费用流模型选讲

刘昀

天津市南开中学

2018年1月10日

在流量最大的前提下，最小化费用

在流量最大的前提下，最小化费用

- spfa费用流，适用于流量较小、费用较大的情况

在流量最大的前提下，最小化费用

- spfa费用流，适用于流量较小、费用较大的情况
- zkw费用流，适用于流量较大、费用较小的情况

无源汇上下界最小费用可行流

无源汇上下界最小费用可行流

- 类比无源汇上下界可行流即可

无源汇上下界最小费用可行流

- 类比无源汇上下界可行流即可
- 上下界最小费用最大流、最小流也可类比网络流

最小费用可行流（每个点有盈余量e），如何消除负边和负圈？

最小费用可行流（每个点有盈余量e），如何消除负边和负圈？

把所有负边(x, y)强制满流，修改e_x和e_y。

最小费用最大流，如何消除负边和负圈？

最小费用最大流，如何消除负边和负圈？

【方法一】

$T \to S$连边$(+\infty, 0)$，将所有负边强制满流。

先建立超级源汇求最小费用最大流，再求$S \to T$的最小费用最大流。

最小费用最大流，如何消除负边和负圈？

【方法一】

$T \to S$连边$(+\infty, 0)$，将所有负边强制满流。

先建立超级源汇求最小费用最大流，再求$S \to T$的最小费用最大流。

【方法二】

$T \to S$连边$(+\infty, -x)$，其中x是一个大数，将所有负边强制满流。

最小费用最大流，如何消除负边和负圈？

【方法一】

$T \to S$连边$(+\infty, 0)$，将所有负边强制满流。

先建立超级源汇求最小费用最大流，再求$S \to T$的最小费用最大流。

【方法二】

$T \to S$连边$(+\infty, -x)$，其中x是一个大数，将所有负边强制满流。

建立超级源汇求最小费用最大流，由于$-x$最优，故此时已保证原图得到最大流。

最小费用最大流，如何消除负边和负圈？

【方法一】

$T \to S$连边$(+\infty, 0)$，将所有负边强制满流。

先建立超级源汇求最小费用最大流，再求$S \to T$的最小费用最大流。

【方法二】

$T \to S$连边$(+\infty, -x)$，其中x是一个大数，将所有负边强制满流。

建立超级源汇求最小费用最大流，由于$-x$最优，故此时已保证原图得到最大流。

计算费用中$-x$的个数，即可得到最大流的数值。

有n种数字，第i种数字是a_i，有b_i个，权值是c_i。

若两个数字a_i、a_j满足a_i是a_j的质数倍，那么它们可以配对，获得$c_i \times c_j$的收益。

每个数字最多配对1次。

在收益非负的前提下，最大化配对次数。

$n \leq 200$

把每个数质因数分解，按质因子指数之和的奇偶性建出二分图。

费用流，不断增广直至收益< 0。

有一个芯片，芯片上有$n \times n$个插槽，可以在里面装零件。

插槽分为3种：不能装零件、必须装零件、可装可不装。

要求装好所有零件后必须满足下列要求：

- 第i行和第i列的零件数量必须相等
- 第i行的零件数量不能超过总零件数量的$\frac{A}{B}$

最大化总零件数量。

$n \leq 40$

一个直观的想法：行列建二分图，3种插槽都通过容量设定上下界来限制。

一个直观的想法：行列建二分图，3种插槽都通过容量设定上下界来限制。

答案能不能二分？

一个直观的想法：行列建二分图，3种插槽都通过容量设定上下界来限制。

答案能不能二分？

枚举总零件数量，判断此时最大流能否达到这个数量。

一个直观的想法：行列建二分图，3种插槽都通过容量设定上下界来限制。

答案能不能二分？

枚举总零件数量，判断此时最大流能否达到这个数量。

但是我们没有限制第i行和第i列的零件数量相等，怎么办？

为了处理这个条件，我们需要引入费用。

为了处理这个条件，我们需要引入费用。

