# 图论

laofu

陈江伦 (清华大学交叉信息研究院)

July 16, 2021

# 图的存储方式

图的存储方式一般有两种

# 图的存储方式

图的存储方式一般有两种

**邻接矩阵** 邻接矩阵一般使用于在点数不是特别大的情况下使用主要功能是维护任意两个点之间的连通性、最短路径等信息但是邻接矩阵实用性并不强，因为它本身的空间消耗极大，而且不便于处理重边的情况

**邻接表** 邻接表是链表的一种，邻接矩阵的主要存储方式是存点，它的核心在于存边，链表的连接方式为由出点指向该点的第一条出边，对于其它所有的出边，记录他所对应的下一条出边，这样可以把所有需要维护的边的信息都存储起来，插入的复杂度为 o(1), 是图论问题最广泛的存储方式

# 强连通分量

# 强连通分量

在一张有向图中，强连通分量定义
为这张图的一个点集，满足这张点集任意两个点都可以互相到达。

# 强连通分量

在一张有向图中，强连通分量定义
为这张图的一个点集，满足这张点集任意两个点都可以互相到达。

用于求强连通的算法为 Tarjan

# tarjan 算法

对每个点记录

- dfn[k] 表示 k 的 dfs 序
- low[k] 表示 k 及 k 的后继能到达的点的 dfs 序的最小值

用一个栈依次存储访问过的节点,in[k] 表示这个节点是否在栈中

# tarjan 算法

对每个点记录

- dfn[k] 表示 k 的 dfs 序
- low[k] 表示 k 及 k 的后继能到达的点的 dfs 序的最小值

用一个栈依次存储访问过的节点,in[k] 表示这个节点是否在栈中

首先从所有没有访问过的节点 k 开始 dfs，访问到一个节点时，给他标上 dfs 序，并把 low 的初值设置为自己的 dfs 序，接着把当前元素放入栈中，用 in 数组打标记

# Tarjan 算法

接下来就访问当前元素的所有出边，对于每条出边指向的点 v 分为两种情况
第一种情况是 v 未访问过，那么在 dfs 树上 v 为 k 的孩子，那么直接 dfs(v)，并且在 dfs(v) 结束后用 low[v] 更新 lov[k]
第二种情况是 v 访问过，如果 v 在栈中，在 dfs 树上当前边为返祖边，那么用 v 的 dfn[1] 来更新 low[k]，如果 v 不在栈中，那么 k 和 v 不在同一颗 dfs 树上，直接忽略此边
在访问出边结束后，检查点 k 的 low 是否等于 dfn，如果相等说明 k 及 k 下方未出现返祖边，那么 k 与 k 的父亲不是强连通，此时持续弹栈一直弹出 k 为止，把弹出的元素编号同一个强连通分量，并把 in 标记取消。

[1]在有向图中取 low 和 dfn 都可以，但是在无向图中必须取 dfn，要严格按照 low 的定义:dfs 树上 k 的子树中能访问到的最早访问节点

# 双联通分量

双联通分量是无向图的联通分量，具体分为两类: 边双联通分量和点双联通分量。
边双联通分量: 任意两点间存在两条不重复边的路径
点双联通分量: 任意两点间存在两条不重复点的路径 (当前这两点除外)。
边双联通分量中每一条边至少在一个简单环中，点双联通分量中每两条边都至少共处于一个简单环。

# 双联通分量求法

边双联通分量的求法和有向图的强连通分量非常类似，但是注意因为在邻接表中我们采用的方法是把一条无向边转化为两条有向边，所以在访问时记录每条边是否走过 (不能记点，因为有重边)，强制一个点不能走反向边。
然后无向图不存在单向联通的情况，所以不需要用栈和 in 数组来判断。

# 双联通分量求法

边双联通分量的求法和有向图的强连通分量非常类似，但是注意因为在邻接表中我们采用的方法是把一条无向边转化为两条有向边，所以在访问时记录每条边是否走过 (不能记点，因为有重边)，强制一个点不能走反向边。
然后无向图不存在单向联通的情况，所以不需要用栈和 in 数组来判断。
桥: 两个双联通分量之间唯一的边

# 双联通分量求法

边双联通分量的求法和有向图的强连通分量非常类似，但是注意因为在邻接表中我们采用的方法是把一条无向边转化为两条有向边，所以在访问时记录每条边是否走过 (不能记点，因为有重边)，强制一个点不能走反向边。
然后无向图不存在单向联通的情况，所以不需要用栈和 in 数组来判断。
桥: 两个双联通分量之间唯一的边
双联通分量的算法也可以用来求桥，在访问一条边时如果该边指向的点的 low 值大于当前点的 dfs 序，即该边为桥

# 双联通分量求法

边双联通分量的求法和有向图的强连通分量非常类似，但是注意因为在邻接表中我们采用的方法是把一条无向边转化为两条有向边，所以在访问时记录每条边是否走过 (不能记点，因为有重边)，强制一个点不能走反向边。
然后无向图不存在单向联通的情况，所以不需要用栈和 in 数组来判断。
桥: 两个双联通分量之间唯一的边
双联通分量的算法也可以用来求桥，在访问一条边时如果该边指向的点的 low 值大于当前点的 dfs 序，即该边为桥
点双联通分量的判断方法与边双联通分量有很大的区别，对于一个点双联通分量，我们通过割点来找到这个点双联通分量。当访问某一个孩子，发现孩子的 low 值大于等于当前点时，说明这个孩子 v 即之后拓展的点不能回到 k 之前的点，那么 k 是一个割点，此时把栈中的元素持续弹出直到弹出 v 点为止，并标记它们在同一个点双联通分量中。

# 连通分量的应用

对于强联通分量的问题，我们采取的一般方法是缩点，并且缩点后整张图变成了一个 DAG。
对于边双联通分量的问题一般也是缩点，对于点双联通分量的问题我们一般是新建点，对于一个点双联通分量，我们删除原图中的所有边，并新建一个虚点，把这个点向点双联通分量内的所有点全部连边，这样最后构造出来的结构为一棵树，树型结构的维护比图要方便得多，并且这棵树还有一些很好的性质：两个点在树上的路径中经过的实点为它们在原图上路径的必经点。所以可以利用它统计两个点之间必经点的处理问题。

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件:

