图论专题第二次讲座

UESTC XCPC 集训队
图论专题负责组

2025 年 5 月 17 日

Statistics

本次专题共有二十六道题目，其中二十三道题目计入一血。

截至今日早上 9 点：

- 一共有 107 名同学报名参加本次专题，有 67 名同学通过了至少一道题目，较去年图论专题同期增长 1.17%。
- 一共有 10 名同学抢到了一血（其中 xzh 同学抢到了六道题的一血，图论之神啊 Orz）
- 一共有 341 发 AC 提交，较数据结构专题同期降低 10%。为什么呢呜呜呜 QAQ。
- 其中 G 的 std 被 zsy 同学薄纱了 Orz。

A. 欢迎来到 (省略若干形容词) 的图论专题

简要题意

给定一张有向图，求从一个固定点到每个点的最短路。

本题不计一血。
知识点：最短路。

A. 欢迎来到 (省略若干形容词) 的图论专题

题面背景里其实说的很清楚了，非负权图单源最短路 Dijkstra 算法是最佳选择。

A. 欢迎来到 (省略若干形容词) 的图论专题

题面背景里其实说的很清楚了，非负权图单源最短路 Dijkstra 算法是最佳选择。

下面是一些常见最短路算法的比较：

算法	Floyd	Bellman-Ford	Dijkstra	Johnson
类型	全源	单源	单源	全源
作用于	任意图	任意图	非负权图	任意图
检测负环？	能	能	不能	能
复杂度	$O(V^3)$	$O(VE)$	$O(E \log E)$	$O(VE \log E)$

在不引起歧义的情况下，V 表示图的顶点数，E 表示图的边数，后同。

B. 求求你跟我组一辈子 ACM 队吧我什么都会做的

简要题意

给一张 n 个点的无向图，点有点权 $\{a_i\}$。初始没有边，连边有代价 $\{b_{ij}\}$。要让每个连通块至少有一个点被选中，求（连边代价 + 选中点的点权和）的最小值。

一血：蒋帅泉
分享者：蒋帅泉
知识点：最小生成树。

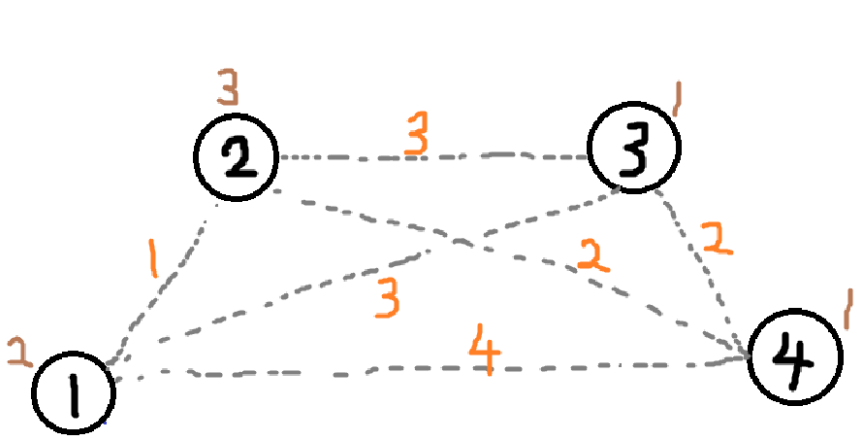
B. 求求你跟我组一辈子 ACM 队吧我什么都会做的

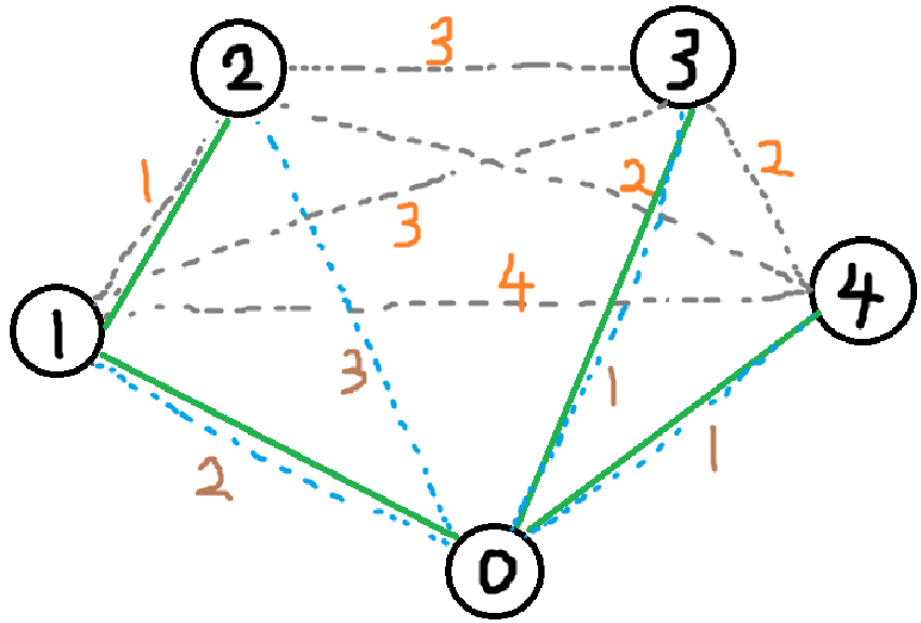
建虚点。新建一个 0 号点，0 号点向其它所有 n 个点连边，规定边 $(0, i)$ 的边权是 a_i。这个新图的最小生成树的权值就是所求答案。

此时最小生成树中的每个点，要么与 0 直接相连，要么与 0 间接相连。与 0 直接相连的点就是其所在连通块的选中点，其点权也被算到最小生成树中了。因为最小生成树的意义就是用尽可能小的边权和让整个图连通，容易说明这么做是正确的。

常用 Kruskal 算法求解最小生成树，时间复杂度为 $O(E \log E)$。本题中，共有 $E = n^2 + n$ 条边，故时间复杂度 $O(n^2 \log n)$。

B. 求求你跟我组一辈子 ACM 队吧我什么都会做的





如图为一个 $\{a\} = \{2, 3, 1, 1\}$ 的示例，0 是虚点，最小生成树以绿色标出。1, 2 一个队，3 一个队，4 一个队。

P. Teleporter

简要题意

给定起点终点和 m $(1 \le m \le 10^5)$ 个中转点，坐标范围 $[0, 10^9]$，每次移动时只能横（纵）坐标加（减）一，代价是 1。当前坐标和某一中转点在横（纵）坐标相同时，可以立刻瞬移到中转点（不消耗时间）。在这种条件下求起点到终点的最短路。

一血：贾承羲
分享者：贾承羲
知识点：最短路 / 建图。

P. Teleporter

本题关键在于如何建图跑最短路，直接两两连边显然会爆掉。

由于本题的距离只用考虑一维，也就是只要有三个点满足 $x_1 < x_2 < x_3$，则 x_2 的点加入不影响 x_1 的点到 x_3 的点的最短距离。因此可以依次按照 x, y 关键字进行两次排序，每次排序后将相邻的点连边。最后将所有时空锚点再与起点终点连边，跑一遍堆优化的 Dijkstra 即可。

时间复杂度 $\mathcal{O}(m \log m)$。

Q. 不要等到失去……

简要题意

给一张简单无向图，求割点数量，割边数量，点双连通分量数量，边双连通分量数量。

本题不计一血。
知识点：tarjan 全家桶。

Q. 不要等到失去.....

Tarjan 算法的模板题。

参考资料：OI wiki。
Link1：双连通分量。
Link2：割点，割边。

C. Brainrot Memes 2

简要题意

给出 n 个长度为 3 的字符串，判断这 n 个字符串能否拼接为长度为 $n+2$ 的字符串。即，判断这 n 个字符串是否为某个长度为 $n+2$ 的字符串的所有长度为 3 的子串。

一血：蒋帅泉
分享者：蒋帅泉
知识点：欧拉路径。

C. Brainrot Memes 2

欧拉路径的定义

欧拉通路：通过图中每条边恰好一次的通路。
欧拉回路：通过图中每条边恰好一次的回路。

无向图欧拉路径的判别（前提是非零度点连通）

欧拉通路：图中恰有两个点的度数为奇数，其余点度数均为偶数。两个奇度点分别作为起点和终点。
欧拉回路：图中所有点的度数均为偶数。

有向图欧拉路径的判别（前提是非零度点弱连通）

欧拉通路：图中恰有一个点的出度比入度大 1（作为起点），恰有一个点的入度比出度大 1（作为终点），其余点入度均等于出度。
欧拉回路：图中所有点的入度等于出度。

C. Brainrot Memes 2

考虑对于原字符串中连续的一部分，其分割为长度为 3 的子串后具有的特征。比如 abcde，那么对应产生 abc、bcd 和 cde。很容易知道，原来连续的字符串具有两个字符的重叠。

于是我们考虑建图：将给出的长度为 3 的字符串中，s_0s_1 作为一个点，s_1s_2 作为一个点，在 s_0s_1 和 s_1s_2 之间连一条有向边。这样子原本的每个字符串对应图中的一条边，而字符串之间的连接则在图中表现为具有相同的端点。这样子，字符串之间自然就产生了关联。

