# 图论和网络流

King_George

2019.3.

# 三元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个三元组 $(u, v, w)$ 满足两两之间有边相连。

# 三元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个三元组 $(u, v, w)$ 满足两两之间有边相连。
将所有点按度数从小到大排序，如果度数相同按点编号排序，$u$ 的排名记作 $rnk_u$。

# 三元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个三元组 $(u, v, w)$ 满足两两之间有边相连。
将所有点按度数从小到大排序，如果度数相同按点编号排序，$u$ 的排名记作 $rnk_u$。
枚举三元组中 $rnk$ 较小的两个点 $u, v$，再枚举与 $u$ 相连的 $w$ $(rnk_w > rnk_u, rnk_w > rnk_v)$。

# 三元环计数

## 复杂度分析

如果 $u$ 的度数 $>$ $\sqrt{m}$ 则称 $u$ 为大点，否则为小点。

# 三元环计数

## 复杂度分析

如果 $u$ 的度数 $> \sqrt{m}$ 则称 $u$ 为大点，否则为小点。

1. 如果 $u$，$v$ 都是小点，则 $u$ 的邻居不超过 $\sqrt{m}$ 个。

# 三元环计数

## 复杂度分析

如果 $u$ 的度数 $>\sqrt{m}$ 则称 $u$ 为大点，否则为小点。

1. 如果 $u$，$v$ 都是小点，则 $u$ 的邻居不超过 $\sqrt{m}$ 个。
2. 如果 $u$，$v$ 至少一个是大点，则满足 $rnk_w > rnk_u$，$rnk_w > rnk_v$ 的 $w$ 不超过 $\sqrt{m}$ 个。

# 三元环计数

## 复杂度分析

如果 $u$ 的度数 $>\sqrt{m}$ 则称 $u$ 为大点，否则为小点。

1. 如果 $u$, $v$ 都是小点，则 $u$ 的邻居不超过 $\sqrt{m}$ 个。
2. 如果 $u$, $v$ 至少一个是大点，则满足 $rnk_w > rnk_u$, $rnk_w > rnk_v$ 的 $w$ 不超过 $\sqrt{m}$ 个。

所以总复杂度 $O(m\sqrt{m})$。

# 四元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个四元组 $(u,\ v,\ w,\ x)$ 满足 $(u,\ v),\ (v,\ w),\ (w,\ x),\ (x,\ u)$ 有边相连。

# 四元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个四元组 $(u, v, w, x)$ 满足 $(u, v), (v, w), (w, x), (x, u)$ 有边相连。
将所有点按度数从小到大排序，如果度数相同按点编号排序，$u$ 的排名记作 $rnk_u$。

# 四元环计数

给定一个 $n$ 个点 $m$ 条边无向图，问有多少个四元组 $(u,\ v,\ w,\ x)$ 满足 $(u,\ v),\ (v,\ w),\ (w,\ x),\ (x,\ u)$ 有边相连。
将所有点按度数从小到大排序，如果度数相同按点编号排序，$u$ 的排名记作 $rnk_u$。
对于每个点 $u$，枚举两条边 $(u,\ v),\ (v,\ w)$，满足 $rnk_w > rnk_u,\ rnk_w > rnk_v$，每访问到一个 $w$ 给答案加上 $w$ 的标记，并给 $w$ 的标记加 1。

# 四元环计数

## 复杂度分析

与三元环计数类似。

# 四元环计数

代码大概长这样：

```
for (int u = 0; u < n; ++u) {
    for (int v: g[u]) {
        for (int i = g[v].size() - 1; ~i; --i) {
            int w = g[v][i];
            if(w <= u || w <= v) break;
            ret = (ret + tmp[w]) % mod;
            ++tmp[w];
        }
    }
    for (int v: g[u]) {
        for (int i = g[v].size() - 1; ~i; --i) {
            int w = g[v][i];
            if(w <= u || w <= v) break;
            tmp[w] = 0;
        }
    }
```