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件: 所有点的度数都为偶数

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件: 所有点的度数都为偶数
有向图欧拉回路判定条件:

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意: 任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件: 所有点的度数都为偶数
有向图欧拉回路判定条件: 所有点的入度 = 出度

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件: 所有点的度数都为偶数
有向图欧拉回路判定条件: 所有点的入度 = 出度
有向图的欧拉回路计数:

# 欧拉回路

定义: 在一张图中经过每条边恰好一次的路径
注意：任何欧拉回路问题都以连通性为前提
无向图欧拉回路判定条件: 所有点的度数都为偶数
有向图欧拉回路判定条件: 所有点的入度 = 出度
有向图的欧拉回路计数：在有向图中，以点 k 为起点的欧拉回路条数为以 k 为根的树形图个数 *k 的度数 *(每个点的入度-1)!
求树形图的个数可以用 matrix_tree 定理

# 无向图欧拉回路

从任意一个点开始 dfs。每次找一条没有访问过的边，然后先递归这条边对应的点，然后再加入这条边。
相当于是每次 dfs 出一个环，再从后往前输出方案。

# 有向图欧拉回路

方法类似，不过变成了有向边。我们把有向边反向之后问题就一样了。

注意：欧拉回路问题中一个点可能会被经过多次，那么每访问完一条边，就需要把这条边从邻接表删除，否则复杂度会达到平方。

# 混合图欧拉回路

给定一张图，又有有向边又有无向边。问是否存在一种把所有无向边都定向的方式使得形成的有向图存在欧拉回路。

# 混合图欧拉回路

给定一张图，又有有向边又有无向边。问是否存在一种把所有无向边都定向的方式使得形成的有向图存在欧拉回路。
有向图存在欧拉回路的条件是入度 = 出度。我们不妨先给每条无向边假设一个方向，然后修改一条边发生的变化是某个点入度-出度的值减小 2，另一个点入度-出度的值增加 2。
这就可以看成一个网络流问题，构建源汇 S，T，如果入度-出度 <0 则向 T 连边流量为 (出度-入度)/2，否则 S 向该点连边容量为 (入度-出度)/2。然后中间就连容量为 1 的边。

# 最小生成树

指图的一棵生成树，要满足所有点都联通且边权总和最小、

# Prim 算法

Prim 算法与 dijkstra 算法非常类似，随机选择一个初始点开始拓展，每次选择剩下的点集中距离当前点集最近的点进行拓展。基本不用。

# kruskal 算法

把所有边按照边权排序，从小到大加入边，只要当前边所连接的两个点未联通就加入，用并查集动态维护点和点的连通性。

# Borůvka 算法

每次让每个连通块连向与它相邻的, 边权最小的连通块。
这个算法最多进行 log 轮，因为每进行一轮，最小连通块的大小至少 *2。

# Problem

给你一棵 n 个点的树，点有点权 w，边有边权 v，现在有一张 n 个点的完全图，图中两个点 (a,b) 的边权是 w[a]+w[b]+a 和 b 在树上路径的所有边权之和。求完全图最小生成树。

# Problem

给你一棵 n 个点的树，点有点权 w，边有边权 v，现在有一张 n 个点的完全图，图中两个点 (a,b) 的边权是 w[a]+w[b]+a 和 b 在树上路径的所有边权之和。求完全图最小生成树。
因为是完全图，所以 prim 和 kruskal 的复杂度都是平方级别的。
考虑如果要求距离每个点的最近点，只需要线性的树形 DP 就可以了。
那么我们每一轮就 DP 距离每个点最近的不同连通块的点，总复杂度 nlogn。

# Problem

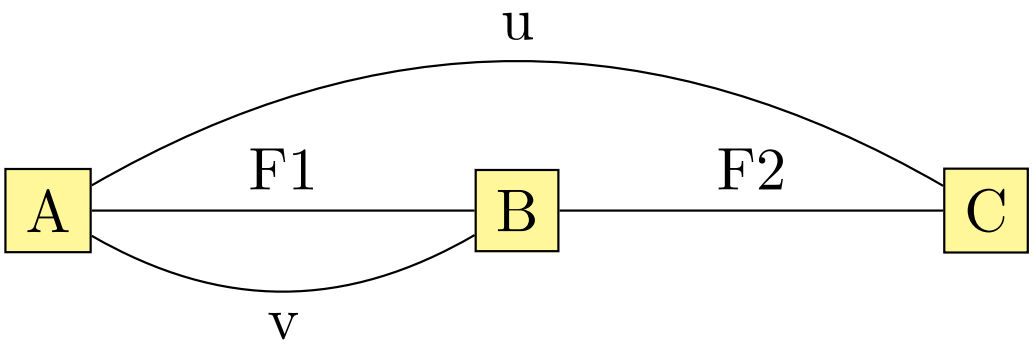
在图中给定一个点 k，求使得这个点度数恰好为 t 的最小生成树。

# Problem

在图中给定一个点 k，求使得这个点度数恰好为 t 的最小生成树。
做法：贪心 + 调整
我们首先删除点 k，求出最小生成树，然后再把点 k 加入继续求最小生成树。
如果这样求出来的生成树 k 度数超过 t 则无解。
然后我们需要继续加边使得 k 的度数变为 k。加入一条边时会形成一个环，我们需要在环上删除除了这条边之外的最大边。那么我们把加入边的权值-环上最大边的权值为关键字，每次取一个最大的加入，直到 k 度数恰好为 t 为止。

# 正确性

加入一条边后会使得其它边的关键字改变，假设有两条边 u,v 它们形成的环的最大边是同一条边。




如果 u 比 v 关键字小，那么会先删除 F1 加入 u，那么再加入 v 时就会删除 F2。反之，如果先删 F2，那么接下来就会删 F1。所以如果两条边都要加入最小生成树中，那么它们相互无影响。如果两条边只有一条边加入，那么选关键字最大的显然更优。

# 最短路算法

loyd 算法 枚举中转点，更细任意两个点之间的最短路

stra 算法 每次找一个未拓展过的离当前已拓展集合最近的节点，然后把这个点标记为已拓展，并更新到其它未拓展点的距离

Ford 算法 给每个节点标记一个活跃状态，每次用活跃节点更新其它节点

# 差分约束问题

差分约束系统是一类特殊的线性规划问题，基本模型为有若干个变量，并给出若干变量之间形如 $A \leq B + k$ 的限制，求在满足限制的前提下是否存在可行解或求某些变量的最值。
对于这个问题我们直接抽离图论模型，把一个变量看做一个点，如果要求 $y >= x+2$，那么把 x 向 y 连一条长度为 2 的边，然后对这张图跑一边最长路，如果图中存在正权环，那么表示不存在合法解，否则跑完一边最短路后每个点的距离就是它的最小值。