C. Brainrot Memes 2

下面我们需要判断题目中给出的 n 个字符串是否合法，能否拼接为一个字符串，如果能，如何拼接。很容易想到，在我们尝试直接拼接字符串的时候，当给出的所有字符串都用到且只用了一次时，最终可以获得合法结果。那么对于其转化后的图，由于给出的每个字符串对应图上的一条边，我们需要找到一条路径，使得其经过图中每条边恰好一次。也就是找到原图的一条欧拉路径。

C. Brainrot Memes 2

那么我们判断之后，使用 Hierholzer 算法即可。其思想简单：从起点开始 dfs，遍历没走过的边，遍历完一个顶点所有的边之后将顶点入栈。最终我们按照栈的顺序取栈内元素，即可得到对应欧拉路径或欧拉回路。时间复杂度 $O(n)$。

代码实现上比较简单，唯一可能麻烦的点在于对于字符和点的映射的处理可能略微麻烦，也可以直接使用 `unordered_map`。此外需要注意记录每个点尝试到了哪一条边，每次从未经过的边开始遍历，类似当前弧优化。

K. 罗盘指向了黑暗

简要题意

给定一棵有根树，你可以重排任意节点的儿子，问这棵树能否是对称的。

一血：蒋帅泉
分享者：蒋帅泉
知识点：树哈希。

K. 罗盘指向了黑暗

解决本题首先需要了解一种叫树哈希的算法。所谓树哈希，就是通过某些方法将一颗树转化为哈希值，可以通过比较哈希值快速比较两棵树是否同构。

一种常见的哈希方法如下（$f(x)$ 表示以 x 为根的子树的哈希值）：

$$f(x) = \left(c + \sum_{y \in son[x]} g(f(y)) \right) \bmod m$$

其中 c 为常数，一般使用 1。m 为模数，一般使用 2^{32} 或 2^{64} 进行自然溢出，也可使用大素数。g 为整数到整数的映射，可以使用 xor shift，也可以选用其他的函数。同时为了预防出题人对着 xor hash 卡，还可以在映射前后异或一个随机常数。

K. 罗盘指向了黑暗

一种 xor shift 实现：

```
using ull = unsigned long long;

mt19937_64 rng(chrono::steady_clock::now().
    time_since_epoch().count());
ull mask = rng();
ull shift(ull x) {
    x ^= mask;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    x ^= mask;
    return x;
}
```

K. 罗盘指向了黑暗

一种 $c = 1$，自然溢出，$g =$ xor shift 的树哈希实现：

```
ull f[N];
void dfs(int x, int fa) {
    f[x] = 1;
    for (auto y : son[x]) {
        ...
        dfs(y, x);
        ...
        f[x] += shift(f[y]);
    }
}
```

K. 罗盘指向了黑暗

回到本题，可以先考虑一个点 x 有偶数个儿子的情况，此时如果能有一种重排方式使得 x 安全，那么所有子树的哈希值必须出现偶数次。

有奇数个儿子的情况呢？试想，我们将相同的子树一左一右从外侧向中间填充，最后最中间的那棵子树可以没有其它子树与它相同。

于是，我们算出 x 所有子树的哈希值，看哪个哈希值出现了奇数次，对应的那棵子树只能放在中间。

K. 罗盘指向了黑暗

当然，题目中给出的是一个递归的定义：也就是最中间的那棵子树仍然需要是对称的。

一种直观的想法是先以 1 为根跑一遍树哈希，算出所有点子树对应的哈希值。接着从根开始 dfs，如果当前点有奇数个儿子，就按之前的方法找出需要在中间的那棵，递归下去判断。

更加稳妥的写法是记一个 $h[x] = 0/1$ 表示以 x 为根的子树是否安全，dfs 回溯的时候我们已经确定了 x 所有儿子 y 的 $h[y]$，这样就能直接判断了。最后答案也就是 $h[1]$。

时间复杂度 $O(n)$。

F. 徽章商店

简要题意

有 n 种物品，每种若干，所有物品被分成 m 堆。每次你能用 1 的代价清空一堆，或所有堆中的某一特定类别，问清空所有物品的最小代价。

一血：贾承羲
分享者：贾承羲
知识点：König 定理 / 二分图匹配。

F. 徽章商店

我们可以构建一张二分图，这张二分图有 m 个左部点，第 i 个左部点表示第 i 堆；有 n 个右部点，第 j 个右部点表示第 j 种物品。如果第 i 堆里有第 j 种物品，我们就从第 i 个左部点向第 j 个右部点连一条边。

现在考虑答案表现在图中是什么。

- 第一种操作：清空一堆。相当于把一个左部点标记为选中。
- 第二种操作：把所有堆中的一类物品选走。相当于把一个右部点标记为选中。

什么是清空所有物品？一个物品就是一条边。因此相当于选中最少的点使得图中每一条边都至少有一个端点被选中。这在图论中称为「最小点覆盖」。

F. 徽章商店

König 定理：二分图中，最小点覆盖 = 最大匹配。

常见求解二分图最大匹配的算法是匈牙利算法（Hungarian Algorithm），可以做到 $O(VE)$。

如果你已经学到了什么是网络流，那么 Dinic 算法也是求解二分图最大匹配的不错选择，复杂度是 $O(E\sqrt{V})$。

此外，Hopcroft-Karp 算法同样能做到 $O(E\sqrt{V})$。

本题数据范围十分宽松，预期使用上述任意一种算法即可通过。

O. The World of Goo 2

简要题意

给定 n 行 m 列的网格图，且每个网格左上角到右下角存在边连接。每条边有正整数边权。删去一条边的代价为其边权，求使得网格图最左上角顶点与最右下角顶点不连通所需最小代价。

一血：幸子豪
分享者：丁柏宇
知识点：平面图转对偶图，最短路。

O. The World of Goo 2

如果了解网络流的话，可以简单地看出题目就是求最小割。那么直接按照题意建图，跑最大流算法，是不是就可以了呢？

考虑到数据范围很大，其实跑普通的网络流算法（比如 dinic）并不是一个良好的选择。虽然很多时候网络流算法的时间复杂度跑不满，但是这也不意味着它一定能过。造数据时使用特殊生成的数据使得常见的 dinic 算法获得 TLE 的结果。

O. The World of Goo 2

如果了解网络流的话，可以简单地看出题目就是求最小割。那么直接按照题意建图，跑最大流算法，是不是就可以了呢？

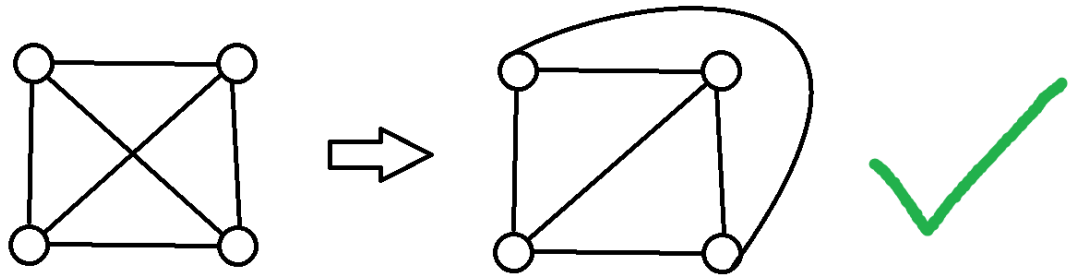
考虑到数据范围很大，其实跑普通的网络流算法（比如 dinic）并不是一个良好的选择。虽然很多时候网络流算法的时间复杂度跑不满，但是这也不意味着它一定能过。造数据时使用特殊生成的数据使得常见的 dinic 算法获得 TLE 的结果。

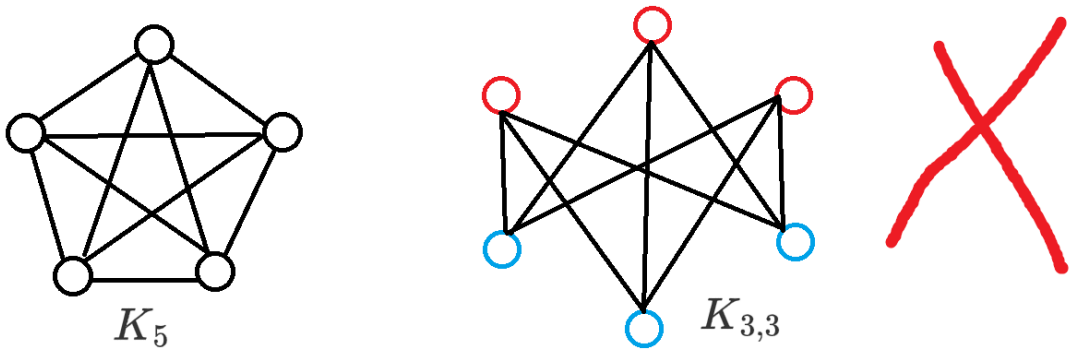
那么跑复杂的充满人类智慧的快到飞起的网络流算法（比如 hlpp）呢？你赢了... 它确实很快，比 std 都快。那么，代价是什么呢？需要开很多数组，空间开销大！因此造数据的时候将边权上限放大至 10^9，那么使用 hlpp 算法时，`long long` 的数组就开不下了，会获得 MLE 的结果。