在第i行和第i列之间连$(+\infty, 0)$的边，每个零件对应的边增设费用1。

为了处理这个条件，我们需要引入费用。

在第i行和第i列之间连$(+\infty, 0)$的边，每个零件对应的边增设费用1。

此时的最大流必然是满流，而为使流量最大，$(+\infty, 0)$边的两端点其余边流量之和必然相等，即满足了这个条件。

为了处理这个条件，我们需要引入费用。

在第i行和第i列之间连$(+\infty, 0)$的边，每个零件对应的边增设费用1。

此时的最大流必然是满流，而为使流量最大，$(+\infty, 0)$边的两端点其余边流量之和必然相等，即满足了这个条件。

同样是枚举总零件数量，但此时通过上下界最大费用最大流来判断可行性。

一个小技巧：

对于必须装零件的插槽，我们可以去掉其流量下界，并把费用增加一个大数x。

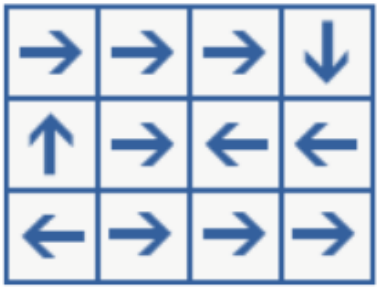
这样去掉了所有下界。

直接求解最大费用最大流，计算最大费用包含了多少个x，判断是否满足下界。

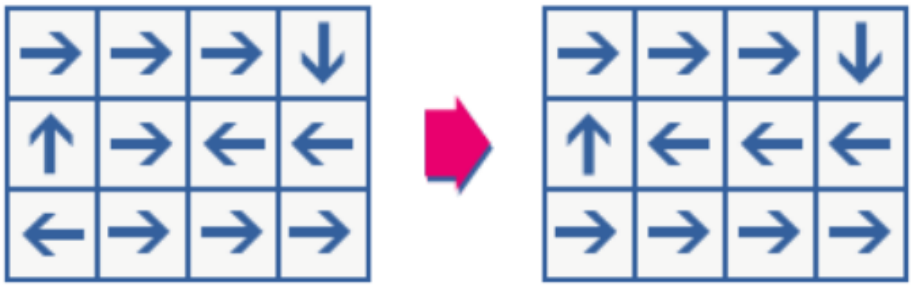
这类问题通常是限制路径为若干回路。

和网络流类似，通过度数限制建模。

给出一个$n \times m$的网格，每个位置都有一个箭头。



修改最少的箭头，使箭头构成若干回路。



$n \leq 15$，$m \leq 15$

回路的性质？

构成若干回路，当且仅当每个点出度=入度=1。

回路的性质？

构成若干回路，当且仅当每个点出度=入度=1。

出度已保证，我们只需保证入度。

回路的性质？

构成若干回路，当且仅当每个点出度=入度=1。

出度已保证，我们只需保证入度。

左边箭头右边点，考虑每个箭头会影响哪些点的入度，若方向改变则增设费用1。

回路的性质？

构成若干回路，当且仅当每个点出度=入度=1。

出度已保证，我们只需保证入度。

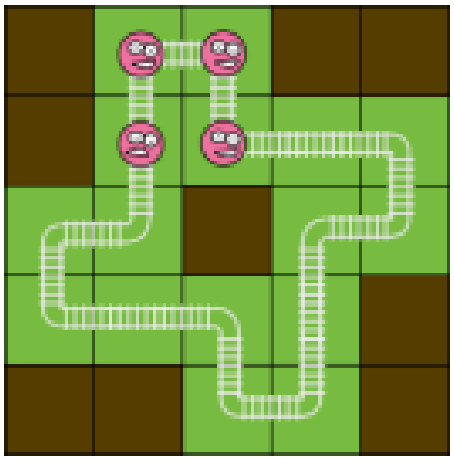
左边箭头右边点，考虑每个箭头会影响哪些点的入度，若方向改变则增设费用1。

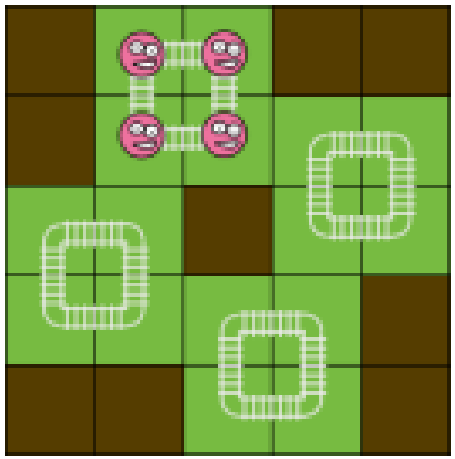
二分图最小权匹配。

$n \times m$的网格图，格子为空地或障碍。

你需要在每块空地上铺铁轨，铺成若干回路。

对于某些空地，若上面的铁路是直的，你必须付出1的代价。





最小化总代价。

$n \leq 30$，$m \leq 30$

首先判断是否有解，网络流经典问题。

构成若干回路，当且仅当每个点度数=2。

黑白染色后二分图建模。

如何处理代价？

如何处理代价？

对于每个点，什么情况下会产生1的代价？

如何处理代价？

对于每个点，什么情况下会产生1的代价？

当且仅当它的两条边被分配到了同一行或同一列。

如何处理代价？

对于每个点，什么情况下会产生1的代价？

当且仅当它的两条边被分配到了同一行或同一列。

这就启发我们，对于每个点，分开行列。拆点试试？

如何处理代价？

对于每个点，什么情况下会产生1的代价？

当且仅当它的两条边被分配到了同一行或同一列。

这就启发我们，对于每个点，分开行列。拆点试试？

行点和左右两点连边，列点和上下两点连边，行点→列点连$(1,1)$的边。

如何处理代价？

对于每个点，什么情况下会产生1的代价？

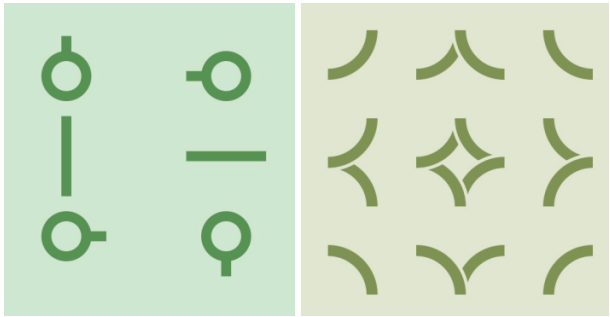
当且仅当它的两条边被分配到了同一行或同一列。

这就启发我们，对于每个点，分开行列。拆点试试？

行点和左右两点连边，列点和上下两点连边，行点→列点连$(1,1)$的边。

最小费用最大流即为答案。

一个$n \times m$的网格，其中有些位置存在水管，水管有以下15种形状：



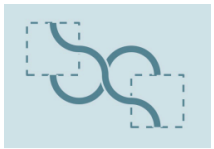
水管可能在方格某些方向的边界的中点有接口，若存在某个接口没有和其它接口相接，则水管会漏水。