# 负权环问题

给定一张图，判断图中是否存在负权环

# 负权环问题

给定一张图，判断图中是否存在负权环
dfs 版的 spfa
我们把每个点的初始距离设置为 0，然后从每个点开始 dfs 更新其它点，每递归进入一个点时就把这个点标记为在递归栈中，在 dfs 时如果发现当前点可以更新某一个点，就直接递归那个点，如果图中有负权环时，会存在一个时刻，一个已经在递归栈中点被其它点更新，这时就可以立即返回出现负权环。如果不存在这种情况就无负权环。

# 同余最短路问题

有 $n$ 种面额的钞票，问能否组合出 $m$ 元钱

# 同余最短路问题

这一题朴素实现的复杂度都是指数级或 O(值域) 的。比较经典的方法是利用每个数可以选任意次，对于一个数 $x \in S$，如果能构造出数 $t$，那么 $t + kx(k \in N^*)$ 一定都能构造出来。
所以我们的方法是任意取一个 $x$，对于每一个 $r \in [0, x)$，求出最小的 $k$ 使得 $kx + r$ 能够被构造出来。令 $f[r] = kx + r$。那么判断能否构造出 $m$ 即判断 $m \geq f[m \bmod x]$。
这其实就类似于一个最短路问题，一开始仅有一个数 0，然后每一轮选择一个数，让它加上 S 中的每一个数，去更新其它状态。很显然我们取 x 为 S 中的最小值效率最高，那么它的复杂度就是 $|S|xlogx$ 的。

# 同余最短路问题

小性质：我们这里取 S 的最小值 x 作为状态量，状态之间互相转移，那么从 0 转移到任意一个状态的步数不超过 x，同时每一步的代价最大为 max(S)，所以，f 数组所有元素都不超过 min(S)*max(S)。这也说明了，能够组成的数是以 min(S)*max(S) 为周期循环的。(注：这里的循环是 $\rho$ 型，$0 \sim min(S) \times max(S) - 1$ 不在周期内)

# 矩阵树 (Matrix-Tree) 定理

令 $D(G)$ 为图 G 的度数矩阵，$A(G)$ 为图 G 的邻接矩阵，
$C(G) = D(G) - A(G)$
把 $C(G)$ 去掉任意一行任意一列后剩下的矩阵，行列式值即为生成树个数。

# Problem

求一张左边 $n$ 个点右边 $m$ 个点的完全二分图的生成树个数。

# Problem

求一张左边 $n$ 个点右边 $m$ 个点的完全二分图的生成树个数。
列出 $C(G)$ 去掉最后一行最后一列后的矩阵

$$\begin{pmatrix} m & & & -1 & -1 & -1 \\ & m & & -1 & -1 & -1 \\ & & m & -1 & -1 & -1 \\ -1 & -1 & -1 & n & & \\ -1 & -1 & -1 & & n & \\ -1 & -1 & -1 & & & n \end{pmatrix}$$

# Problem

把所有行全加到第一行

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ & m & & -1 & -1 & -1 \\ & & m & -1 & -1 & -1 \\ -1 & -1 & -1 & n & & \\ -1 & -1 & -1 & & n & \\ -1 & -1 & -1 & & & n \end{pmatrix}$$

消元之后

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ & m & & -1 & -1 & -1 \\ & & m & -1 & -1 & -1 \\ 0 & 0 & 0 & n & & \\ 0 & 0 & 0 & & n & \\ 0 & 0 & 0 & & & n \end{pmatrix}$$

答案即为 $n^{m-1} \times m^{n-1}$

# Prufer 序列

Prufer 序列与所有有标号无根树是一一对应的。

# Prufer 序列

Prufer 序列与所有有标号无根树是一一对应的。

构造方法：每次选择一个标号最小的叶子节点，把与它相邻的点加入序列，并删除这个点。

# Prufer 序列

Prufer 序列与所有有标号无根树是一一对应的。

构造方法：每次选择一个标号最小的叶子节点，把与它相邻的点加入序列，并删除这个点。
还原方法：每次把 Prufer 序列的第一个元素和可选集合中未出现在 Prufer 序列的最小的点
连边，然后从可选集合中删除这个最小点以及 Prufer 序列的首项。

# Prufer 序列

Prufer 序列与所有有标号无根树是一一对应的。

构造方法：每次选择一个标号最小的叶子节点，把与它相邻的点加入序列，并删除这个点。
还原方法：每次把 Prufer 序列的第一个元素和可选集合中未出现在 Prufer 序列的最小的点
连边，然后从可选集合中删除这个最小点以及 Prufer 序列的首项。

n 个点的完全图的生成树个数？

# Prufer 序列

Prufer 序列与所有有标号无根树是一一对应的。

构造方法：每次选择一个标号最小的叶子节点，把与它相邻的点加入序列，并删除这个点。
还原方法：每次把 Prufer 序列的第一个元素和可选集合中未出现在 Prufer 序列的最小的点
连边，然后从可选集合中删除这个最小点以及 Prufer 序列的首项。

n 个点的完全图的生成树个数？
$n^{n-2}$

在一张完全图中标记若干关键边，问包含这些关键边的生成树个数。

在一张完全图中标记若干关键边，问包含这些关键边的生成树个数。
一个点的度数 = 在 Prufer 中的出现次数 +1
如果 n 个点已经形成了 m 个连通块，每个连通块大小为 $a_i$，那么这 n 个点的生成树个数为

$$n^{m-2}\prod_{i=1}^{m}a_i$$

现在 Prufer 序列长度为 m-2，每个元素的取值范围是 $1\sim n$
然后每次把一个度数为 1 的连通块和它相邻点相连时可以选择 $a_i$ 个点中的任意一个。

# Problem

给定一张图，求它的最小生成树个数。

# Problem

给定一张图，求它的最小生成树个数。
性质: 对于一个给定的图，在最小生成树中对于不同权值的边它们是相互独立的，对于等权值的边，它们选择的数目一定，并且等权值的所有边所联通的点也是一定的。