O. The World of Goo 2

言归正传。本题的特殊性质在于图的结构：具有特定结构的网格图。事实上，这个网格图是一个平面图。

平面图是什么呢？最简单的话来讲，就是能把它画在纸上（平面上），画出来的边不会相交。比如：



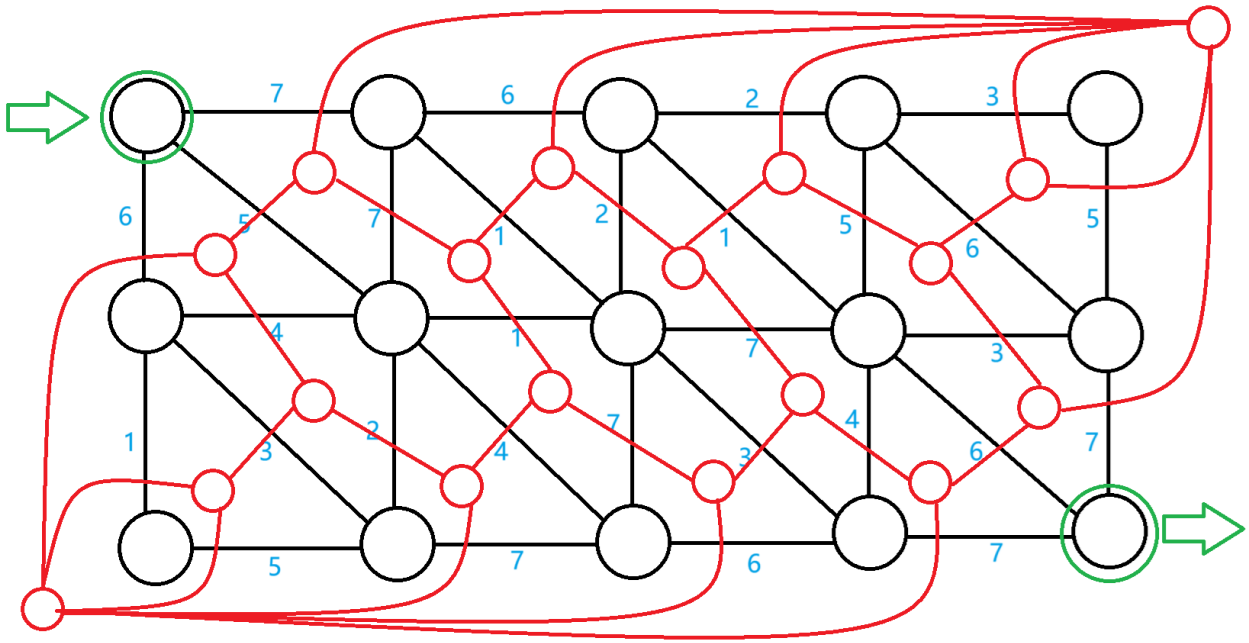


库拉托夫斯基定理：图 G 是平面图当且仅当 G 不含与 K_5 或 $K_{3,3}$ 同胚的子图。此处具体不展开讲解。

O. The World of Goo 2

下面我们介绍对偶图。对偶图简单来讲就是在平面图的每个面上放一个顶点，之后对于原图的边 e，对偶图上的对应边 e' 与且仅与 e 相交，连接对偶图上的两个顶点。其具体严格的定义可以自行查阅。

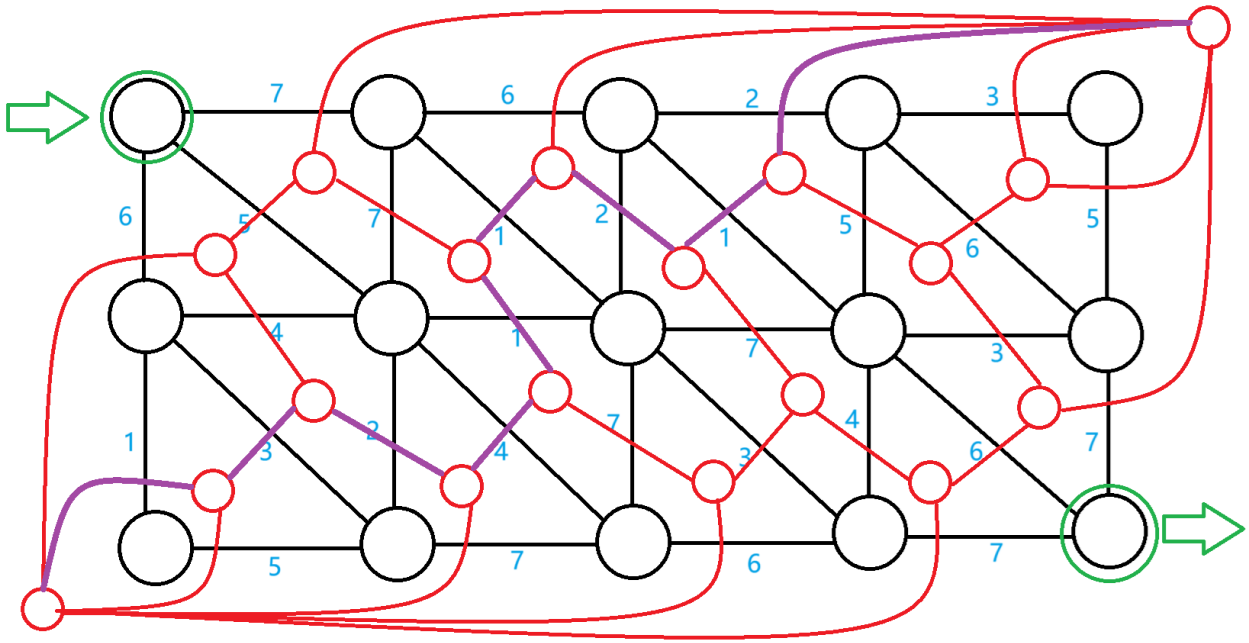
其应用在于，平面图的最小割可以转化为在其对偶图上求最短路，比如对于题目中的网格图，可以建图如下：



O. The World of Goo 2

虽然其不是严格按照对偶图定义进行转化，但是本质相同，只是把最外面平面对应的一个点进行了拆点处理。

那么在这个对偶图上求其最短路，就对应原图的一个最小割：



即可极大简化时间复杂度，跑最短路显然要快得多。时间复杂度 $O(nm\log(nm))$。

N. The World of Goo

简要题意

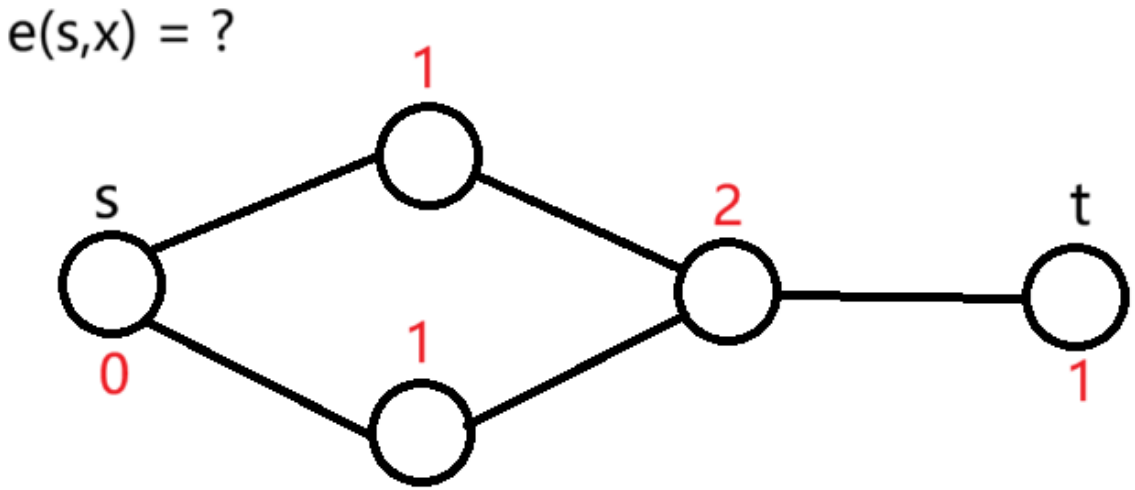
给定 n 个点，m 条边的无向图，对于所有 $1 \leq s < t \leq n$，计算 $f(s,t)$：位于 s 到 t 至少一条最短路上的边的数量。

一血：李鸣宇
分享者：李鸣宇
知识点：Floyd。

N. The World of Goo

首先我们考虑“位于从 s 到 t 至少一条最短路上边的数量”如何计算。

定义 $e(u,v)$ 表示和 v 直接相连的，位于某条从 u 到 v 的最短路上的边的数量。
定义 $S(s,t)$ 表示从 s 到 t 所有最短路上包含的点的集合。则 $f(s,t)=\sum_{v\in S(s,t)} e(s,v)$.



N. The World of Goo

首先我们考虑“位于从 s 到 t 至少一条最短路上边的数量”如何计算。

定义 $e(u,v)$ 表示和 v 直接相连的，位于某条从 u 到 v 的最短路上的边的数量。
定义 $S(s,t)$ 表示从 s 到 t 所有最短路上包含的点的集合。则 $f(s,t)=\sum_{v\in S(s,t)} e(s,v)$.