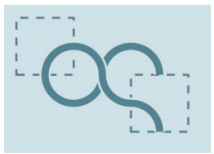
你可以进行一种操作：选定一个非直线型水管，将其顺时针或逆时针旋转90度。

在水管不漏水的前提下，最小化操作次数。

$n \times m \leq 2000$

一个样例：





最少需要2次操作。

不是回路，每个点度数0到4都有可能，似乎难以下手……

不是回路，每个点度数0到4都有可能，似乎难以下手……

但它具有回路的某些特点，同样是从度数角度思考。

不是回路，每个点度数0到4都有可能，似乎难以下手……

但它具有回路的某些特点，同样是从度数角度思考。

合法的图形具有什么度数性质？

不是回路，每个点度数0到4都有可能，似乎难以下手……

但它具有回路的某些特点，同样是从度数角度思考。

合法的图形具有什么度数性质？

考虑网格的边。

不是回路，每个点度数0到4都有可能，似乎难以下手……

但它具有回路的某些特点，同样是从度数角度思考。

合法的图形具有什么度数性质？

考虑网格的边。

终于还是发现了性质：一个图案合法，当且仅当网格的每条边两侧接口存在性相同，类似于出度入度的平衡条件。

对网格图黑白染色，把每个点拆成4个点表示上下左右4个接口。

对网格图黑白染色，把每个点拆成4个点表示上下左右4个接口。

若黑白两部分接口总数不相等，显然无解。

对网格图黑白染色，把每个点拆成4个点表示上下左右4个接口。

若黑白两部分接口总数不相等，显然无解。

先不考虑旋转操作，建出一个直观模型：

$S \rightarrow$黑点接口，白点接口$\rightarrow T$，黑白相邻接口连边，容量均为1。

对网格图黑白染色，把每个点拆成4个点表示上下左右4个接口。

若黑白两部分接口总数不相等，显然无解。

先不考虑旋转操作，建出一个直观模型：

$S \rightarrow$黑点接口，白点接口$\rightarrow T$，黑白相邻接口连边，容量均为1。

最大流S和T均满流即有解。

旋转操作会带来什么影响？

旋转操作会带来什么影响？

接口方向改变，但接口数量不变。

旋转操作会带来什么影响？

接口方向改变，但接口数量不变。

于是和刚才一样，最大流S和T均满流即有解。

旋转操作会带来什么影响？

接口方向改变，但接口数量不变。