# dfs树

对一个图运行 dfs 算法，每个点 $u$ 的父亲定义为第一次遍历 $u$ 时的前驱结点，若无则为根。

# dfs树

对一个图运行 dfs 算法，每个点 $u$ 的父亲定义为第一次遍历 $u$ 时的前驱结点，若无则为根。
无向图的 dfs 树没有横叉边。

# dfs树

对一个图运行 dfs 算法，每个点 $u$ 的父亲定义为第一次遍历 $u$ 时的前驱结点，若无则为根。
无向图的 dfs 树没有横叉边。
有向图的 dfs 树横叉边方向唯一。

# 画仙人掌

给 $n$ 个点仙人掌，给每个点分配一个坐标 $(x,\ y)$，使得边不相交。$x,\ y$ 为 1 到 $n$ 的整数。
$n \leq 10^5$

# 画仙人掌

考虑树怎么做，每个点 $x$ 坐标为深度，$y$ 坐标为 dfs 序。

# 画仙人掌

考虑树怎么做，每个点 $x$ 坐标为深度，$y$ 坐标为 dfs 序。
首先建出 *dfs* 树，对每个环中间新建一个点，向环上每一个点连边。新建的点成为虚点，虚点坐标可以是实数，然后不管环上边，对得到的树进行分配坐标。

# bfs树

对一个图运行 bfs 算法，每个点 $u$ 的父亲定义为第一次遍历 $u$ 时的前驱结点，若无则为根。

# bfs树

对一个图运行 bfs 算法，每个点 $u$ 的父亲定义为第一次遍历 $u$ 时的前驱结点，若无则为根。
非树边只存在在同一层的两个点和相邻层的点中。

# 时空阵

问 1 号点到 $n$ 号点距离恰好为 $m$ 的图的个数。图的边权为 1。
$n,\ m \leq 100$

# 时空阵

$dp(i,j,k)$ 表示做了前 $i$ 层，上一层 $j$ 个点，共 $k$ 个点的方案。
转移枚举这一层的连边方式，做到 $m$ 层即可。
对于 $m$ 层之后的边可以随便乱连。
一个小问题如何保证 $n$ 在第 $m$ 层，只要对答案 $*\frac{j}{n-1}$ 即可。

# 最短路

有负权边：spfa

# 最短路

有负权边：spfa
无负权边：dijkstra
每次选择距离最小的点更新周围点的距离。

# 最短路树

每个点 $u$ 的父亲为使 $u$ 得到最短距离的前驱节点，若有多个，则取任意一个。

# 传送门

有 $n$ 个结点，$m$ 个传送门，一个传送门双向连接两个结点，每当你从这两个结点之一跳进这个传送门时，你会花费一定的时间传送到另一个点处。现在你想用尽量短的时间从 $S$ 点走到 $T$ 点。这是一个简单的最短路问题。

但实际情况是这些传送门里有一个是损坏的。但你不知道损坏的是哪个，只有当你站在一个传送门的连接的两个结点之一时才能知道这个传送门是否是坏的。问你最少需要多少时间使得你能保证在这些时间内能从 $S$ 点走到 $T$ 点。

$n, m \leq 2 * 10^5$

# 传送门

首先考虑怎么求删掉一条边后相邻两个点到 $T$ 的距离。建出最短路树，如果删掉的不是连向父亲的边，则最短路不变，否则肯定是走到子树（包括自己）中的一个点，然后跳到子树外面。对每个点维护一下连接子树内和子树外的所有边以及价值，可以弄个堆维护一下。求 $u$ 时，将它所有孩子的边集的权值更新一下，再合并起来，从小往大找到第一条可以跳出子树的边，把在这条边之前的边都删掉就行了。可以使用可并堆或启发式合并。

# 传送门

对每个点 $u$，记 $d(u)$ 表示 $u$ 到 $T$ 的最短路，$e(u)$ 表示删掉它和最短路上父亲的边后的最短路。令 $dp(u)$ 表示 $S = u$ 时的答案。每次找到 $dp$ 值最小的点来更新其它的点的 $dp$ 值即可。用 $u$ 更新 $v$ 时的转移为 $dp(v) = \min(\max(dp(u) + w(u, v), u == parent\ v?e(v) : d(v)))$。