# Problem

给定一张图，求它的最小生成树个数。
性质: 对于一个给定的图，在最小生成树中对于不同权值的边它们是相互独立的，对于等权值的边，它们选择的数目一定，并且等权值的所有边所联通的点也是一定的。
方法: 一般此类问题的等边权的边不会太多，通常利用 matrix-tree 定理求出每一种边权的边的方案数，然后利用乘法原理统计总方案数。

# 2-SAT 问题

2-sat 是一类布尔变量的取值问题，给定一些布尔变量的逻辑关系，求满足这些逻辑关系的一组方案。

建模：把一个布尔变量 k 拆成两个点 $k_0, k_1$，分别表示这个变量的取值为 False 和 True。一组方案中会选择 $k_0, k_1$ 中的恰好一个点。然后在图中进行连边，边 $(u, v)$ 的含义是如果选择了点 u 就必须选择点 v。

如果给定的逻辑关系能够用图上连边来表示，我们就可以用 2-sat 来求解。

Example:

要求 $a\&b = 0$: 连边 $a_1 \to b_0, b_1 \to a_0$

要求 $a|b = 1$: 连边 $a_0 \to b_1, b_0 \to a_1$

要求 $a$ 必须为 False: 连边 $a_1 \to a_0$

# 2-SAT 求解

## Tarjan 算法

如果图中存在一个环，则只要有一个点选了，则剩下的点都会选，只要有一个点不选，则其他点都不能选。
所以我们可以进行 tarjan 缩点，在同一个强连通分量中的点都是一样的状态。
如果此时存在 k 使得 $k_0, k_1$ 在同一个强连通分量，则这个点既不能选也不能不选，肯定无解。否则就是有解的。
复杂度为线性。

# 2-SAT 输出方案

每次找一个没有出度的点，把它作为选，然后把它相反的点作为不选。(在 2-sat 的图中，一条边 $a \to b$ 的含义为选了 a 就必须选 b，所以选择一个没有入度的点不会对其它点造成限制，而由于图是对称的，所以与没有出度的点相反的点一定没有入度，既然没有入度，那么不选这个点也不会对指向它的点造成限制)

小技巧: 可以不需要拓扑排序来寻找没有出度的点，由于在 dfs 树中没有出度的点是叶子节点，而 tarjan 算法一般会从下往上给每个强连通分量标号，所以我们按照标号从小往大扫描就一定是一个合法的拓扑序。

# Problem

求一组字典序最小的 2-SAT 解

# Problem

求一组字典序最小的 2-SAT 解
做法：爆搜
2-SAT 爆搜的复杂度是 O(nm) 的。我们每次搜索一个 k 是选择 $k_0$ 还是选择 $k_1$。假设先选择 $k_0$，然后我们把图中 $k_0$ 的后继全部标为选，把 $k_1$ 的后继全部标为不选。如果产生了冲突则改为选 $k_1$，否则就可以选择 $k_0$。
判断一次的复杂度是 O(m) 的，有 n 个变量，所以总复杂度为 O(nm)。

# 2-SAT 建模注意事项

需要注意的是，不是所有的图都满足 2-SAT 条件，2-SAT 的图必须为对称图，即如果存在边 $a_p \to b_q$，则必然存在边 $b_{1-q} \to a_{1-p}$ 解释一下就是如果选了 a 就必须选 b，则没选 b 就不可能选 a。两个命题互为逆否命题。如果图满足这个条件，则它就无法用逻辑意义描述。

# 定义

二分图指的是能把图划分为两个点集，满足任何一个点集内部没有边。

一些二分图结论：

最小点覆盖 (选择最少的点使得所有边的两端点至少一个被选择)= 最大匹配

最大独立集 (选择最多的点使得它们两两没有边直接相连)= 点数-最大匹配

最小边覆盖 (选择最少的边取覆盖所有的顶点)= 点数-最大匹配

# 二分图判定

判定方法: 一个图为二分图当前仅当它不存在长度为奇数的环。
实现方法 1: 用并查集维护一个点的颜色 (黑、白)，每次用一条边合并两个集合，如果这条边联通的两个点颜色相同就给其中一个打上标记，表示颜色全部反转。当某一条边加入之前两个点已经联通时，就检查它们颜色是否相同，如果相同就存在奇环。

# NOIP2010 关押罪犯

S 城现有两座监狱，一共关押着 N 名罪犯，他们之间的关系自然也极不和谐，很多罪犯之间甚至积怨已久，如果客观条件具备则随时可能爆发冲突。

我们用“怨气值”来表示某两名罪犯之间的仇恨程度，怨气值越大，则这两名罪犯之间的积怨越多。如果两名怨气值为 c 的罪犯被关押在同一监狱，他们俩之间会发生摩擦，并造成影响力为 c 的冲突事件。

现在警察局需要把罪犯进行合理分配，使得影响力最大的冲突事件的影响力最小。

# 二分图判定

用并查集维护当前强制在一个集合的元素，用数组 enemy[x] 表示与 x 有边的任意一个元素
每次要合并两个元素 x,y 时，先检查两个点是否已经有出边，如果 y 有出边那么就把 y 的出边指向的点与 x 合并，如果 x 有出边就把 x 的出边指向的点与 y 合并。当在某一时刻加入边的两个端点已经处于同一集合则不是二分图。

# 二分图染色

因为二分图两个点集的内部无边，故可以对二分图进行黑白染色，让所有边的两个端点都染上不同的颜色

方法：枚举所有的未染色点，然后随机给它染色，接着遍历所有与它相邻的未染色的点，给它们染上相反色，并持续递归直到当前连通块内所有点全部染色完毕。

# 二分图匹配

匈牙利算法的核心思想在于找增广路，它是一种贪心 + 调整的思路。

算法过程：从所有没有匹配的节点 S 开始寻找增广路，首先访问它的所有出边，如果发现有某个点 T 还未加入匹配就打通 S 到 T 的增广路，在这条增广路上一共有 2n 个点和 2n-1 条边，其中 n-1 条边为已匹配边，打通增广路的过程就是把增广路上所有边的状态全部取反，把 n-1 条边从当前匹配中删除，并加入其它的 n 条边。