于是和刚才一样，最大流S和T均满流即有解。

最小化操作次数，容易想到增设费用一维。

旋转操作会带来什么影响？

接口方向改变，但接口数量不变。

于是和刚才一样，最大流S和T均满流即有解。

最小化操作次数，容易想到增设费用一维。

大体思路有了，具体如何建图呢？

旋转操作会带来什么影响？

接口方向改变，但接口数量不变。

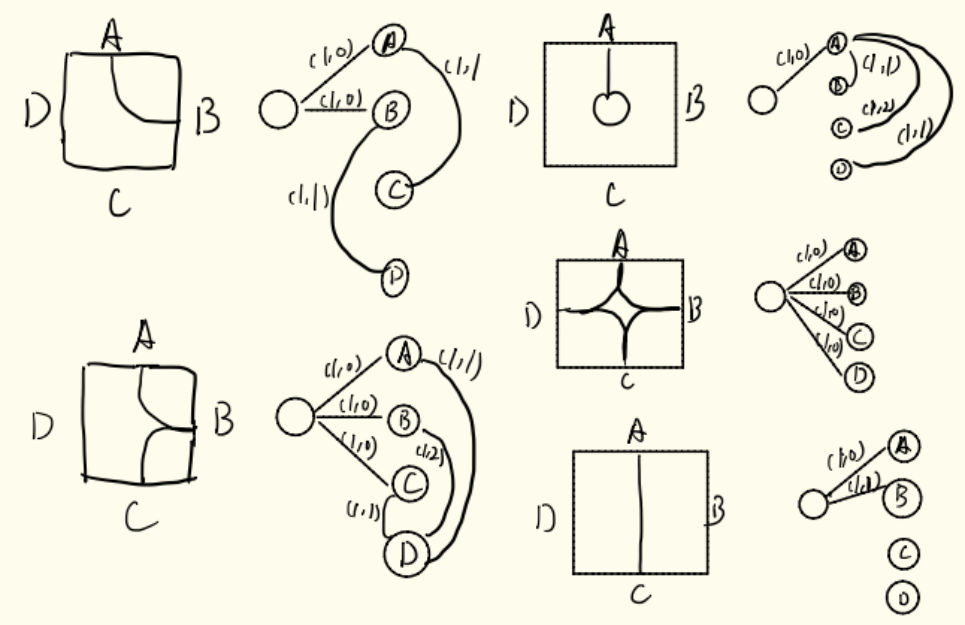
于是和刚才一样，最大流S和T均满流即有解。

最小化操作次数，容易想到增设费用一维。

大体思路有了，具体如何建图呢？

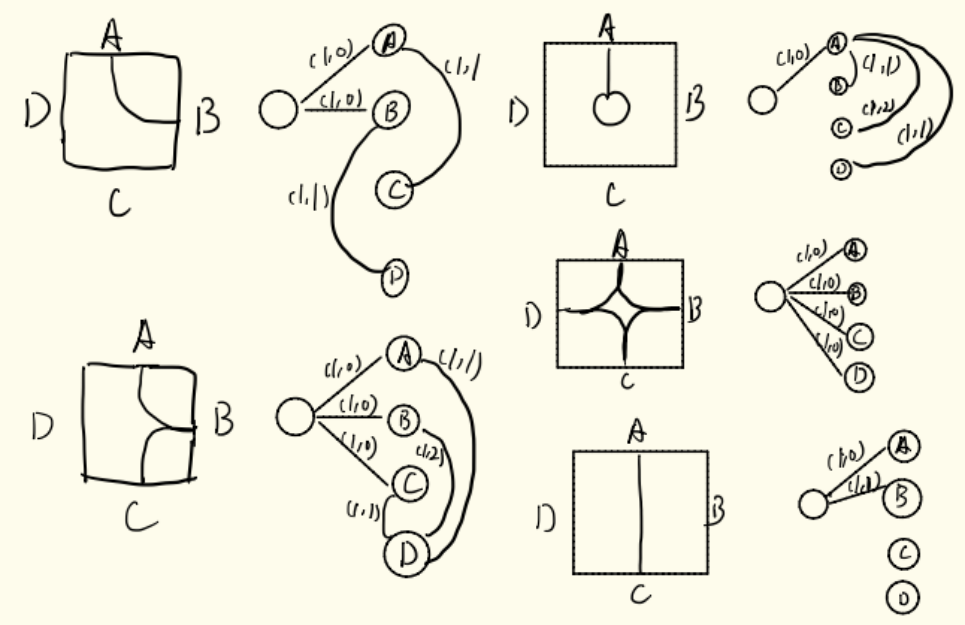
这时就需要对本题水管的形状分类讨论了！

所有格子大致分为6种，建图如下：



注意直线型水管不能旋转。

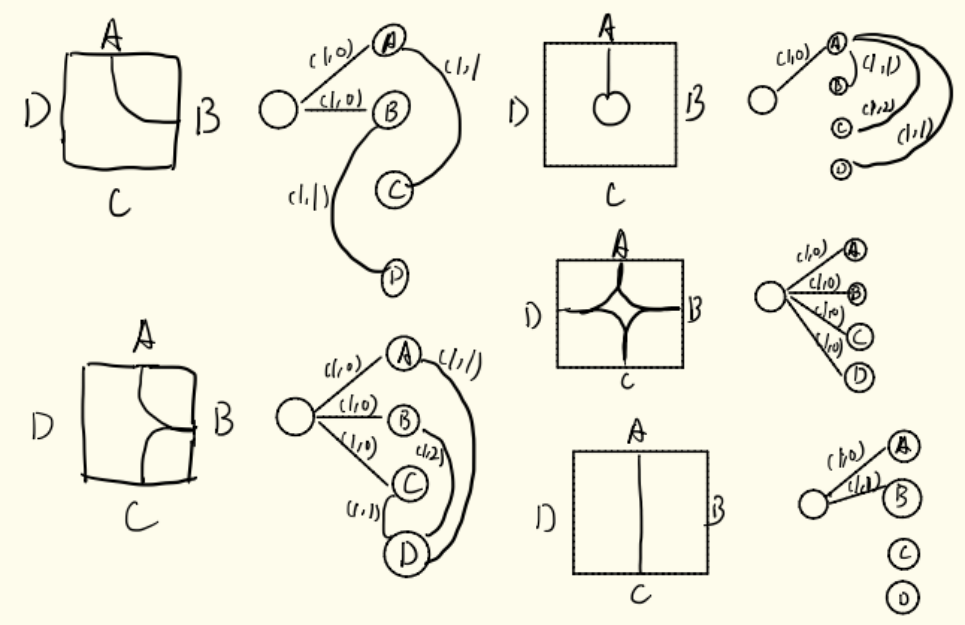
所有格子大致分为6种，建图如下：



注意直线型水管不能旋转。

至此建模完成，最小费用最大流即为答案。

所有格子大致分为6种，建图如下：



注意直线型水管不能旋转。

至此建模完成，最小费用最大流即为答案。

若直线型水管可以旋转，若只允许顺时针旋转，又应如何建模？

欢迎课后讨论！

这类问题通常是一个物品可以选择多次，而每次选择的代价递增。

最小费用最大流必然优先选择代价较小的边，符合要求。

有n种菜和m个厨师，已知每个厨师制作每一种菜的时间。

有p个学生，每个学生都点了一道菜。

你需要把这p道菜的制作任务交给这m个厨师，你可以任意分配厨师，任意决定顺序。

每个学生的等待时间为从开始到自己的菜完成为止的时间总长度，最小化等待时间之和。

$n \leq 40$，$m \leq 100$，$p \leq 800$

考虑每个厨师的第i道菜对答案的贡献。

考虑每个厨师的第i道菜对答案的贡献。

设这个厨师一共做了s道菜，那么贡献=做菜时间$\times(s-i+1)$)。

考虑每个厨师的第i道菜对答案的贡献。

设这个厨师一共做了s道菜，那么贡献=做菜时间$\times(s-i+1)$)。

费用递减，且我们不知道s的值。

费用递减，最短路每次优先选出靠后的位置。

这给了我们启发，我们可以倒过来考虑。

费用递减，最短路每次优先选出靠后的位置。

