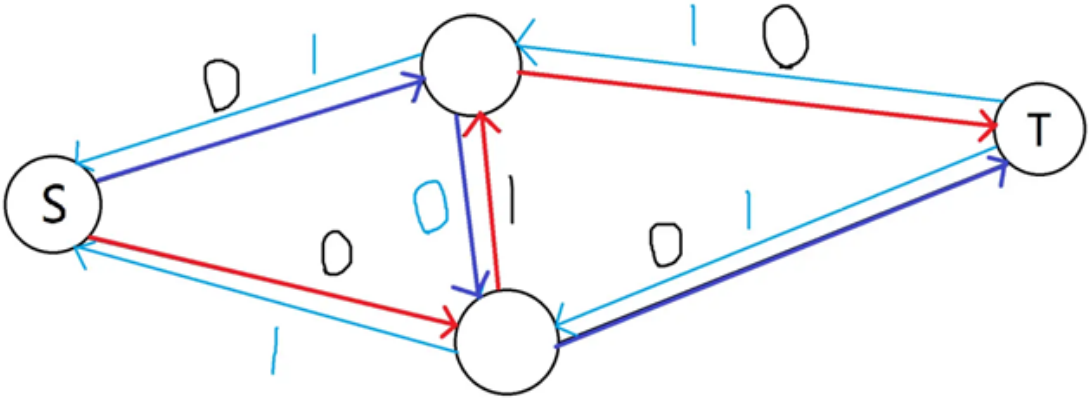
定义 2.1.2（增广）

> 对于一条增广路 p，定义增广操作：
>
> - 令 $w(p) = \min\limits_{(x_i \to y_i, w_i) \in p} \{w_i\}$，即 p 中的边的最小容量；
> - 令 p 中边的容量均减少 $w(p)$，**令 p 中边对应的反向边的容量均增加** $w(p)$；
> - 令答案（最大流大小）增加 $w(p)$；

例如对于例子中的网络，进行一轮新的增广后是这样的：



不难发现现在网络中还存在增广路，还能进行增广操作：



那么修正后的贪心就成功求出了一组最大流，**最终每条边反向边的容量即为这组最大流中该边使用的流量。**

这其实是一个反悔贪心，增广过程中走反向边相当于退流。

容易证明，不断重复上述操作一定能求出一组最大流。

2.1.2 EK 算法

直接模拟上述过程，可以得出求解网络流的朴素算法。

建完图后，用 bfs 来找增广路，并记录下每个点的前驱节点（令其入队的点）以便找到增广路上的边。如果 bfs 时访问到了汇点，那么就找到了一条增广路。

为了储存反向边，不妨令边的编号从 2 开始，并令编号为 i 的边和编号为 $i \oplus 1$ 的边互为反向边。这样实现起来会方便很多。

不断跑 bfs 找增广路去进行增广操作直到 S 和 T 不连通，即可求解出网络的一组最大流。该算法被称作 EK 算法，时间复杂度是 $O((n+m)V)$ 的，其中 V 是最大流的大小，即值域。

2.1.3 dinic 算法

dinic 算法是 EK 算法的一种改进，其可以同时增广多条增广路。

其大体流程如下：

- 每次增广前先跑一次 bfs，将节点按照距离 S 的最短路（边数）dep_u 分层，增广时要求增广路相邻点的 dep_u 恰好相差 1；
- 接下来跑 dfs，对所有满足条件的增广路同时进行增广操作；
- 不断重复上两步直至 bfs 时 S 不能到达 T，即不存在增广路；

伪代码如下：

```
int dfs(int u,int w) // u 表示当前节点的编号，w 表示流向当前点的流量，返回值为成功增广的流量
{
    int sum=0;
    遍历所有从 u 出发的边 i
    {
        int v=i 到达的节点;
        if(dep[v]==dep[u]+1)
        {
            int re=dfs(v,min(w,i 的容量));
            if(re!=0)
            {
                更新正向边和反向边的容量;
                w-=re;
                sum+=re;
            }
        }
    }
    return sum;
}
int dinic()
{
    int res=0;
    while(bfs 可以到达 汇点) res+=dfs(s,inf);
    return res;
}
```