匈牙利有两种实现的方法:dfs 和 bfs，但是 dfs 版的匈牙利算法的效率在绝大部分情况下都劣于 bfs，所以不建议用 dfs 实现。

bfs 版就是把在交错树中的所有末端点方在队列中拓展，并沿途记录每个点的前驱，只要发现增广路就沿前驱指针和匹配边跳回到 S 并把边取反。

# 二分图最大权匹配

## KM 算法

km 算法运用了顶标模型，对于图中的每一个点，我们都给它一个顶标，左集合的顶标为 lx[x]，右集合的顶标为 ly[y]，一组合法顶标必须满足对于任何一条边 (x,y) 都有 w(x,y)<=lx[x]+ly[y]，w(x,y) 为 x,y 两点间边的边权。
顶标模型满足对于任意一组匹配，它们的匹配权值之和要小于等于它们联通的点的顶标之和。
在顶标模型中还有一个概念叫做相等子图，即只保留 w(x,y)=lx[x]+ly[y] 的边，把其它的边全部从图中删除。那么如果在相等子图中有完备匹配则一定是该二分图的最大权匹配。(所有匹配的权值都小于当前顶标和，那么只要当前顶标和能达到就是最大权匹配)

# 二分图最大权匹配

## KM 算法过程

km 算法的基本过程还是调整思想，在保证顶标合法的情况下调整顶标使得更多的边加入匹配中。首先可以构造出一组初始解 lx[x]=max(w(x,y))，ry[y]=0，然后枚举每个未进入最大匹配中的点开始寻找增广路，对于当前点 x 的每一个相连点 y, 如果当前边 (x,y) 在相等子图中，那么我们像匈牙利算法一样去寻找增广路。

进行一次增广后如果发现交错树中不存在增广路，那么我们要调整顶标，首先找到所有边中最容易进入匹配的边 (即 ly[y]+lx[x]-w(x,y) 的最小值)，然后把左集合在交错树中的点全部加上这个最小值，把右集合在交错树中的点全部减去这个最小值。这样操作之后可以发现，原来已经匹配的边仍然在匹配中，而调整后有集合有一部分点因此而加入交错树，匹配数时单调不降的，所以一定能在若干次调整后找到完备匹配。

# 定义

在图论中，网络流（英语：Network flow）是指在一个每条边都有容量（capacity）的有向图分配流，使一条边的流量不会超过它的容量。
通常在运筹学中，有向图称为网络。顶点称为节点（node）而边称为弧（arc）。
一道流必须符合一个结点的进出的流量相同的限制，除非这是一个源点（source） 有较多向外的流，或是一个汇点（sink） 有较多向内的流。
一个网络可以用来模拟道路系统的交通量、管中的液体、电路中的电流或类似一些东西在一个结点的网络中游动的任何事物。

# 符号定义

1. $V$ 表示整个图中的所有结点的集合。
2. $E$ 表示整个图中所有边的集合。
3. $G = (V, E)$，表示整个图。
4. $S$ 表示网络的源点，$T$ 表示网络的汇点。
5. 对于每条边 $(u, v)$，有一个容量 $C(u, v)(C(u, v) \geq 0)$
6. 如果 $C(u, v) = 0$，则表示 $(u, v)$ 不存在在网络中。
7. 如果原网络中不存在边 $(u, v)$，则令 $C(u, v) = 0$。
8. 对于每条边 $(u, v)$，有一个流量 $f(u, v)$。

# 储存方式

一个网络流一般用邻接矩阵储存，这样就能非常方便地读取任意两点之间的容量与流量。在增广路算法中，向残量网络中加入方向变也十分容易。

对于每条边 $(u, v)$, 有一个容量 $C(u, v)(C(u, v) \geq 0)$。

对于每条边 $(u, v)$, 有一个流量 $f(u, v)$。

这样的储存方式可以帮助程序变得简短精炼。

# 容量限制

对于所有的节点 $u, v \in V$，要求 $0 \leq f(u, v) \leq c(u, v)$。
简而言之，就是任何一条弧的流量不能超过其最大容量。
可以想象成一个水管中流过的水不能超过其容积（不然水管会炸）。
这个特殊的性质使网络流能与许多实际问题联系在一起，如：交通疏通、管道运输。

# 流量守恒

对于所有的结点 $u \in V - \{S, T\}$，要求 $\sum_v f(v, u) = \sum_v f(u, v)$
即流入流量与流出流量相等，每个结点既不能存流量也不能超支流量。
就好像一个快递员，既不能把别人给他的快件据为己有，也不能凭空拿出东西来给别人。
这样就保证所有的流量都从源点流到了汇点。

# 最大流问题

定义一个网络的流量为（用 $|f|$ 表示）：

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

即从源点流出的流量的和。
最大流问题就是在满足网络流性质的情况下求出最大的 $|f|$。

# 最大流算法

- Ford-Fulkerson
- Edmonds-Karp
- ★ Dinic
- 预流推进算法

# 割

将拥有源点 $S$ 和汇点 $T$ 的结点集合 $V$ 分成两个子集 $s$ 和 $t$，当且仅当 $S$ 属于 $s$ 且 $T$ 属于 $t$ 时，称这种分割为 $S-T$ 割。

# 割

将拥有源点 $S$ 和汇点 $T$ 的结点集合 $V$ 分成两个子集 $s$ 和 $t$，当且仅当 $S$ 属于 $s$ 且 $T$ 属于 $t$ 时，称这种分割为 $S-T$ 割。$s-t$ 的容量 $C[s,t]$ 指的是所有从 $s$ 到 $t$ 的边的容量之和。具体公式如下：

$$C[s,t] = \sum_{(u,v)\in E, u\in s, v\in t} C[u][v]$$

最小割，就是所有可能的 $S-T$ 割中容量最小的 $S-T$ 割。

# 最大流最小割定理

在一个流网络中，最小割 = 最大流。

# 最小割输出方案

用最大流最小割仅仅能做到求出最小割的具体数值，但是如果题目要求的是求出最小割的方案，那么还需要进一步的操作。
我们可以利用 dinic 算法的分层图，如果一条边 (u,v) 已经满载，并且 u 和 S 联通,v 和 S 不连通，则把它输出，最后输出的所有边就是一个可行的最小割的方案，这个问题的拓展还有，求哪些边一定在最小割中 (把这条边退流，删除这条边后重新寻找的增广路流量与这条边原来的流量相同)，求哪些边可能在最小割中 (对某条边退流后重新找增广路可以使得这条边满载且 u 和 S 联通，v 不和 S 联通)。