这给了我们启发，我们可以倒过来考虑。

考虑每个厨师的倒数第i道菜对答案的贡献，贡献=做菜时间$\times i$。

费用递增，最小费用最大流符合要求。

费用递减，最短路每次优先选出靠后的位置。

这给了我们启发，我们可以倒过来考虑。

考虑每个厨师的倒数第i道菜对答案的贡献，贡献=做菜时间$\times i$。

费用递增，最小费用最大流符合要求。

把每个厨师拆成p个点，第i个点表示倒数第i道菜，限制容量为1即可。

设$spfa$为线性，这个做法的时间复杂度为$O(nmp^2)$，不能通过此题。

注意到图中存在大量无意义的边，我们可以只建出少量的边，一边增广一边加边。

初始只有$O(nm)$条边，每次增广后加入$O(1)$条边。

时间复杂度为$O(nmp + p^2)$，通过此题。

有n个球队，目前第i个球队赢了win_i场，输了$lose_i$场。

接下来有m场比赛，已知每场比赛的双方球队，你可以决定这m场比赛的输赢。

对于第i个球队，若其最终赢x场，输y场，则获得收益$C_i \times x^2 + D_i \times y^2$。

最小化所有球队的总收益。

$n \leq 5000$，$m \leq 1000$，$0 \leq D_i \leq C_i \leq 10$。

若输球不带来影响，即$D_i = 0$。

若输球不带来影响，即$D_i = 0$。

$C_i \geq 0$，一个球队赢的场数越多，每多赢一场获得的收益越大，符合模型。

每场比赛和双方球队连边，容量为1，一条增广路表示一个获胜事件。

若输球带来影响，则这个方法一场比赛会带来两个影响，不好处理。

若输球带来影响，则这个方法一场比赛会带来两个影响，不好处理。

不妨假设每场比赛双方全输，每场比赛会使一个球队多赢一场且少输一场。

若输球带来影响，则这个方法一场比赛会带来两个影响，不好处理。

不妨假设每场比赛双方全输，每场比赛会使一个球队多赢一场且少输一场。

设第i个球队当前赢a场，输b场，则球队多赢一场比赛的收益增量

$$= C_i \times ((a+1)^2 - a^2) + D_i \times ((b-1)^2 - b^2)$$
$$= C_i \times (2 \times a + 1) - D_i \times (2 \times b - 1)$$