注意其定义，$e(u,v)$ 比起题目中要求的 $f(s,t)$，增加了一个“和 v 直接相连的”这一限制。这就使得我们可以把 $f(s,t)$ 的计算拆解为：对于 $S(s,t)$ 这个从 s 到 t 上所有最短路包含的点的集合，累加 $e(s,v)$。

N. The World of Goo

所以我们只需要考虑如何计算出所有的 $e(u,v)$ 和 $S(s,t)$。实际上，其计算是容易的。

首先使用 floyd 算法求出全源最短路，时间复杂度 $O(n^3)$。

之后枚举所有边 (u,v,w)，再枚举所有点 i，如果 $dis(i,u)+w=dis(i,v)$，说明“从 i 到 u 的最短路”加上“当前边 (u,v)”就是" 从 i 到 v 的最短路"，也就说明了当前边 (u,v) 是从 i 到 v 最短路上的一条边。由于其端点包含 v，自然满足“与 v 直接相连”。那么我们就为 $e(i,v)$ 加一。当然，对于 $e(i,u)$ 需要做相同判断和处理。这个过程中，由于边数 $m \le \frac{n\cdot(n-1)}{2}$ ，所以枚举边再枚举点的时间复杂度为 $O(n^3)$。

N. The World of Goo

对于 S，我们其实不需要实际计算出该集合。我们仍然通过 floyd 得到的结果，通过类似思想完成计算。首先枚举 s 和 t 作为起点和终点，之后直接枚举所有点，判断其是否为从 s 到 t 最短路上的一点即可。这个其实是很简单的，将枚举的点 x 作为中间点，如果 $dis(s,x)+dis(x,t)=dis(s,t)$，就说明“从 s 到 x 的最短路”加上“从 x 到 t 的最短路”等于“从 s 到 t 的最短路”，那么 x 自然属于集合 $S(s,t)$。那么我们直接累加 $e(s,x)$ 进答案即可。这个过程的时间复杂度仍然为 $O(n^3)$。

因此总时间复杂度 $O(n^3)$。

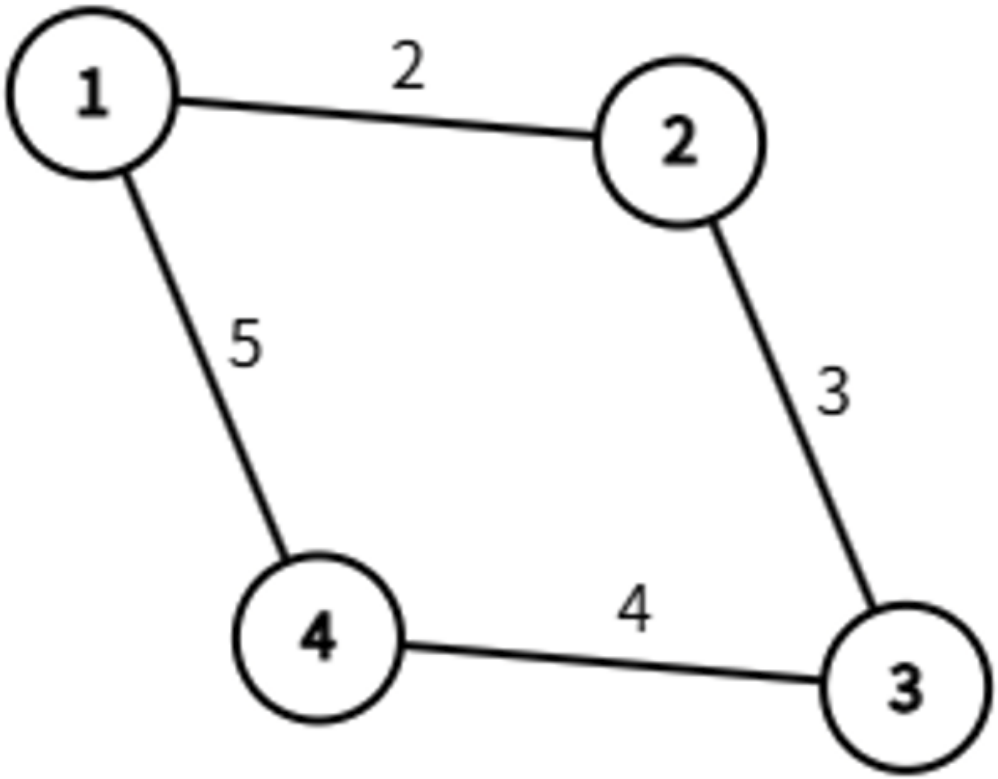
H. 鸡煲拯救尖塔

简要题意

一个四边形，每条边有一个长度，从第一个点开始跑，在第一个点结束。求大于等于 K 的最短距离。
$(1 \leq K \leq 10^{18}$，边最长 $2 \cdot 10^5)$

一血：吴长胜
分享者：吴长胜
知识点：同余最短路。

H. 鸡煲拯救尖塔



H. 鸡煲拯救尖塔

因为我们要从第一个点开始，并从第一个点结束。所以一种显然的方案是，我们可以来回跑跟第一个点相连的两条边。

设 $w = \min(d_{41}, d_{12})$ ，假设我们有一种跑法可以得到 d 的距离，则我们肯定能得到距离为 $d + 2wt, (t \geq 0)$ 的跑法。接下来思考怎么求出 d。

H. 鸡煲拯救尖塔

因为我们要从第一个点开始，并从第一个点结束。所以一种显然的方案是，我们可以来回跑跟第一个点相连的两条边。

设 $w = \min(d_{41}, d_{12})$ ，假设我们有一种跑法可以得到 d 的距离，则我们肯定能得到距离为 $d + 2wt, (t \geq 0)$ 的跑法。接下来思考怎么求出 d。

设 $d[i][j]$ 表示从起点出发到达点 i ，距离模 $2w$ 为 j 的最短路距离。之后用最短路求出解，我们就可以枚举 $d[1][j]$ 的答案来求出大于等于 K 的最短距离。

时间复杂度为 $O(w \log w)$。

X. 狂气の Mad Scientist

简要题意

有一个 $n \times m$ 的矩阵，c_{ij} 表示这个矩阵第 i 行，第 j 列的元素。并给你两个值 L 和 R。请找出序列 $a_1, a_2, \ldots, a_n$ 和序列 $b_1, b_2, \ldots, b_m$，使得你得到一个新的 $n \times m$ 的矩阵，该矩阵元素 c'_{ij} 的值等于 $\frac{c_{ij} \times a_i}{b_j}$。要求新矩阵的每个元素的值在 L 和 R 之间。

一血：丁柏宇
分享者：丁柏宇
知识点：差分约束，负环。

由题中式子：

$$L \leq \frac{c_{ij} \times a_i}{b_j} \leq R$$

两边取 log 得到

$$\log L \leq \log c_{ij} + \log a_i - \log b_j \leq \log R$$

继续进行变换得到

$$\log a_i - \log b_j \leq \log R - \log c_{ij}$$
$$\log b_j - \log a_i \leq \log c_{ij} - \log L$$

对于这种不等式形式，我们可以用差分约束来解。

X. 狂气の Mad Scientist

差分约束系统是一种特殊的 n 元一次不等式组，它包含 n 个变量 $x_1, x_2, \ldots, x_n$ 以及 m 个约束条件，每个约束条件是由两个其中的变量做差构成的，形如 $x_i - x_j \leq c_k$，其中 $1 \leq i, j \leq n,\ i \neq j,\ 1 \leq k \leq m$ 并且 c_k 是常数（可以是非负数，也可以是负数）。

我们要解决的问题是：求一组解 $\begin{cases} x_1 = a_1 \\ x_2 = a_2 \\ \cdots \\ x_n = a_n \end{cases}$，使得所有的约束条件得到满足，否则判断出无解。

X. 狂气の Mad Scientist

差分约束系统中的每个约束条件 $x_i - x_j \leq c_k$ 都可以变形成 $x_i \leq x_j + c_k$，这与单源最短路中的三角形不等式 $dist[y] \leq dist[x] + z$ 非常相似。

因此，我们可以把每个变量 x_i 看做图中的一个结点，对于每个约束条件 $x_i - x_j \leq c_k$，从结点 j 向结点 i 连一条长度为 c_k 的有向边。从而将差分约束问题转化为最短路问题。

注意到，如果 $a_1, a_2, \ldots, a_n$ 是该差分约束系统的一组解，那么对于任意的常数 d，$a_1 + d, a_2 + d, \ldots, a_n + d$ 显然也是该差分约束系统的一组解，因为这样做差后 d 刚好被消掉。

所以对于差分约束系统，只有两种情况：有无穷多解或无解。

设 $dist[0] = 0$ 并向每一个点连一条权重为 0 的边，跑单源最短路，若图中存在负环，则给定的差分约束系统无解，否则，$x_i = dist[i]$ 为该差分约束系统的一组解。

X. 狂气の Mad Scientist

差分约束问题一般让我们求最大解、最小解、以及有无解。

- 求最大值，找 $x_1 - x_2 \leq k$ 类型的式子，从 x_2 向 x_1 连接一条边权为 k 的边，跑最短路，有负环则无解。
- 求最小值，找 $x_1 - x_2 \geq k$ 类型的式子，从 x_2 向 x_1 连接一条边权为 k 的边，跑最长路，有正环则无解。