# 强连通分量

有向图强连通分量：在有向图 $G$ 中，如果两个顶点 $v_i, v_j$ 间有一条从 $v_i$ 到 $v_j$ 的有向路径，同时还有一条从 $v_j$ 到 $v_i$ 的有向路径，则称两个顶点强连通。
如果有向图 $G$ 的每两个顶点都强连通，称 $G$ 是一个强连通图。
有向图的极大强连通子图，称为强连通分量。

# Korasaju

将有向图 $G$ 中的每一条边反向形成的图称为 $G$ 的转置 $G^T$。
1.深度优先遍历 $G$,算出每个结点 $u$ 的结束时间 $f[u]$。
2.深度优先遍历 $G$ 的转置图 $G^T$ ,选择遍历的起点时,按照结点的结束时间从大到小进行。遍历的过程中,一边遍历,一边给结点做分类标记,每找到一个新的起点,分类标记值就加 1。
3. 第 2 步中产生的标记值相同的结点构成深度优先森林中的一棵树,也即一个强连通分量。

# SAT

有 $n$ 个需要满足的式子，每个式子形如 $(a \oplus b \oplus c \oplus \cdots)$。
2SAT 要求每个式子都只有不超过两个变量。

# 2SAT

对每个变量建立两个点 $a_0$，$a_1$，分别表示 $a$ 为假和 $a$ 为真。
对于 $(a \oplus b)$ 进行如下连边：
$a_0 -> b_1$
$b_0 -> a_1$
对得到的图进行强连通分量缩点，若存在一个变量 $a$ 使得 $a_0$ 和 $a_1$ 在同一个联通块中则无解，否则取两者更靠后的值。

# 游戏

小 L 计划进行 n 场游戏，每场游戏使用一张地图，小 L 会选择一辆车在该地图上完成游戏。
小 L 的赛车有三辆，分别用大写字母 A、B、C 表示。地图一共有四种，分别用小写字母 x、a、b、c 表示。其中，赛车 A 不适合在地图 a 上使用，赛车 B 不适合在地图 b 上使用，赛车 C 不适合在地图 c 上使用，而地图 x 则适合所有赛车参加。适合所有赛车参加的地图并不多见，最多只会有 d 张。
n 场游戏的地图可以用一个小写字母组成的字符串描述。例如：S=xaabxcbc 表示小 L 计划进行 8 场游戏，其中第 1 场和第 5 场的地图类型是 x，适合所有赛车，第 2 场和第 3 场的地图是 a，不适合赛车 A，第 4 场和第 7 场的地图是 b，不适合赛车 B，第 6 场和第 8 场的地图是 c，不适合赛车 C。

# 游戏

小 L 对游戏有一些特殊的要求，这些要求可以用四元组 (i,hi,j,hj)(i,hi,j,hj) 来描述，表示若在第 i 场使用型号为 hi 的车子，则第 j 场游戏要使用型号为 hj 的车子。
你能帮小 L 选择每场游戏使用的赛车吗？如果有多种方案，输出任意一种方案。如果无解，输出“-1”（不含双引号）。
$n \leq 50000$，$d \leq 8$，$m \leq 10^5$。

# 游戏

$d = 0$ 直接 2SAT 即可。
对于每张 $x$ 地图，枚举不用哪个车，可以发现只有两种不用车的情况。复杂度 $O(2^d(n+m))$。

# 网络流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $0 \leq f(e) \leq c(e)$，并且除了 $S$，$T$ 每个点流入量和流出量相等。最大化 $S$ 的流出量。