若输球带来影响，则这个方法一场比赛会带来两个影响，不好处理。

不妨假设每场比赛双方全输，每场比赛会使一个球队多赢一场且少输一场。

设第i个球队当前赢a场，输b场，则球队多赢一场比赛的收益增量

$$= C_i \times ((a+1)^2 - a^2) + D_i \times ((b-1)^2 - b^2)$$

$$= C_i \times (2 \times a + 1) - D_i \times (2 \times b - 1)$$

$C_i, D_i \geq 0$，故a越大，多赢一场比赛的收益增量越大。

直接套用模型。

n个点的完全图，某些边已经确定了方向。

你需要给其余的边定向，最大化三元环的个数，输出一种方案。

$n \leq 100$

合法的三元组满足什么条件？

合法的三元组满足什么条件？

不合法的三元组满足什么条件？

合法的三元组满足什么条件？

不合法的三元组满足什么条件？

若一个三元组不合法，当且仅当存在一个点A指向另外两个点。

合法的三元组满足什么条件？

不合法的三元组满足什么条件？

若一个三元组不合法，当且仅当存在一个点A指向另外两个点。

我们认为点A“贡献”了这个不合法三元组。

若点A出度为d，由于原图是完全图，故从这d条边中任选2条边都出现一个不合法三元组。

若点A出度为d，由于原图是完全图，故从这d条边中任选2条边都出现一个不合法三元组。

故一个点的贡献仅和它的出度d有关，贡献$=\frac{d\times(d-1)}{2}=\frac{d^2}{2}-\frac{d}{2}$。

若点A出度为d，由于原图是完全图，故从这d条边中任选2条边都出现一个不合法三元组。

故一个点的贡献仅和它的出度d有关，贡献$=\frac{d\times(d-1)}{2}=\frac{d^2}{2}-\frac{d}{2}$。

最大化三元环的个数，即最小化所有点的贡献之和。

若点A出度为d，由于原图是完全图，故从这d条边中任选2条边都出现一个不合法三元组。

故一个点的贡献仅和它的出度d有关，贡献$=\frac{d\times(d-1)}{2}=\frac{d^2}{2}-\frac{d}{2}$。

最大化三元环的个数，即最小化所有点的贡献之和。

- $\frac{d}{2}$的部分$O(1)$计算。
- $\frac{d^2}{2}$的部分满足费用递增，费用流解决，利用残量网络构造方案。

这类问题通常会限制必须遍历一些点，或一些点必须遍历多少次。

把每个点拆成2个点，一个点$\rightarrow T$，$S\rightarrow$另一个点。

和T相连的点视为到此签到，和S相连的点视为从此出发。

事实上，这类问题也可通过对每个点限制上下界直接解决，且建出的图一模一样。

有n天，每天都需要不同数量的餐巾，有3种提供餐巾的方式：

- 购买餐巾，每块餐巾需要p的代价
- 把旧餐巾送到快洗部，洗一块需要a天，洗一块的代价为b
- 把旧餐巾送到慢洗部，洗一块需要c天，洗一块的代价为d

在满足每一天的餐巾需求的情况下，最小化总代价。

$n \leq 1000$

每一天的餐巾需求相当于必须遍历某些点若干次。

每一天的餐巾需求相当于必须遍历某些点若干次。

每一天拆成2个点A_i和B_i，设这一天餐巾需求量为x。

- A_i和T连边$(x, 0)$，表示签到
- S和B_i连边$(x, 0)$，表示签到后剩下了x块餐巾可利用

接下来处理3种提供餐巾的方式：

接下来处理3种提供餐巾的方式：

- 购买餐巾：$S \to A_i$连边$(+\infty, p)$
- 快洗部：$B_i \to A_{i+a}$连边$(+\infty, b)$
- 慢洗部：$B_i \to A_{i+c}$连边$(+\infty, d)$

注意餐巾可以放到后面几天再洗，故还需$B_i \to B_{i+1}$连边$(+\infty, 0)$。

接下来处理3种提供餐巾的方式：

- 购买餐巾：$S \to A_i$连边$(+\infty, p)$
- 快洗部：$B_i \to A_{i+a}$连边$(+\infty, b)$
- 慢洗部：$B_i \to A_{i+c}$连边$(+\infty, d)$

注意餐巾可以放到后面几天再洗，故还需$B_i \to B_{i+1}$连边$(+\infty, 0)$。

最小费用最大流。

n个点m条边的有向图，每条边都由编号较小的点指向编号较大的点，经过每条边都有代价。

为到达第i个点，你可以在图上移动，也可以直接瞬移到那个点，后者需要代价a_i。

在遍历过每个点恰好1次的条件下，最小化总代价。

$n \leq 800$，$m \leq 15000$

每个点必选一次。

每个点必选一次。

直接应用模型，每个点拆成2个点A_i和B_i。

- A_i和T连边$(1,0)$，表示到这个点签到
- S和B_i连边$(1,0)$，表示签完到从这个点出发

每个点必选一次。

直接应用模型，每个点拆成2个点A_i和B_i。

- A_i和T连边$(1,0)$，表示到这个点签到
- S和B_i连边$(1,0)$，表示签完到从这个点出发

$S \to A_i$连边$(1,a_i)$表示瞬移。

对于一条边(x,y)，权值为z，$B_x \to A_y$连边$(1,z)$表示经过这条边。

最小费用最大流。

事实上这个做法的正确性在于原图是*DAG*。

若原图不是*DAG*，此题应该怎么做？

欢迎课后讨论!