回到此题，我们利用上述不等式将图建出来，若图中存在负环，则给定的差分约束系统无解。判负环使用 SPFA 算法。最坏时间复杂度为 $O(VE)$。

M. 影舞之心

简要题意

求 n 个点的带标号生成树个数。

一血：滕鎧泽
分享者：滕鎧泽
知识点：Prüfer 序，Cayley 定理。

M. 影舞之心

Prüfer 序列可以将一个带标号 n 个结点的树用 $[1, n]$ 中的 $n-2$ 个整数表示。

也可以把它理解为完全图的生成树与数列之间的双射。

Prüfer 是这样建立的：每次选择一个编号最小的叶结点并删掉它，然后在序列中记录下它连接到的那个结点。重复 $n-2$ 次后就只剩下两个结点，算法结束。

M. 影舞之心

那么实际上将一个初始没有边的图连成一棵生成树，可以看作用 Prüfer 序列还原原树，那么任意一个长度为 $n-2$，值域为 $[1,n]$ 的序列都可以对应还原出来一个树，所以可以得到如下 Cayley 定理。

- 完全图 K_n 有 n^{n-2} 棵生成树。

回到本题，本题实际上对于一段区间的答案可以差分为两个前缀和相减，然后对于固定选 n 个点的方案数用组合数来进行计算即可。

D. 比赛选址

简要题意

给你每个场地的类型和一堆场地之间的限制条件，问是否存在一组合法的方案使得所有限制都能满足。

一血：幸子豪
分享者：幸子豪
知识点：2-SAT。

D. 比赛选址

我们可以把限制条件写成如下形式：

- 表达式 $(b_i = x) \vee (b_j = y)$ 为真。

首先考虑 $n = 0$ 的情况，此时每个场地恰好只有 2 个选择，可以根据这个建立一个 2-SAT 模型。

具体如何去建呢？

D. 比赛选址

考虑将 2 个选择表示成 0/1，那么所有点都可以拆成 2 个点 $2i-1$ 和 $2i$ 分别表示这个点选择了 0/1，后续表达中用 i' 表示 $i\ xor\ 1$。

然后通过有向边来表示这些关系，定义有向边 $x \to y$ 表示选择了 x 则必须选择 y。

那么如下操作都可以被表示出来：

- i,j 不能同时选：$i \to j',\ j \to i'$。
- i,j 必须同时选：$i \to j,\ j \to i$。
- i,j 至少选一个：$i \to j',\ j \to i',\ i' \to j,\ j' \to i$。
- i 必须选：$i' \to i$。

根据题中给的要求按上述方式建图即可。

D. 比赛选址

- 如何判断无解？
  根据上述的建图，我们可以发现只有 i 和 i' 在同一个强连通分量时才无解。
- 怎么做？
  Tarjan 求强连通分量后判断即可。
- 如何构造方案？
  输出方案时可以通过变量在图中的拓扑序确定该变量的取值。
  如果变量 x 的拓扑序在 $\neg x$ 之后，那么取 x 值为真。应用到 Tarjan 算法的缩点，即 x 所在 SCC 编号在 $\neg x$ 之前时，取 x 为真。
  因为 Tarjan 算法求强连通分量时使用了栈，所以 Tarjan 求得的 SCC 编号相当于反拓扑序。

D. 比赛选址

现在考虑有 n 的情况。因为 $d \leq 8$，所以我们可以考虑 3^d 枚举每个 n 是 I，S 还是 R。

但我们发现，我们其实只用枚举每个 n 是 I,S,R 中的两种就可以了。例如，枚举了 n 是 I，代表这个地方只能举办 S 和 R；枚举了 n 是 S，代表这个地方只能举办 I 和 R。如此一来，已经覆盖了 S,I,R 这三种情况了，就不需要再枚举 n 是 R 了。

总复杂度为 $O(2^d \times (n+m))$。

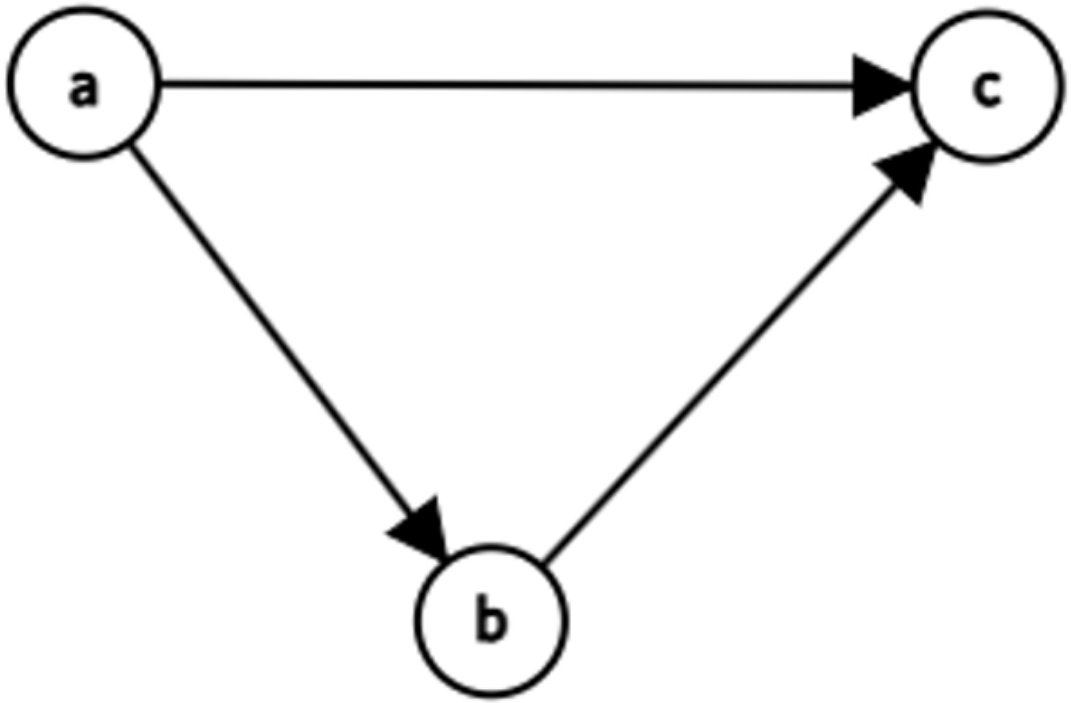
Z. 咔咔

简要题意

有一个带点权的有向图 G，若存在三元组 (a, b, c) 使得 a 到 b 有边，b 到 c 有边，则连一条 a 到 c 的边，如此不断操作。求图 G 最长链的最小权重。

一血：周航远
分享者：周航远
知识点：强连通分量 / 拓扑排序。

Z. 咔咔



Z. 咔咔

分析操作，我们发现该操作并不能对图的连通性有影响，即原来 a，b 可达，则操作后 a，b 仍可达；若原来 a，b 不可达，则操作后 a，b 仍不可达。对于一个强连通分量来说，因为强连通分量里的任意两个点连通，使用这个操作可以使得强连通分量里的图变成一个完全子图，即任意两点可以直接到达。

Z. 咔咔

分析操作，我们发现该操作并不能对图的连通性有影响，即原来 a，b 可达，则操作后 a，b 仍可达；若原来 a，b 不可达，则操作后 a，b 仍不可达。对于一个强连通分量来说，因为强连通分量里的任意两个点连通，使用这个操作可以使得强连通分量里的图变成一个完全子图，即任意两点可以直接到达。

现在想走的策略。假设你想从 a 走到 b 且 a 到 b 可达。

- 如果 a，b 在同一个强连通分量里，因为是完全子图，所以你可以从 a 不重复地走完子图，之后从 b 出来；
- 如果 a，b 并不在同一个强连通分量里，因为上述操作你可以直接从 a 到 b，但是因为 a，b 不在同一个强连通分量里，即你不能从 b 到 a（如果 b 到 a 可达，则 a，b 应该在同一个强连通分量里。）所以为了走最多的点，最优策略是尽可能把 a 到 b 路径上的点都走完。

Z. 咔咔

因为在本题中我们可以从强连通分量的点进去，然后从任一个点走出。我们就可以把这个强连通分量看成一个点。我们把所有强连通分量各缩成一个点，然后建新图，新图即是一个 DAG（有向无环图）。

最后该题简化为在 DAG 上找最长链。既可以拓扑排序 + dp，也可以 dfs 记忆化爆搜。

时间复杂度 $O(n)$。

Z. 咔咔

- 拓扑排序 + dp:
  设 f_u 为以 u 为起点的最长链长度，g_u 为以 u 为起点的最长链的最小点权和。

  有转移方程 $f_u = \max\{f_u, f_v + w_v\}, (u, v) \in E$。

  显然当 $f_u < f_v + w_v$ 时，$g_u = g_v + w'_v$。

  而当 $f_u = f_v + w_v$ 时，$g_u = \min\{g_u, g_v + w'_v\}$。

  上述 dp 顺序使用拓扑排序顺序。

- dfs 记忆化爆搜:
  dfs 所有点，每个点记录状态 f_u, g_u，含义与上述相同，利用 dfs 更新状态即可。

J. 太阳向海的东方落下

简要题意

给定一棵 n 个点的树，你需要找出一条长度不超过 k 的链（链的长度定义为点的数量），使得离这条链最远的点到这条链的距离最小，输出这个距离。

一血：吴俊达
分享者：吴俊达
知识点：树的直径。

J. 太阳向海的东方落下

树的直径

定义：树上任意两节点之间最长的简单路径。

性质：

- 树的直径不唯一，但所有直径一定交于一点。
- 对于树上任意一点，离它最远的点一定是直径的某个端点。
- 对于一棵树的任意两个连通子图 G, H，设 G 的两个直径端点为 $\{a, b\}$，H 的两个直径端点为 $\{c, d\}$，则 $G \cup H$ 的两个直径端点一定落在集合 $\{a, b, c, d\}$ 中。