# dinic

每次通过 *bfs* 算法得到最短的增广路，构出分层图，在上面用 *dfs* 算法寻找增广路，如果找不到增广路则重新 *bfs* 构图，直到无增广路为止。

# dinic

每次通过 *bfs* 算法得到最短的增广路，构出分层图，在上面用 *dfs* 算法寻找增广路，如果找不到增广路则重新 *bfs* 构图，直到无增广路为止。
最短增广路距离不超过 $|V|$，故分层图个数是 $O(|V|)$。*dfs* 深度不会超过 $O(|V|)$，如果每次 *dfs* 后将得不到增广路的边在此次分层图中删去，则相当于每次花 $O(|V|)$ 复杂度删去一条边，故复杂度为 $O(|V||E|)$，总复杂度 $O(|V|^2|E|)$。

# Dining

有 $F$ 种食物和 $D$ 种饮料，每个人都有喜欢的饮料和食物的集合，而每种饮料和食物只能给最多一个人，问最多有多少个人同时得到喜欢的饮料和食物？

$1 \leq n, F, D \leq 100$。

# Dining

对食物、饮料分别建立一个点，对人建立两个点 $a$，$b$。$a$ 和喜欢的食物连边，$b$ 和喜欢的饮料连边。边的方向为 S $->$ 食物 $->$ a $->$ b $->$ 饮料 $->$ T。

# 反质数序列

对于一个长度为 $L \geq 2$ 的序列 $X : \{x_1, x_2, \cdots, x_L\}$，如果满足对于任意 $1 \leq i < j \leq L$，均有 $x_i + x_j$ 不为质数，则 JYY 认为序列 $X$ 是一个「反质数序列」。
JYY 有一个长度为 $N$ 的序列 $A : \{a_1, a_2, \cdots, a_N\}$，他希望从中选出一个包含元素最多的子序列，使得这个子序列是一个反质数序列。
$2 \leq N \leq 3000, 1 \leq a_i \leq 10^5$

# 反质数序列

可以发现两个数和为质数必然是 $1+1$ 或奇数 $+$ 偶数。
首先讨论下 $1$ 的情况。
对剩下的点建个图，两个点有边当且仅当两个点和为素数，问题变为删去最少的点使得不存在边，即为最小点覆盖。由于二分图有 $|$ 最小点覆盖 $|=|$ 最大匹配 $|$，故跑最大匹配即可。

# 无源汇有上下界可行流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$ 和最小容量 $d(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $d(e) \leq f(e) \leq c(e)$，并且每个点流入量和流出量相等。

# 无源汇有上下界可行流

新建源点 $S$ 和汇点 $T$，对于每条边 $(u,\ v),\ c(e),\ d(e)$，连接：
$(u,\ v),\ c(e) - d(e)$
$(S,\ v),\ d(e)$
$(u,\ T),\ d(e)$
如果 $S$ 到 $T$ 满流则有解。

# 有源汇有上下界最大流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$ 和最小容量 $d(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $d(e) \leq f(e) \leq c(e)$，并且除 $S$ 和 $T$ 每个点流入量和流出量相等。最大化 $S$ 到 $T$ 的流量。

# 有源汇有上下界最大流

与无源汇情况一样，新建 $SS$ 和 $TT$，每条边拆成三条。不同的是 $S$ 和 $T$ 不需要流量平衡，只用增加一条 $T$ 到 $S$，容量为 $\infty$ 的边即可。
先跑 $SS$ 到 $TT$ 的最大流，看是否满流。再跑一次 $S$ 到 $T$ 的最大流就是答案。

# 有源汇有上下界最小流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$ 和最小容量 $d(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $d(e) \leq f(e) \leq c(e)$，并且除 $S$ 和 $T$ 每个点流入量和流出量相等。最小化 $S$ 到 $T$ 的流量。

# 有源汇有上下界最小流

图与有源汇有上下界最大流一样。
跑完 $SS$ 到 $TT$ 的最大流，记录 $T$ 到 $S$ 的边流量为 $x$，删去 $T$ 到 $S$ 的边，跑 $T$ 到 $S$ 的最大流 $y$，答案为 $x-y$。