$n+1$个点m条无向边，第i条边每经过一次就要付出代价L_i。

初始0号点有k个人，k个人可以分开行动。

只有访问过$i-1$号点，才能访问i号点。

在顺利访问n号点的前提下，最小化每个人的代价和。

$n \leq 150$，$m \leq 20000$，$k \leq 10$

首先发现一个结论：若想访问某个点，一定会走当前的最短路。

首先发现一个结论：若想访问某个点，一定会走当前的最短路。

于是我们第一个任务是：在若干点被封锁的情况下，计算出任意两个点的最短路。

首先发现一个结论：若想访问某个点，一定会走当前的最短路。

于是我们第一个任务是：在若干点被封锁的情况下，计算出任意两个点的最短路。

*Floyd*可以$O(n^3)$解决。

访问到n号点，实际上就是访问了所有点。

贪心地发现每个点应恰好访问1次。

访问到n号点，实际上就是访问了所有点。

贪心地发现每个点应恰好访问1次。

同样的模型，建图方式和前两题相同，$S \to B_0$连边$(k, 0)$即可保证k个人。

注意费用，由于某些点被封锁，应使用当前最短路。

这类问题通常是一类特殊的线性规划问题，它们具有区间的性质。

可以尝试使用费用流解决。

推导步骤：

- 添加辅助变量，将不等式转化为等式
- 将等式差分
- 对差分后的每个等式建点，相等关系视作流量平衡，变量视作流量转移

有n天，第i天需要至少a_i个志愿者。

志愿者有m种，第i种志愿者从第s_i天工作到第t_i天，每人花费代价为c_i。

在满足每天人数下限的条件下，最小化总代价。

$n \leq 1000$，$m \leq 10000$

设x_i =第i类志愿者的数量。

设x_i =第i类志愿者的数量。

对于第i天：$\sum x \geq a_i$

设x_i =第i类志愿者的数量。

对于第i天：$\sum x \geq a_i$

添加辅助变量$y(\geq 0)$，将不等式转化为等式：

P_i：$\sum x - y_i = a_i$

设P_0：$0 = 0$，P_{n+1}：$0 = 0$

将等式差分，得到的等式设为Q_1至Q_{n+1}。

设P_0：$0=0$，P_{n+1}：$0=0$

将等式差分，得到的等式设为Q_1至Q_{n+1}。

把当前每个等式看成一个点，相等关系看成流量平衡，尝试网络流建模。

设P_0：$0 = 0$，P_{n+1}：$0 = 0$

将等式差分，得到的等式设为Q_1至Q_{n+1}。

把当前每个等式看成一个点，相等关系看成流量平衡，尝试网络流建模。

常数部分表示这个点的盈余量，和S或T连边。

由于每个志愿者工作时间是连续区间，故变量x_i只会出现2次：在Q_{s_i}中为正，在Q_{t_i+1}中为负。

y_i也只出现2次：在Q_i中为正，在Q_{i+1}中为负。

由于每个志愿者工作时间是连续区间，故变量x_i只会出现2次：在Q_{s_i}中为正，在Q_{t_i+1}中为负。

y_i也只出现2次：在Q_i中为正，在Q_{i+1}中为负。

把变量看成流量在两点间转移的过程，每个变量建出一条边(a, b)，a表示这个变量的上界（本题为$+\infty$），b表示这个变量$+1$需要的代价。

由于每个志愿者工作时间是连续区间，故变量x_i只会出现2次：在Q_{s_i}中为正，在Q_{t_i+1}中为负。

y_i也只出现2次：在Q_i中为正，在Q_{i+1}中为负。

把变量看成流量在两点间转移的过程，每个变量建出一条边(a, b)，a表示这个变量的上界（本题为$+\infty$），b表示这个变量$+1$需要的代价。

最小费用最大流。

你在接下来的n个小时里每个小时都要做事情A或事情B，已知每个小时做每件事情的收益。

对于任意连续的k个小时，你做事情A的时间不能少于t_1小时，做事情B的时间不能少于t_2小时。

安排每个小时做的事情，最大化总收益，并输出一种方案。

$n \leq 1000$

由于限制条件是区间，我们可以尝试使用这个模型。

由于限制条件是区间，我们可以尝试使用这个模型。

设$x_i = 0$或1表示第i天做事情A或B。

添加辅助变量，列出等式，差分后和上题相似。

输出方案也很容易实现。

这类问题通常都存在较为直观的费用流做法，但其效率太低。

使用更高效的算法模拟费用流的过程。

一个n个数的数列，有m个操作，操作分为2种：

- 修改某个元素的值
- 查询在一个区间$[l, r]$中选择k个不相交的区间，其元素和的最大值

$n \leq 10^5$，$m \leq 10^5$，$k \leq 20$

首先考虑用线段树朴素实现：

线段树上每个点维护$O(k)$的信息，区间合并时复杂度为$O(k^2)$。

这样总复杂度为$O(mk^2logn)$，不能通过此题。

首先考虑用线段树朴素实现：

线段树上每个点维护$O(k)$的信息，区间合并时复杂度为$O(k^2)$。

这样总复杂度为$O(mk^2logn)$，不能通过此题。

建出一条$n+1$个点的链，链上每个点和S、T连边，这样数列中的区间和图中的增广路一一对应。