# 最小费用最大流问题

最小费用最大流问题是指，在最大流问题的基础上，给每条边增加一个费用。一个网络的总费用为每条边的费用 × 这条边的流量。
目标是在保证最大流的情况下，使得网络的总费用最少。

# 费用流算法

- Edmonds-Karp
- ZKW 费用流

# 二分图匹配问题

## 可能在二分图最大匹配中的点

所有有度数的点都是可能在最大匹配中的点。

## 一定在二分图最大匹配中的点

先求任意一个最大匹配 S，那么一定在二分图最大匹配中的点一定是 S 的子集。
源点能到达的左侧的点，和能到达汇点的右侧的点就都不是一定在最大匹配中。
剩下的在 S 中的点就一定在最大匹配中。

# 二分图匹配问题

## 可能在二分图最大匹配中的边

先求任意一个最大匹配 S。S 中的边都为可能在二分图最大匹配中的边。
然后考虑其它边，如果一条未匹配边的两个端点中有一个不再 S 中，则这条边可能在最大匹配中，否则，如果想把这条边加入匹配，则需要一个交错环。那么我们把所有边定向，匹配边从右往左，非匹配边从左往右，如果一条边两个端点在同一个 SCC 中则也可能在最大匹配中。

# 二分图匹配问题

## 一定在二分图最大匹配中的边

先求任意一个最大匹配 S。那么 S 中的边中不能被非 S 的边替代的边就是一定在最大匹配中的边。所以方法也是类似的，从所有未盖点出发，能够访问到的边就都不是一定在最大匹配中的边。(如果是从左侧的点出发，那么从左到右走非匹配边，从右到左走匹配边)
然后再缩点，如果 S 中某条边两段在同一个 SCC 中也不一定在最大匹配中。

# 可能在最小割中的边

# 可能在最小割中的边

首先把所有还有流量的边拿出来跑 tarjan 缩强连通分量。如果某条满流边的两个端点不在同一个 SCC 中则可能在最小割中。

**Proof.**

减少一条边的容量，最小割也随之减少说明这条边可能在最小割中。
采用反证，假设 (u,v) 在同一个强连通分量中，(u,v) 的容量将减少 d，那么我们找到这个环，把环上所有边的流量全部减去 d，仍然满足流量平衡，而此时最大流不变。反之最大流一定会改变，所有如果 (u,v) 不在同一个强连通分量中则这条边可能在最小割中。 □

# 一定在最小割中的边

首先把所有还有流量的边拿出来跑 tarjan 缩强连通分量。如果某条满流边的两个端点 (u,v) 满足 s 和 u 在同一个强连通分量，v 和 t 在同一个强连通分量则这条边一定在最小割中。

**Proof.**

增加一条边的容量，最小割也随之增加说明这条边一定在最小割中。
因为 (u,v) 满流，所以必然存在 s 到 u 的流和 v 到 t 的流，那么沿反向边 u 可以到达 s，t 可以到达 v。如果 s,u 在同一个强连通分量，v,t 在同一个强连通分量，则说明 s 到 u 还有增广路，v 到 t 还有增广路，那么如果我们增加 (u,v) 边的容量，则最大流必然会增加。 □

# 最小路径覆盖问题

# 最小路径覆盖问题

给定一张图，要求选出尽量少的简单路径覆盖所有的点。

# 最小路径覆盖问题

给定一张图，要求选出尽量少的简单路径覆盖所有的点。
直接求解看似很难，我们不妨换一个理解方式：一条简单路径必然包含的点数 = 包含的边数 +1。包含的点数总和恒为 $n$, 那么 $n-$ 最多包含的边数就是答案。
现在问题变成了要选出尽量多的边，同时为了保证方案的合法性，每个点最多有一条出边被选择，最多有一条入边被选择。这实际上就是一个二分图匹配问题，每个点最多只能与一个点匹配。
所以直接用匈牙利/网络流即可。

# 环覆盖问题

# 环覆盖问题

一张 $n$ 个点 $m$ 条边的有向图，你要从中选出若干个点不相交的环。记这些环覆盖的点集为 $S$。
求一种方案使得 $S$ 中的点从大到小排序后字典序最大。

# 环覆盖问题

一张 $n$ 个点 $m$ 条边的有向图，你要从中选出若干个点不相交的环。记这些环覆盖的点集为 $S$。
求一种方案使得 $S$ 中的点从大到小排序后字典序最大。
一样的方法，我们给每个点都加一个自环，这样选出若干个不相交的环 $\Leftrightarrow$ 每个点入度出度都为 1。
然后贪心地从大往小扫每个点，能不选自环就不选自环。

# 最大权闭合图

# 最大权闭合图

问题：有若干个物品，每个物品有一个权值，可正可负。现在有若干限制，每个限制为一个二元组 $(a, b)$，表示如果选了 $a$ 就一定要选 $b$。问选出一个合法物品集合的最大权值和。

# 最大权闭合图

问题：有若干个物品，每个物品有一个权值，可正可负。现在有若干限制，每个限制为一个二元组 $(a, b)$，表示如果选了 $a$ 就一定要选 $b$。问选出一个合法物品集合的最大权值和。

首先如果没有这些限制，我们的方案肯定是选择所有正权物品，丢弃所有负权物品。

但是这个时候会出现一种矛盾的情况：如果一个限制 $(a, b)$，其中 $a$ 是正权而 $b$ 是负权则不合法。

那么我们可以用最小割来描述这种不合法的情况。从 $S$ 到 $a$ 连流量为 $val[a]$ 的边，从 $b$ 到 $T$ 连流量为 $val[b]$ 的边，对于一个限制 $(a, b)$ 我们从 $a$ 到 $b$ 连流量为 $\infty$ 的边。

这样，如果割掉 $S$ 到 $a$ 的边则代表放弃 $a$，如果割掉 $b$ 到 $T$ 的边则代表选择 $b$，这样就保证了解的合法性。那么正权和-这张图的最小割就是问题的解。