J. 太阳向海的东方落下

结论：最佳的路径一定被直径包含。

也就是说，设 D 表示直径的长度（这里长度同样定义为在直径上的点的数量）。

- 如果 $D \leq k$，我们选出的那条链就是直径。
- 如果 $D > k$，我们选出的链就是直径上的一段。

J. 太阳向海的东方落下

证明如下，设 d_{AB} 表示 A, B 两点间的距离。

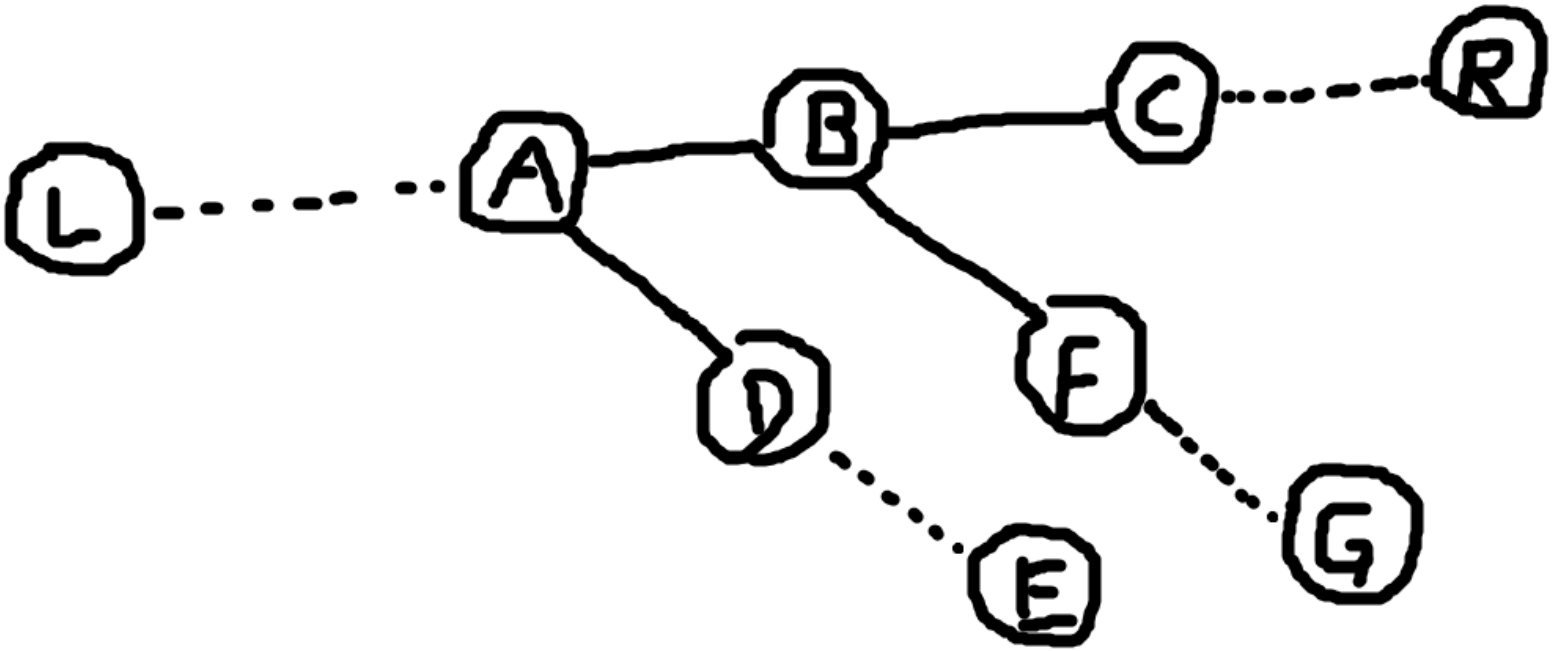
- 首先，一个经典的结论是：对于树上任意一点，离它最远的点一定是直径的两个端点之一。对于直径外一点 x，设离 x 最近的直径上的点是点 y，当 x 向 y 靠近时，离 x 最远的点仍然是直径的两个端点，但最远距离减小，故取 y 优于 x。于是，我们选的第一个点肯定是在直径上。

J. 太阳向海的东方落下

证明如下，设 d_{AB} 表示 A, B 两点间的距离。

- 其次，设我们选的第一个点是 A，接下来证明 A 沿着直径扩展最佳。如下图，L, R 是直径的两个端点，E, G 分别是 D, F 向下延伸出去最远的点。若从 A 出发沿直径选了 B，答案是 $\max(d_{LA}, d_{AE}, d_{BG}, d_{BR})$；若沿子树方向选了 D，答案是 $\max(d_{LA}, d_{DE}, d_{AR}, d_{AG})$。依据直径的性质有 $d_{BG} < d_{BR}$，$d_{AG} < d_{AR}$，$d_{AE} < d_{LA}$，$d_{DE} < d_{LA}$，此时相当于比较 $\max(d_{LA}, d_{BR})$ 与 $\max(d_{LA}, d_{AR})$，又 $d_{AR} > d_{BR}$，故选 B 相比选 D 更优。将 A, B 看成一个整体，同理可证明选 C 比选 F 要更优，以此类推..... 因此路径肯定是沿着直径方向一直扩展。

J. 太阳向海的东方落下



J. 太阳向海的东方落下

结论：最佳的路径一定被直径包含。

现在我们知道了选哪条链，答案的计算就比较简单了。$D \leq k$ 的时候，选完整条直径，我们可以以直径上每个点为根向远离直径方向的子树 dfs，求出最远点到这条直径的距离。$D > k$ 的时候，我们可以在直径上搞一个长度为 k 的滑动窗口，贡献有两种情况。

- 第一种在滑动窗口内，和 $D \leq k$ 时一样，可以预处理出直径上每个点向远离直径方向延申出去的最远距离 `dis`。这样答案就是一段区间 max。
- 第二种是滑动窗口两端，贡献由直径上的一段和上述提到的 `dis` 构成，可以在直径上预处理出一段前缀 max 以及后缀 max 快速判断。

时间复杂度 $O(n \log n)$，若用单调队列求极值就是 $O(n)$。

Y. 翁法罗斯的百界门计划

简要题意

给定一个 $(1 \leq n \leq 200)$ 个点，m $(1 \leq m \leq 50000)$ 条边的图，同时给定两个正整数 G, S，每条边有 g, s 两个权重，求图的一棵生成树 T，使得 $G \times \max_{e \in T} g_e + S \times \max_{e \in T} s_e$ 最小，求出这个最小值。

一血：幸子豪
分享者：幸子豪
知识点：最小生成树。

Y. 翁法罗斯的百界门计划

由于是双变量最优化问题，考虑枚举某一个变量对另外一个变量进行优化。在本题中可以枚举最大的 g 值，用所有小于等于 g 的边建图，然后在这张图上以最小化 $\max_{e\in T} s_e$ 跑 Kruskal 最小生成树。

这样做的时间复杂度为 $\mathcal{O}(m^2 \log m)$。

Y. 翁法罗斯的百界门计划

如何优化？可以发现以下性质：

- 如果一个边在上一次枚举 g 的时候没有被选中，那么在后续的枚举（假设是从小到大枚举 g 值）中一定不会被选中。

Y. 翁法罗斯的百界门计划

如何优化？可以发现以下性质：

- 如果一个边在上一次枚举 g 的时候没有被选中，那么在后续的枚举（假设是从小到大枚举 g 值）中一定不会被选中。

因为在上一轮中已经可以选择一些拥有更小 s 的边来连接使一些点连通，没有选中的边对于这些点的连通性不做贡献；而且可选的边会越来越多，肯定没有必要选当前没有选中的边，直接删掉这些边即可，这样可以保证每一轮所剩下的边只有最多 $n-1$ 条。时间复杂度降为 $\mathcal{O}(nm\log n)$。

Y. 翁法罗斯的百界门计划

如何优化？可以发现以下性质：

- 如果一个边在上一次枚举 g 的时候没有被选中，那么在后续的枚举（假设是从小到大枚举 g 值）中一定不会被选中。

因为在上一轮中已经可以选择一些拥有更小 s 的边来连接使一些点连通，没有选中的边对于这些点的连通性不做贡献；而且可选的边会越来越多，肯定没有必要选当前没有选中的边，直接删掉这些边即可，这样可以保证每一轮所剩下的边只有最多 $n-1$ 条。时间复杂度降为 $\mathcal{O}(nm\log n)$。