此外，dinic 算法还有几个优化，分别是**当前弧优化**和**断层优化**：

- 当前弧优化：把已经“榨干”（容量为 0）的边删掉；
- 断层优化：把已经“榨干”的节点“删掉”（`return` 时若 `sum==0` 则令 `dep[u]=0`）；

一般默认会加上这两个优化。

dinic 的最劣复杂度为 $O(n^2m)$，证明考虑每次找到的增广路边数一定最小，故 dep_T 递增；而单次 dfs 中每条增广路都会删掉至少一条边（当前弧优化）故最多有 m 条增广路，而每条增广路长度至多为 n。

并且 dinic 在容量均为 1 的网络上的复杂度为 $O(m\sqrt{n})$，具体证明及更多关于复杂度的信息可以看 Dinic的几种复杂度 - myee - 博客园。

但由于网络流题目的图一般都有特殊性质，故加上优化后的 dinic 常常十分快速，基本不会被卡，在 OI 中最常用。

求解网络流还有 ISAP 和 HLPP，由于太过复杂且 OI 中不考，故不再介绍。

#3 最小费用最大流

在最小费用最大流问题中，每条边新增了一个**单位代价** w_i，表示这条边流过 1 流量需要 w_i 的代价，需要求解一组总代价最小的最大流。

也可以使用 dinic 算法解决，每次增广单位代价最小的增广路即可，正确性显然。建边时反向边代价为 $-w_i$，而且 bfs 需要换成 spfa，且增广时要求增广路相邻点的 dep_u 恰好相差它们之间的边的代价。

值得注意的是，由于代价可能为 0，所以 dfs 时需要防止重复访问点。

#4 有向图最小割

定义 4.1（有向图的割）

> 对于一个带权有向图 G 及图上的两个点 S 和 T，定义一组 S 到 T 的割 C 如下：
>
> - $C \in E(G)$；
> - 删掉 C 中的边后 S 不能到达 T；
>
> 定义割 C 的大小 $|C|$ 为 $\sum\limits_{(x_i \to y_i, w_i) \in C} w_i$，即割中的边权和。

定义 4.2（有向图的最小割）

> 对于一个带权有向图 G 及图上的两个点 S 和 T，定义其最小割 $\mathrm{MinCut}(G, S, T)$ 为 $|C|$ 最小的一组割 C。

定理 4.1（最大流等于最小割）

> 对于一个带权有向图 G 及图上的两个点 S 和 T，有：
>
> $$|\mathrm{MaxFlow}(G, S, T)| = |\mathrm{MinCut}(G, S, T)|$$

证明很简单，首先由于跑完最大流算法后 S 不能到达 T，故 $|\mathrm{MaxFlow}(G, S, T)| \geq |\mathrm{MinCut}(G, S, T)|$，又显然 $|\mathrm{MaxFlow}(G, S, T)| \leq |\mathrm{MinCut}(G, S, T)|$（最大流中满流的边集构成割），故得证。