# 最大密度子图

# 最大密度子图

一张图的密度为图上的边数除以点数。现在要求一张图的最大密度子图。

# 最大密度子图

一张图的密度为图上的边数除以点数。现在要求一张图的最大密度子图。
计算答案的式子是两个量相除的形式，很显然可以用分数规划把两个量变成另一个量。
二分答案 mid，问题变成了判断是否存在一个子图
边数 $-$ 点数 $*\ mid > 0$(注意，这里必须是严格大于零，否则选择空图就一定满足)。也就是说，选择一条边可以增加权值 1，选择一个点会损失权值 1。
不考虑合法性方案一定是选择所有的边，放弃所有的点。然后考虑如何让方案合法。选择了一条边就必须选择它两端的点，那么 $S$ 到每条边流量为 1，每个点到 $T$ 流量为 $mid$。每条边向它连接的两个点流量为 $\infty$。
这样要么把两个点全选，要么放弃这条边。直接判断边数 $-$ 这张图的最小割是否 $> 0$ 即可。

# 二元关系最小割模型

对于这样一类问题：有若干个变量，每个变量有两种取值。有若干个限制，每个限制形如：如果变量 $x$ 取值为 $a$，变量 $y$ 取值为 $b$ 则需要额外付出 $c$ 的代价。然后要最大化所有变量的值之和减去额外代价。

这类问题的一个经典做法就是构建最小割模型。每个变量建一个点，$S$ 向每个点连边、每个点向 $T$ 连边表示两种取值中要选择其中一个。然后对于一个限制就在两个变量之间连边。

# 二元关系最小割模型

## NOI2006 最大获利

# 二元关系最小割模型

## NOI2006 最大获利

有 $n$ 个站点和 $m$ 个用户群。每个站点建立有一个成本。每个用户群会指定一个站点集合。如果一个用户群使用的站点集合都已经建立则可以获得一定收益。要求最大化收益-成本。

# 二元关系最小割模型

## NOI2006 最大获利

有 $n$ 个站点和 $m$ 个用户群。每个站点建立有一个成本。每个用户群会指定一个站点集合。如果一个用户群使用的站点集合都已经建立则可以获得一定收益。要求最大化收益-成本。

对于一个站点有两种取值，分别代表建立和不建立，一个用户群也有两种取值，分别代表能获利和不能获利。

$S$ 到每个站点的边，如果割掉表示建立这个站点。每个用户群到 $T$ 的边，如果割掉表示这个用户群不能获利。

然后站点向用户群连边，容量为 $\infty$，表示要么建立站点，要么不获利。

# 二元关系最小割模型

文理分科

# 二元关系最小割模型

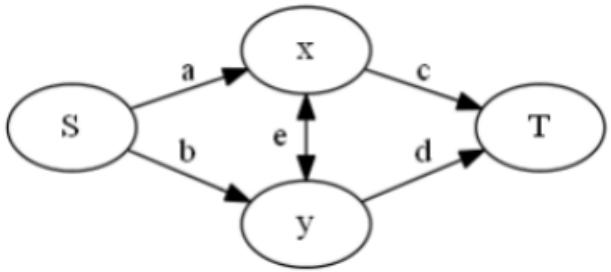
## 文理分科

一个班级的座位表是一个 $n * m$ 的矩阵。现在要进行文理分科。
同学 $i$ 选择理科的喜悦值是 $p_i$，选择文科的喜悦值是 $q_i$。
对于矩阵中相邻两个同学，如果同时选择理科，他们将共同获得喜悦值 $v_{i1}$，如果同时选择文科将共同获得喜悦值 $v_{i2}$。
要求最大化所有同学喜悦度之和。

# 二元关系最小割模型

## 文理分科

一个班级的座位表是一个 $n * m$ 的矩阵。现在要进行文理分科。
同学 $i$ 选择理科的喜悦值是 $p_i$，选择文科的喜悦值是 $q_i$。
对于矩阵中相邻两个同学，如果同时选择理科，他们将共同获得喜悦值 $v_{i1}$，如果同时选择文科将共同获得喜悦值 $v_{i2}$。
要求最大化所有同学喜悦度之和。
对每个同学建立一个点，与 $S$ 和 $T$ 的边分别表示选文科和选理科。




对于一个同学，如果保留和 S 之间的边则表示选理科，如果保留和 T 之间的边表示选文科。$v_1$ 表示同时选理科的收益，$v_2$ 表示同时选文科的收益。那么，可以列出方程：

# 二元关系最小割模型

## 文理分科

$$
\begin{aligned}
a+b &= v_1+p_1+p_2 \\
c+d &= v_2+q_1+q_2 \\
a+d+e &= v_1+v_2+p_1+q_2 \\
b+c+e &= v_1+v_2+q_1+p_2
\end{aligned}
$$

然后，后两个式子相加减去前两个式子得到

$$2e = v_1 + v_2$$

则

$$
\begin{aligned}
a &= \frac{v_1}{2}+p_1 \\
b &= \frac{v_1}{2}+q_1 \\
c &= \frac{v_2}{2}+p_2 \\
d &= \frac{v_2}{2}+q_2
\end{aligned}
$$

# 距离限制模型

## 切糕

# 距离限制模型

## 切糕

给出一个 $P \times Q$ 的网格，现需要在每一格选择一个 $[1, R]$ 之间的高度，且在第 $i$ 行第 $j$ 列选择高度 $h$ 的代价为 $v(i, j, h)$。
要求任意四连通相邻的格子选择的高度差的绝对值不超过 $D$，求最小的总代价。

# 距离限制模型

## 切糕

给出一个 $P \times Q$ 的网格，现需要在每一格选择一个 $[1, R]$ 之间的高度，且在第 $i$ 行第 $j$ 列选择高度 $h$ 的代价为 $v(i, j, h)$。
要求任意四连通相邻的格子选择的高度差的绝对值不超过 $D$，求最小的总代价。
当没有高度限制时，将每一格的每个高度看作一个节点，同一格的相邻高度代表的节点之间连边，再建立虚拟源点和汇点，将指向一个节点的边的流量设为其相应的代价，则网络最小割即为所求最小总代价。

# 距离限制模型

## 切糕

如何限制相邻高度差不超过 $D$?