继续优化。由于每次枚举 g 值时相对于上一轮只会多加入一条边，不用每次都重新排序然后跑 Kruskal，只需要在上一次得到的边集中按照插入排序的方式插入这一条边，并将这一轮得到的边集传递给下一轮。

时间复杂度 $\mathcal{O}(nm)$。

T. 播种之谣

简要题意

给定一个 $n \times m$ 的网格图，有空地，障碍，储能站三种。列车在空地上运行消耗 1 的代价，障碍不能通行，储能站能补充能源。若干询问，每次给定一个起点和一个终点，问使列车正常运行，热储罐的大小最小是多少。

一血：贾承羲
分享者：贾承羲
知识点：多源 bfs / 最小生成树 / kruskal 重构树。

T. 播种之谣

每一个询问的起点和终点都是储能站。把储能站看成点，列车一定只会在这 k 个点的最小生成树上运行。因为最小生成树是连通所有点，且用到的最大的边都尽可能小的体现。（可以回想 kruskal 算法就是按边权从小到大加边）

T. 播种之谣

每一个询问的起点和终点都是储能站。把储能站看成点，列车一定只会在这 k 个点的最小生成树上运行。因为最小生成树是连通所有点，且用到的最大的边都尽可能小的体现。（可以回想 kruskal 算法就是按边权从小到大加边）

怎么求出这棵最小生成树呢？如果知道了 k 个储能站两两之间的距离，那建树轻而易举，按照边权排序跑 kruskal 就行了。但 k 个储能站两两之间的距离相当于要以每个储能站为起点跑 bfs，每次 bfs 是 $O(nm)$ 的，跑 k 次不能接受。

T. 播种之谣

这个时候就可以引入多源 bfs 了。我们初始将 k 个储能站的坐标都塞进队列，同时以这 k 个点为起点做 bfs，遍历到一个点时，我们可以记录这个点离最近的储能站有多远，以及对应储能站的编号是多少。当遍历到一个曾经经过的点，就意味着两条路径相交了，我们取出这对点对应的储能站的编号，在新图上连一条边。

如此以来，我们就 $O(nm)$ 建出了一张新图，容易说明，我们所需的最小生成树就是这张新图的最小生成树。

T. 播种之谣

建好了最小生成树，接下来是回答询问。

每次询问，相当于问两个点，求它们在树上形成的简单路径中，边权最大值是多少。

这是一个经典题（NOIP 货车运输），主要有两种解决方式。一种是倍增，一种是 kruskal 重构树。两种方法回答单个询问的时间复杂度均为 $O(\log k)$。

T. 播种之谣

法一、倍增

预处理两个数组：

- fa[x][k]：节点 x 向上跳 2^k 步的祖先。
- mx[x][k]：节点 x 向上跳 2^k 步所经过的边的最大权值。

```
for (int i = 1; i <= __lg(dep[x]) + 1; i++) {
    fa[x][i] = fa[fa[x][i - 1]][i - 1];
    mx[x][i] = max(mx[x][i - 1], mx[fa[x][i - 1]][i - 1]);
}
```

T. 播种之谣

法一、倍增

查询时用和求解 LCA 类似的方法向上跳。

```
int dist(int x, int y) {
    int res = 0;
    if (dep[x] < dep[y]) {
        swap(x, y);
    }
    while (dep[x] > dep[y]) {
        res = max(res, mx[x][__lg(dep[x] - dep[y])]);
        x = fa[x][__lg(dep[x] - dep[y])];
    }
    return res;
}
int getmx(int x, int y) {
    int lca = LCA(x, y);
    return max(dist(x, lca), dist(y, lca));
}
```

T. 播种之谣

法二、kruskal 重构树
参阅：OI wiki - kruskal 重构树

kruskal 重构树具有如下重要性质：

- 原图中两个点间所有路径上的边最大权值的最小值 = 最小生成树上两点简单路径的边最大权值 = Kruskal 重构树上两点 LCA 的点权。

L. 星际争霸

简要题意

给出一个无向图，和 q 个询问，每次给出 s 个点，问存在几个点，使得这个点和他相连的边被去除后，这 s 个点中，至少存在一对点互不相通。

一血：幸子豪
分享者：幸子豪
知识点：圆方树。

L. 星际争霸

摧毁一个点使得两个选中的关键点不连通，首先被摧毁的点一定是原图中的割点，那我们可以先考虑用 tarjan 求出原图中的割点和点双连通分量，但如何快速的维护呢？这时候我们就需要引入圆方树这一概念。

L. 星际争霸

摧毁一个点使得两个选中的关键点不连通，首先被摧毁的点一定是原图中的割点，那我们可以先考虑用 tarjan 求出原图中的割点和点双连通分量，但如何快速的维护呢？这时候我们就需要引入圆方树这一概念。

在圆方树中，原来的每一个点是圆方树中的一个圆点，将点双连通分量对着圆方树中的一个方点。

所以得到的圆方树一共有 $n+c$ 个点。

对于图中的方点，其向对应的点双连通分量中的每一个点连边，这样每个点双会形成一个菊花图，菊花图之间通过原图中的割点相连。

L. 星际争霸

回到本题，我们可以将问题转化成，圆方树上包含所有关键点的最少点数联通块的圆点数减去关键点的数量。

那么现在问题就转化成了求圆点的个数，我们可以将圆点权值设为 1，方点权值设为 0，将点权放到这个点与其父节点的边上。

然后将关键点集合中的点按 DFS 序排序，计算排序后相邻两点的距离和，答案即为距离和的一半。此处需要注意首尾关键点是否是圆点的细节。

I. 日野春幸的烦恼

题意

有 n 个人，其中有些人之间有仇恨关系，这些人不能相邻而坐。当且仅当，有奇数且大于 1 个人且有一种坐法使得所有人可以围着一个圆桌而坐，即相邻而坐的人不能有仇恨关系，这些人才能开一场会议。求有多少人不能参加任何一场会议。

一血：蒋帅泉
分享者：蒋帅泉
知识点：点双连通分量 / 二分图。

I. 日野春幸的烦恼

我们不去想哪些人不能坐到一起，而去想哪些人能够坐到一起。换句话说，我们先建一个补图。那么开会的一种坐法即为一个奇环。问题转化为输出有多少个人，不在任何一个奇环中。