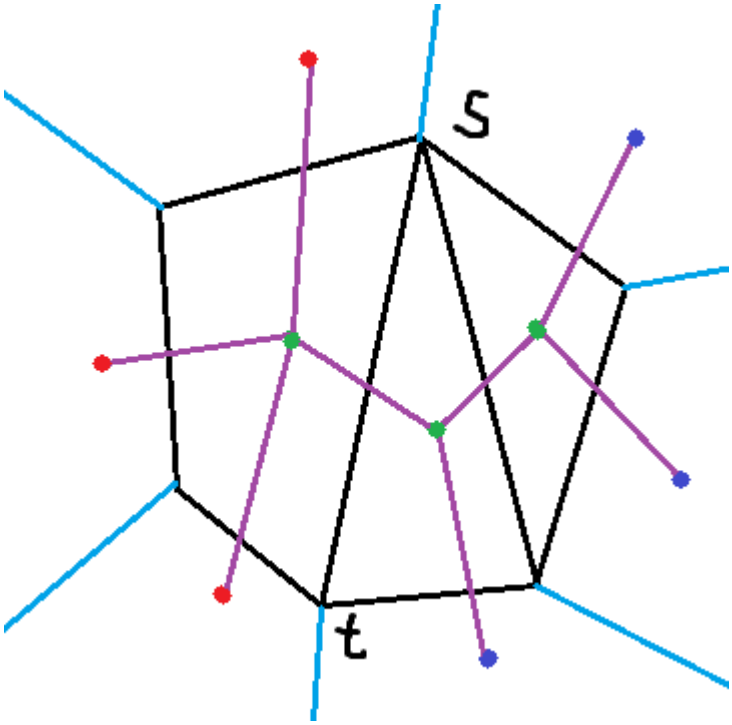
更详细的证明可以自行上网寻找。

并且根据一组最大流 F，可以简单地构造出最小割：

- 令每条边的剩余容量 c_i 为 $w_i - f_i$；
- 从 S 出发，只经过 $c_i > 0$ 的边，求出能到达的点集 A；
- 一组最小割 C 即为 $\{(x \to y, w) \in E(G) | x \in A, y \in (V(G) - A)\}$，即一端在 A 中的边的集合；

定理 4.2（平面图最小割对偶）

> 对于平面图 G 和边界上的两点 S 和 T，再将边界上的每个点都向外引一条射线，则 $\mathrm{MinCut}(G, S, T)$ 等于对偶图中被 S 和 T 划分的两区间的点之间的最短路。



感性理解很容易，理性证明也不难。

考虑任意一条两点集间的路径，其上每一条边对应的原图上的边都被割掉后 S 显然不能到达 T，故证明了 $\leq$；而对于原图上的每一组割，其一定对应对偶图上的一条连接两点集的路径，故证明了 $\geq$，那么证毕。

#5 有上下界网络流

了解即可。

5.1 有上下界循环可行流

> n 个点，m 条有向边，每条边有一个流量下界 l_i 和流量上界 r_i，求该图的一个循环可行流，即你需要构造 m 个变量 f_i 满足：
>
> - 流量合法：$\forall i, l_i \leq f_i \leq r_i$；
> - 流量平衡：对于每个点 i，设 I_i 为 i 的入边集合，U_i 为 i 的出边集合，那么有：$\sum_{j \in I_i} f_j = \sum_{j \in U_i} f_j$；

考虑每条边先钦定流过 l_i 的流量，转化为每条边只有流量上界 $r_i - l_i$，那么流量有可能不平衡。

考虑用点与源点 S 或汇点 T 的连边刻画每个点的额外流量。记 $b_i = \sum_{j \in I_i} l_j - \sum_{j \in U_i} l_j$，即每个点进入的流量减出去的流量，那么：

- 若 $b_i < 0$，连接边 $(i \to T, -b_i)$；
- 若 $b_i > 0$，连接边 $(S \to i, b_i)$；

原先的边 i 直接连 $(x_i \to y_i, r_i - l_i)$。

跑最大流，若所有 $(i \to T, -b_i)$ 和 $(S \to i, b_i)$ 均满流，则有解，否则无解。设跑完后第 i 条边的流量为 c_i，则第 i 条边的流量为 $l_i + c_i$。

时间复杂度和最大流相同。

5.2 有上下界有源汇最大流

考虑将其转化为循环流来消去上下界，设源点为 s，汇点为 t，则不难发现相当于从 s 出发的流量等于到达 t 的流量，连边 $(t \to s, [0, \infty])$。

跑出循环流后，记 res 为 $(t \to s, [0, \infty])$ 的。接下来把 $(t \to s, [0, \infty])$ 去掉，设第 i 条边在循环流中流过了 c_i 流量，则连接边 $(x_i \to y_i)$ 时：