这是最小割模型中一类常见的模型，通常形式为每个点有若干种取值，且对于相邻的两点 $i, j$，要求其取值 $x_i, x_j$ 满足 $x_i - x_j \leq d$，求一种方案使得所有取值对应的代价之和最小。

对于此类模型，可以直接在无限制的网络上进行修改。在最小割模型中，限制条件一般用无穷大的边来表示。

考虑原题中相邻的两格 $(i, j), (i', j')$，对于所有 $D < k \leq R$，在它们对应的节点间增加一条边 $(i, j, k) \to (i', j', k - D)$，容量为无穷大。

显然新边不会成为割边，且增加这样的边后，网络中的割边就自然满足距离限制了，故新网络的最小割即为原题答案。

这就是距离限制问题的一般解法。

# 匹配角度

## 环覆盖

# 匹配角度

## 环覆盖

一张 $n$ 个点 $m$ 条边的图，有 $m$ 条边，每条边有一个代价。
要选择若干个点不相交的环覆盖所有点，最小化环上所有边权之和。

# 匹配角度

## 环覆盖

一张 $n$ 个点 $m$ 条边的图，有 $m$ 条边，每条边有一个代价。
要选择若干个点不相交的环覆盖所有点，最小化环上所有边权之和。
之前提到过的模型，等价于每个点入度和出度均为 1。

# 不等关系建立网络流模型

## NOI2008 志愿者招募

# 不等关系建立网络流模型

## NOI2008 志愿者招募

有 $n$ 天活动，每一天需要至少 $A[i]$ 个志愿者，有 $m$ 类志愿者，招募 1 个第 $i$ 类志愿者，可以在第 $S[i]$ 到 $T[i]$ 天工作，每招募一个花费代价 $C[i]$。求最小代价。

# 不等关系建立网络流模型

## NOI2008 志愿者招募

有 $n$ 天活动，每一天需要至少 $A[i]$ 个志愿者，有 $m$ 类志愿者，招募 1 个第 $i$ 类志愿者，可以在第 $S[i]$ 到 $T[i]$ 天工作，每招募一个花费代价 $C[i]$。求最小代价。

设第 $i$ 类志愿者招募 $x[i]$ 人，第 $j$ 天一共招募了 $P[j]$ 人。那么我们可以列出不等式组：

$$\forall i \in [1,n] \quad P[i] = \sum_{S[j] \le i \le T[j]} \ge A[i]$$

例如，一共四天，需要的人数分别为 $4,2,5,3$，有 5 类志愿者。

| 种类 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 时间 | 1-2 | 1-1 | 2-3 | 3-3 | 3-4 |
| 费用 | 3 | 4 | 3 | 5 | 6 |

# 不等关系建立网络流模型

## NOI2008 志愿者招募

那么我们列出的不等式组就是

$$\begin{cases} P[1]= x[1]+x[2] \geq 4 \\ P[2]=x[1]+x[2]+x[3] \geq 2 \\ P[3]=x[3]+x[4]+x[5] \geq 5 \\ P[4]= x[5] \geq 3 \end{cases}$$

利用线性规划的松弛型的方法，我们可以把不等式用变量替换为等式。设 $y[i]$ 表示第 $i$ 天多招募 $y[i]$ 人。

$$\forall i \in [1,n] \quad P[i]-y[i]=\sum_{S[j] \leq i \leq T[j]} x[j]-y[j]=A[i]$$

那么我们把相邻两个不等式差分一下，注意到一个变量在两个式子中全部出现了，一次正一次负。那么我们可以把一个等式看做一个点，如果等式内的变量为正号则代表流量流入，为负号则代

# 不等关系建立网络流模型

## ZJOI2013 防守战线

# 不等关系建立网络流模型

## ZJOI2013 防守战线

一个长度为 n 的序列，在序列第 i 号位置上建一座塔有 C[i] 的花费。M 个约束，为在 [L[i],R[i]] 范围内要建至少 Di 座塔。求最少花费。

# 不等关系建立网络流模型

## ZJOI2013 防守战线

一个长度为 n 的序列，在序列第 i 号位置上建一座塔有 C[i] 的花费。M 个约束，为在 [L[i],R[i]] 范围内要建至少 Di 座塔。求最少花费。

先把线性规划模型建立出来，我们发现，这个问题并不满足线性规划矩阵每一列的 1 连续，但是它有一个特殊的性质：每一行的 1 是连续的。所以我们可以把线性规划对偶一下。这样就使得每一列的 1 连续，然后再用不等关系建立网络流模型即可。

# 不等关系建立网络流模型

## Problem

# 不等关系建立网络流模型

## Problem

一个长度为 $n$ 的区间，每个位置有两个属性 $A[i]$ 和 $B[i]$，需要在每个位置选择一个属性，使得对于任意一段长度为 $k$ 的区间满足区间内 $A[i]$ 的个数不小于 $P$，$B[i]$ 的个数不小于 $Q$。

# 不等关系建立网络流模型

## Problem

一个长度为 $n$ 的区间，每个位置有两个属性 $A[i]$ 和 $B[i]$，需要在每个位置选择一个属性，使得对于任意一段长度为 $k$ 的区间满足区间内 $A[i]$ 的个数不小于 $P$，$B[i]$ 的个数不小于 $Q$。
一开始先默认所有元素都选择 $A[i]$，接下来我们需要调整一些元素使得既满足条件又费用最大。首先不小于的问题可以变成不大于的问题，$A[i]$ 的个数不小于 $P$ 即 $B$ 的个数不大于 $k-p$。
用 $x[j]$ 表示 $j$ 是否改变属性。那么我们可以列出不等式组

$$\forall i \in [1, n-k+1] L \leq \sum_{j=i}^{i+k-1} x[j] \leq R$$

设 $L + b[i] = \sum_{j=i}^{i+k-1} x[j] = R - a[i]$，则

$$
\begin{aligned}
0 &= 0 \\
x[1] + x[2] + x[3] + \ldots + x[k] + a[1] &= R \\
x[1] + x[2] + x[3] + \ldots + x[k] - b[1] &= L \\
x[2] + x[3] + x[4] + \ldots + x[k+1] + a[2] &= R \\
x[2] + x[3] + x[4] + \ldots + x[k+1] - b[2] &= L \\
\ldots &= \ldots \\
x[n-k+1] + x[n-k+2] + \ldots + x[n] + a[n-k+1] &= R \\
x[n-k+1] + x[n-k+2] + \ldots + x[n] - b[n-k+1] &= L \\
0 &= 0
\end{aligned}
$$

差分之后，每个变量出现两次且一正一负。