# 最小费用最大流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$ 和代价 $d(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $0 \leq f(e) \leq c(e)$，并且除 $S$ 和 $T$ 每个点流入量和流出量相等。最大化 $S$ 到 $T$ 的流量并且最小化 $\sum f(e)d(e)$。

# 最小费用最大流

以费用为边的距离，每次在残留网络中沿着最短路增广即可。

# 有源汇有上下界最小费用最大流

$n$ 个点 $m$ 条边的图，每条边有着最大容量 $c(e)$ 和最小容量 $l(e)$ 代价 $d(e)$。需要给每条边确定实际流量 $f(e)$ 满足 $l(e) \leq f(e) \leq c(e)$，并且除 $S$ 和 $T$ 每个点流入量和流出量相等。最大化 $S$ 到 $T$ 的流量并且最小化 $\sum f(e)d(e)$。

# 有源汇有上下界最小费用最大流

可以与最大流一样处理。
也可以对每个下届连一条代价为 $-\infty$ 的边。

# Snuke the Phantom Thief

二维平面上有 $n$ 个宝石，位置为 $(x,\ y)$ 价值为 $w$。有 $m$ 条限制，每个限制形如在给定的与一条坐标轴平行的直线下方最多只能取 $x$ 个宝石。问最多能取的宝石的价值是多少？

$n \leq 80,\ m \leq 320$。

# Snuke the Phantom Thief

首先枚举选了多少个物品 $k$，则限制可以转化为 $y = a, x = b$ 的直线下方左方选取数量为区间 $[l, r]$。
考虑只有一维怎么建，按坐标大小从小到大连边，容量为限制 $[l, r]$。当一个物品在限制直线的上方右方时，将那个限制连一条到这个物品容量为 1，费用为价值的边即可。
若存在可行流，则有解，更新答案。

# 平衡大师

给定一个 $n$ 个点 $m$ 条边的有向图，最多可以删去 $k$ 条边，要求使得入度与出度的差的绝对值的最大值最小。

$n, m \leq 200$

# 平衡大师

首先二分答案 $t$，然后求最少需要删去多少的边。
对每条边 $(u, v)$，连一条 $(u, v)$ 容量为 1，费用为 1 的边。如果流了表示删去这条边。
记 $d(u)$ 表示 $u$ 的入度-出度。
如果 $d(u) \geq t$，那么首先必需减小 $d(u) - t$,接下来还可以减小 $2t$，所以从 $S$ 向 $u$ 连下界为 $d(u) - t$,上界为 $d(u) + t$ 的边。
如果 $-t < d(u) < t$，那么从 $S$ 向 $u$ 连 $d(u) + t$ 的边，从 $u$ 向 $T$ 连 $t - d(u)$ 的边。
如果 $d(u) \leq -t$同上。
求这个图上的最小费用流。

# 无限之环

游戏在一个 $n \times m$ 的网格状棋盘上进行，其中有些小方格中会有水管，水管可能在方格某些方向的边界的中点有接口，所有水管的粗细都相同，所以如果两个相邻方格的公共边界的中点都有接头，那么可以看作这两个接头互相连接。水管有 15 种形状。
游戏开始时，棋盘中水管可能存在漏水的地方。
形式化地:如果存在某个接头，没有和其它接头相连接，那么它就是一个漏水的地方。
玩家可以进行一种操作：选定一个含有非直线型水管的方格，将其中的水管绕方格中心顺时针或逆时针旋转 90 度。
现给出一个初始局面，请问最少进行多少次操作可以使棋盘上不存在漏水的地方。
$nm \leq 2000$

# 无限之环

对格子黑白染色，对每个格子的四条边建四个点。一种颜色与 $S$ 相连，另一种与 $T$ 相连。
一个插口对初始位置连，再将初始位置向另外三个位置连。
两个插口的先对上左连，上对下连，左对右连。
三个插口对三个初始位置连，每个点再对剩下来一条边连。
四个插口直接连。
跑最小费用最大流即可。

祝大家省选取得好成绩!