每次查询使用最大费用流增广k次。

时间复杂度$O(nmk)$，效率太低。

观察费用流过程的本质。

观察费用流过程的本质。

我们每次增广相当于贪心，本质上只有2种情况：

- 新增一个区间
- 从已选的某个区间中删除一段

观察费用流过程的本质。

我们每次增广相当于贪心，本质上只有2种情况：

- 新增一个区间
- 从已选的某个区间中删除一段

使用线段树实现这个贪心过程，支持区间取反、区间查询最大子段和。

每次查询就在线段树上查询、修改k次，最后把所有修改逐一复原即可。

时间复杂度为$O(mklogn)$，可以通过此题。

一棵n个点的树，第$i(i \geq 2)$个点和第$[\frac{i}{2}]$个点相连，所有边长度均为1。

有m只鼹鼠，已知每只鼹鼠的位置。

树上每个点都有不等量的食物。

你需要为前k只鼹鼠各分配一份食物，鼹鼠们会朝着食物移动，最小化所有鼹鼠移动距离之和。

对于所有的$1 \leq k \leq m$，分别求出答案。

$n \leq 10^5$，$m \leq 10^5$

暴力$O(nm^2)$的费用流非常显然。

暴力$O(nm^2)$的费用流非常显然。

把上一题的建模方法放到树上（树上路径和增广路一一对应），可得$O(nm)$的费用流做法。

暴力$O(nm^2)$的费用流非常显然。

把上一题的建模方法放到树上（树上路径和增广路一一对应），可得$O(nm)$的费用流做法。

同样考虑这个费用流每次增广的本质。

我们发现本质和上一题相似：

贪心地加入若干段路径并删除若干段路径，由于树高$O(logn)$，这些路径的段数也是$O(logn)$的。

我们发现本质和上一题相似：

贪心地加入若干段路径并删除若干段路径，由于树高$O(logn)$，这些路径的段数也是$O(logn)$的。

每次贪心地找出当前最优路径，然后一路暴力修改。

我们发现本质和上一题相似：

贪心地加入若干段路径并删除若干段路径，由于树高$O(logn)$，这些路径的段数也是$O(logn)$的。

每次贪心地找出当前最优路径，然后一路暴力修改。

维护出子树最优解，即可$O(logn)$寻找路径，$O(logn)$暴力修改。

时间复杂度$O(mlogn)$。

有n种蔬菜，第i种蔬菜初始有c_i个单位，每天会有x_i个单位变质，卖出每一单位的收益为a_i，且第一次卖出时还能额外获得收益s_i。

你每天可以卖出最多m个单位的蔬菜。

求销售k天的最大收益。

对于所有的$1 \leq k \leq p$，分别求出答案。

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

朴素的$O(nmp^2)$的拆点费用流较为容易。

和前两题相同，考虑每次增广的本质？

朴素的$O(nmp^2)$的拆点费用流较为容易。

和前两题相同，考虑每次增广的本质？

天数越靠后，增广路限制越多。尝试倒着考虑！

朴素的$O(nmp^2)$的拆点费用流较为容易。

和前两题相同，考虑每次增广的本质？

天数越靠后，增广路限制越多。尝试倒着考虑！

对于第k天的答案，倒着考虑每次卖出，贪心地发现卖出当前可以卖出的最大收益的蔬菜就是全局最优解。

朴素的$O(nmp^2)$的拆点费用流较为容易。

和前两题相同，考虑每次增广的本质？

天数越靠后，增广路限制越多。尝试倒着考虑！

对于第k天的答案，倒着考虑每次卖出，贪心地发现卖出当前可以卖出的最大收益的蔬菜就是全局最优解。

对每个k都朴素实现这个贪心，时间复杂度为$O(nmp^2)$。

考虑优化复杂度。

考虑优化复杂度。

求解第k天的答案时可用堆优化这个模拟。

时间复杂度变为$O(mp^2 logn)$。

瓶颈在于每次都要重新计算，我们再去分析这张图的性质。

对于第i天答案，设卖出的蔬菜集合为S_i。

一些性质：

- ①只考虑最大流，则一定可使得前面的天数满流，即蔬菜都堆到前面的天数卖出
- ②S_k必然是S_{k+1}的子集
- ③从S_{k+1}中删去最劣的若干元素可得S_k

对于第i天答案，设卖出的蔬菜集合为S_i。

一些性质：

- ①只考虑最大流，则一定可使得前面的天数满流，即蔬菜都堆到前面的天数卖出
- ②S_k必然是S_{k+1}的子集
- ③从S_{k+1}中删去最劣的若干元素可得S_k

再次倒着考虑，先$O(mplogn)$求出第p天的答案，再逐一推出前一天的答案。

根据性质①计算出当前这天的退流量，根据性质②③用堆维护答案即可，时间复杂度为$O(mplogn)$。

对于第i天答案，设卖出的蔬菜集合为S_i。

一些性质：

- ①只考虑最大流，则一定可使得前面的天数满流，即蔬菜都堆到前面的天数卖出
- ②S_k必然是S_{k+1}的子集
- ③从S_{k+1}中删去最劣的若干元素可得S_k

再次倒着考虑，先$O(mplogn)$求出第p天的答案，再逐一推出前一天的答案。

根据性质①计算出当前这天的退流量，根据性质②③用堆维护答案即可，时间复杂度为$O(mplogn)$。

总时间复杂度为$O(mplogn)$，可以通过此题。