- 正向边流量为 $r_i - c_i$；
- 反向边流量为 $c_i - l_i$；

则答案即为 res 加上当前图从 s 到 t 的最大流。

5.3 有上下界有源汇最小流

最大流是尽可能从 s 流到 t，那么最小流就是尽可能从 t 退回 s。

跑出 t 到 s 的最大流 rf，答案即为 $res - rf$。

#6 例题&技巧

6.1 最大流

6.1.1 P2472 [SCOI2007] 蜥蜴

> 在一个 r 行 c 列的网格地图中有一些高度不同的石柱，第 i 行 j 列的石柱高度为 $h_{i,j}$。
>
> 一些石柱上站着一些蜥蜴，你的任务是让尽量多的蜥蜴逃到边界外。
>
> 每行每列中相邻石柱的距离为 1，蜥蜴的跳跃距离是 d，即蜥蜴可以跳到**平面距离**不超过 d 的任何一个石柱上。
>
> 每次当蜥蜴跳跃时，所离开的石柱高度减 1，到达的石柱高度不变（如果有）。如果该石柱原来高度为 1，则蜥蜴离开后消失，以后其他蜥蜴不能落脚。
>
> 任何时刻不能有两只蜥蜴在同一个石柱上。
>
> 求最多有多少只蜥蜴可以逃到边界外。
>
> 对于 100% 的数据满足：$1 \le r, c \le 20$，$1 \le d \le 4$，$1 \le h \le 3$。

▶ 题解

6.1.2 P3163 [CQOI2014]危桥

> Alice 和 Bob 居住在一个由 N 座岛屿组成的国家，岛屿被编号为 0 到 $N-1$。某些岛屿之间有桥相连，桥上的道路是双向的，但一次只能供一人通行。其中一些桥由于年久失修成为危桥，最多只能通行两次。
>
> Alice 希望在岛屿 a_1 和 a_2 之间往返 a_n 次（从 $a1$ 到 $a2$ 再从 $a2$ 到 $a1$ 算一次往返）。同时，Bob 希望在岛屿 b_1 和 b_2 之间往返 b_n 次。这个过程中，所有危桥最多通行两次，其余的桥可以无限次通行。请问 Alice 和 Bob 能完成他们的愿望吗?
>
> $4 \le N \le 50,\ 0 \le a_1, a_2, b_1, b_2 \le N-1,\ 1 \le a_n, b_n \le 50$。

▶ 题解

6.1.3 ARC156F Make Same Set

> 给定三个长 n 的序列 A, B, C，找到一个符合以下条件且大小最大的集合 S：

- 其能被这样生成：对于每个 $i \in [1,n]$，向 S 中加入 A_i 或者 B_i；
- 其能被这样生成：对于每个 $i \in [1,n]$，向 S 中加入 A_i 或者 C_i；

需要输出方案。

$1 \le n \le 5000$，$1 \le A_i, B_i, C_i \le 10000$。

▶ 题解

6.2 最小割

设点 u 最终在源点 S 所在的连通块（能通过容量 > 0 的边到达）则 $b_u = 1$，否则 $b_u = 0$。

考虑最小割的数学定义：

- 边 $(S \to x, a)$ 对答案的贡献为 $a \times (1 - b_x)$；
- 边 (x, T, a) 对答案的贡献为 $a \times b_x$；
- 边 (x, y, a) 对答案的贡献为 $a \times b_x(1 - b_y)$；

所以基本限制如下：

有 n 个 01 变量 b_i，你需要确定每个 b_i 的取值使得代价最小。

代价分为三类：

- 若 $b_i = 0$，有代价 $w1_i$；
- 若 $b_i = 1$，有代价 $w2_i$；
- 若 $b_i = 1$ 且 $b_j = 0$，有代价 $w3_{i,j}$；

基本模型如下：

给第 i 个元素建一个点 i。

- 对于第一类限制，连边 $(S, i, w1_i)$；
- 对于第二类限制，连边 $(i, T, w2_i)$；
- 对于第三类限制，连边 $(i, j, w3_{i,j})$；

6.2.1 QOJ1197 Draw in Straight Lines

有一个 $n \times m$ 的网格，刚开始每个格子都是白色的，你要把 (i, j) 的颜色通过以下三种方式变为 $a_{i,j}$：

- 涂白横着/竖着排列的若干格子，若涂了 k 格，则代价为 $Ak + B$；
- 涂黑横着/竖着排列的若干格子，若涂了 k 格，则代价为 $Ak + B$；
- 涂黑/涂白单独某个格子，代价为 C；

一个格子不能被涂色超过两次，并且一个格子不能被涂白再涂黑。

求最小代价。

$1 \le n, m \le 40$，$0 \le A, B, C \le 40$，$C \le A + B$。

▶ 题解

6.2.2 P1361 小M的作物

小 M 在 MC 里开辟了两块巨大的耕地 A 和 B（你可以认为容量是无穷），现在，小 P 有 n 种作物的种子，每种作物的种子有 1 个（就是可以种一棵作物），编号为 1 到 n。

> 现在，第 i 种作物种植在 A 中种植可以获得 a_i 的收益，在 B 中种植可以获得 b_i 的收益，而且，现在还有这么一种神奇的现象，就是某些作物共同种在一块耕地中可以获得额外的收益，小 M 找到了规则中共有 m 种作物组合，第 i 个组合中的作物共同种在 A 中可以获得 $c_{1,i}$ 的额外收益，共同种在 B 中可以获得 $c_{2,i}$ 的额外收益。
>
> 小 M 很快的算出了种植的最大收益，但是他想要考考你，你能回答他这个问题么?
>
> $1 \leq k < n \leq 10^3$，$1 \leq m \leq 10^3$。题目当中出现的所有权值均为不大于 1000 的非负整数。

▶ 题解

6.2.3 P3227 [HNOI2013] 切糕

> 形式化题意大致如下：
>
> 有 n 个整数变量 b_i，第 i 个变量取值范围 $[L_i, R_i]$。
>
> 代价分为两类：
>
> - $b_i = x$ 时有代价 $W_{i,x}$；
> - $b_i \geq x, b_j \geq y$ 时有代价 $W_{i,x,j,y}$；
>
> 求最小代价。

▶ 题解

6.2.4 P8215 [THUPC 2022 初赛] 分组作业

> 有 $2n$ 个人，按照 $(1,2),(3,4),\ldots,(n-1,n)$ 的顺序分成 n 组。
>
> 每个人处于两种状态之一：愿意、不愿意。
>
> 一组两个人都愿意时可以选择合作或不合作，两人有任意一个不愿意都不可合作。
>
> 代价：
>
> - 第 i 个人选择愿意代价为 c_i，选择不愿意代价为 d_i，i 选愿意 i 的队友选不愿意代价为 e_i；
> - 有 m 个四元组 (A,B,a,b) 代表：
>   - A 和 A 的队友不合作（任意一人不愿意或者都愿意且选择不合作）且 B 选择愿意，付出 a 的代价；
>   - B 和 B 的队友合作（都选择愿意且选择愿意合作）且 A 选择不愿意，付出 b 的代价。
>
> 给定上述的 n,m,c,d,e 和 (A,B,a,b)，求最小代价。
>
> 保证 $1 \leq n \leq 5000$，$0 \leq m \leq 10000$，$1 \leq a_i, b_i, c_i, d_i, e_i \leq 10^9$。

▶ 题解

6.2.5 GYM102471K All Pair Maximum Flow

> 给定一个 n 个点 m 条边的无向图，每条边有流量 w_i，保证其是一个逆时针从 1 到 n 编号的正 n 边形，且边只会在顶点处相交，求 n 个点两两间的最大流之和，对 998244353 取模。
>
> $3 \leq n \leq 2 \times 10^5$，$n \leq m \leq 4 \times 10^5$，$0 \leq w_i \leq 10^9$。

▶ 题解