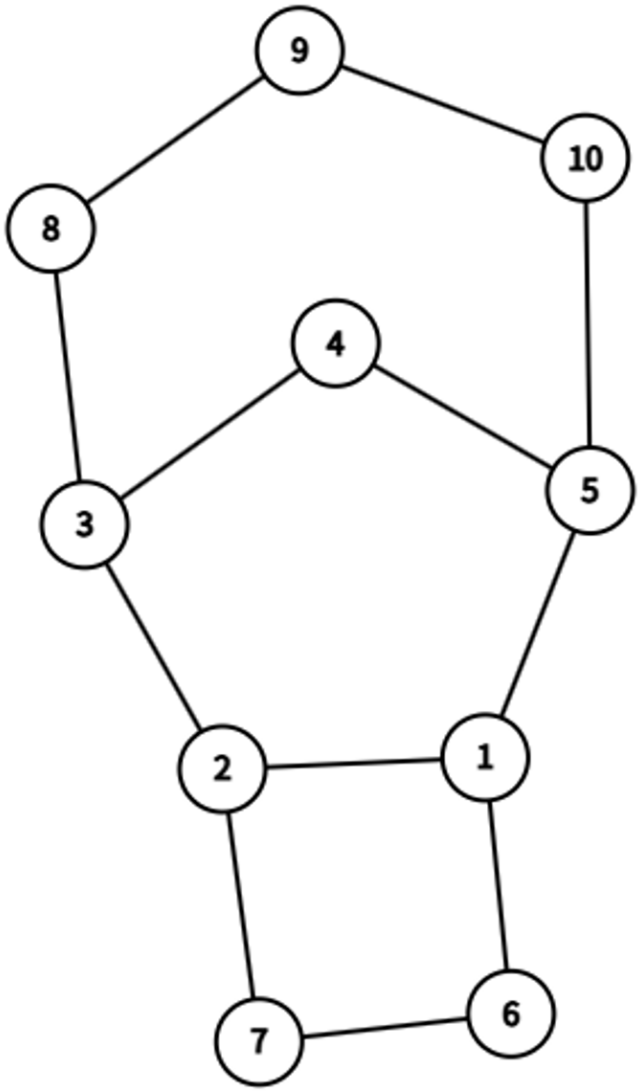
结论：点双连通图中若包含一个奇环，则所有点都在至少一个简单奇环上。

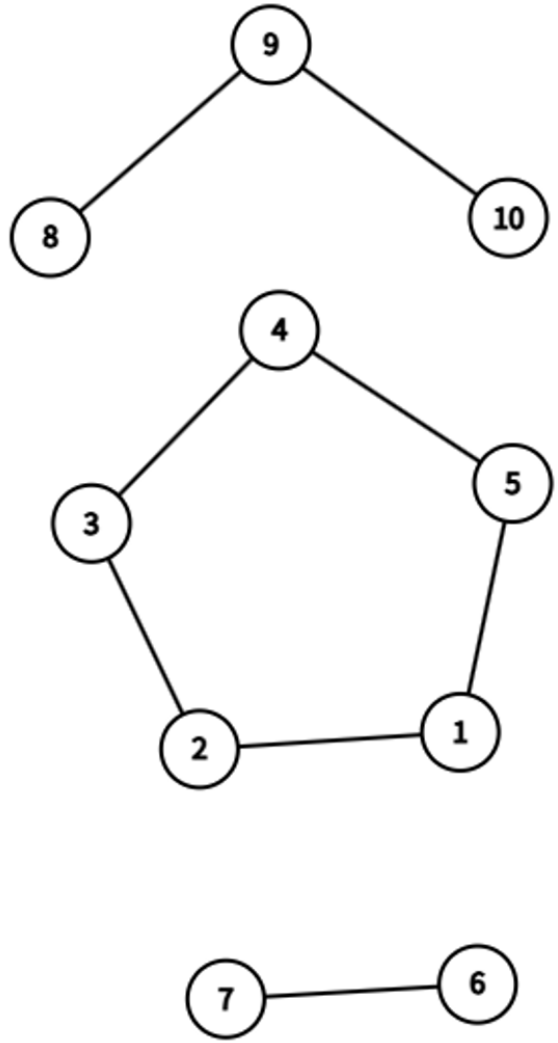
I. 日野春幸的烦恼

结论：点双连通图中若包含一个奇环，则所有点都在至少一个简单奇环上。

- 假设点双中有一个奇环，首先位于奇环上的点一定在奇环中。
- 原来图中，我们扣掉这个奇环。剩下许多连通分量。
- 假设一个连通分量为 H，H 一定与奇环上至少两个点相连。假设 H 不与奇环相连，则该图不连通，所以 H 与奇环相连；假设 H 只与奇环上的一个点相连，则这个点是割点，则该图不是点双图，证得 H 一定与奇环上至少两个点相连。
- 因为 H 连通，所以我们可以找到一条路径，将 H 上的点全都经过一遍。假设 H 与奇环相连的点为 u, v。在奇环上从 $u \to v$ 有两条路径，一条路径长度为奇数，一条路径长度为偶数。如果遍历 H 的路径长度为奇数，那么我们从 $u \to v$ 长度为偶数的路走以组成一个奇环；如果遍历 H 的路径长度为偶数，那么我们从 $u \to v$ 长度为奇数的路走以组成一个奇环。所以 H 上的点在奇环上。

I. 日野春幸的烦恼





I. 日野春幸的烦恼

让我们看上述例子，我们在这个图选取的奇环为 $1-2-3-4-5$。对于 $8-9-10$ 这条链，我们知道在没有去除奇环连的边时，它连接着 3，5 这两个点。我们假设它连接 u, v，且经过上述证明 $u \neq v$。$u-8-9-10-v$ 连成了一个长度为 4 的路径，即偶路径。因为 u, v 在奇环上，所以 u, v 之间必定有一条奇路径，一条偶路径。对应上图 $u=3$，$v=5$，奇路径为 $3-2-1-5$，偶路径为 $3-4-5$。最后 $3-8-9-10-5-1-2$ 组成一个奇环，证明 8，9，10 在奇环上。剩下的 6，7 同理。

I. 日野春幸的烦恼

结论：点双连通图中若包含一个奇环，则所有点都在至少一个简单奇环上。

有了上述结论，我们就可以将原图的点双连通分量求出来，再求出每个点双连通分量中是否有奇环，如果有，则标记该分量中所有点。最后没有被标记的点即是答案。求奇环用二分图染色。

时间复杂度为 $O(n^2)$。

简要题意

给你一个 n 个点，m 条边的带权有向图，让你求出起点到终点的第 k 短路长度。

本题不计一血。
知识点：k 短路，可持久化可并堆。

V. A*

虽然 A* 算法死了，但这里还是简单介绍下如何使用 A* 做法以及他怎么死的。

A* 算法可以更加高效地求解一些最短路径问题对每种状态记录三个函数 $f(n) = g(n) + h(n)$。

- $g(n)$ 是起始状态 S 到达当前状态 n 的代价。
- $h(n)$ 是当前状态 n 到达终止状态 T 的代价。
- $f(n)$ 是二者之和，代表初始状态 S 经由当前状态 n 到达终止状态 T 的估计代价。

估价函数 $h(n)$ 选的越准确，算法效率越高。

选大了，跑得慢。
选小了，可能漏掉最优解。

V. A*

本题中，估价函数直接设置为起点到 x 的最短路长度即可，其中 x 为状态 n 所在的点，该估价函数是绝对准确的。

开始我们将初始状态 $(s, 0)$ 加入优先队列，每次取出估价函数 $f(x)$ 最小的一个状态，枚举该状态所有的出边，将所有后继状态推入优先队列中。

某个点 x 被取出的第 j 次，就是起点到 x 的第 j 短路。

当某个点从优先队列中取出超过 k 次时，不需要再推入后继状态了，因为它不会影响答案。

最坏的情况下是 $O(nk \log n)$ 的，构造一个大环，每轮都要跑至少 n 条边，A* 就寄了。

假设整个图的边集为 G。

首先建出以终点 T 为根的，沿反向边跑的最短路树，设这些边构成了边集 T。

那么每个点沿着树边走到 T，它对于答案的贡献为 0。

我们加入一条非树边，它对答案产生 $delta_e = dis_v + w_e - dis_u$ 的贡献，即如果某条路径经过了这条边，它比起最短路的长度增加 $delta_e$。

一条路径 p 的总长度是 $\sum_{e \in p-G} delta_e$。

现在要求出前 K 条总长度最小的路径长度，也就是找出第 k 大的非树边集！

V. A*

首先向一个堆中加入一个空的边集，然后拓展：每次从堆中取出一条最小权值的边集 p，设该边集最后一条边为 (u, v)，有两种情况：

1. 替换末尾边为一条刚好大于等于它的非树边。
2. 尾部接上一条起点为 v 在 T 中祖先 (包括自己) 连出去的所有非树边的最小边。

这种方法能把所有可能的边集按从小到大的顺序枚举到，每次要么替换最后一个元素为它的后继，要么添加一个可选的最小元素。

操作 2 需要找出可选的最小拓展边，可选的边可能很多，想办法用数据结构维护。

在最短路树上每个点都维护一个非树边集合，操作 2 需要子节点包含父节点的集合，联想到可持久化。

V. A*

操作 1，需要一个有序表来找后继，要用到堆！

可持久化可并堆！

merge 里每次合并都新建节点。

操作 1 就是把最后一条边换成这条边在堆中的左右儿子。

操作 2 直接取堆顶。

时间复杂度 $O(k \log n + n \log n)$。

E. 踩冰块

简要题意

在一个 $n \times m$ 的棋盘上进行博弈，求先手必胜的起始点有哪些。

一血：滕鎧泽
分享者：滕鎧泽
知识点：二分图博弈。

E. 踩冰块

把每个非 # 的格子当成点，共用边的格子所代表的点之间连一条边，发现这是个二分图。

二分图博弈：在二分图上某点放置一枚棋子，双方轮流移动棋子，且不能将棋子移动到已经走过的点。

对于二分图博弈，有如下的结论：

- 先手必胜当且仅当起始点一定会被包含在最大匹配中，否则后手只要一直走匹配边就赢了。

E. 踩冰块

证明如下：

设二分图分为左右两个集合，初始点 H 在左集合中。

① 假设 H 必定在最大匹配中，那么一定不存在在右集合中且在最大匹配中的的点，有连向在左集合中除最大匹配中它所连接的一个或两个点的其他点的边。

不然要么 H 不是必选（可以用这个其他点替换掉 H 成为与之等价的最大匹配，和假设必定矛盾）；

要么当前匹配不是最大匹配，（可以用这个其他点把 H 解放出来，得到更大的匹配，和假设为最大匹配矛盾）。

E. 踩冰块

由上述结论可得，若假设成立，先手选了一个匹配点后，后手必须要选最大匹配内的匹配点，即选择是确定的。

走出第一步，相当于左集合去掉 H，分成选择两部分选择，其中一部分剩余左右集合相等，另一部分右集合比左集合多 1，选择第二个选法，就能保证最后选到是右集合，此时获胜。

则 H 在必定在最大匹配中时，必胜。

E. 踩冰块

② 假设 H 不必定在最大匹配中，那么先手无论选哪个点，都一定在最大匹配中。

设 P 和 H 相连，却不在最大匹配中，$P-H$ 就构成了新的匹配，原匹配就不是最大匹配了。

后面就是先后手交换，转化为第一种情况，从一个必定在最大匹配中的点出发，所以全局的先手是必败的。

结论得证。

E. 踩冰块

问题转化为如何在给定二分图中找到所有" 不一定在最大匹配上" 的点。

我们把在所有最大匹配上的点称为 A 类点，反之称为 B 类点。

我们可以先用任意二分图匹配算法求出一个最大匹配，此时不在这个最大匹配上的点一定是 B 类点。

要想求出所有 B 类点，我们可以从任意一个 B 类点（设为 s）出发 dfs，寻找与它相邻的 A 类点（设为 t），那么此时 t 的所有匹配点均为 B 类点。因为假设一个最大匹配是 $t-u$，那么把 u 替换成 $t-s$，一样能行，即 u 不一定在最大匹配上。

按照该方法不断 dfs 下去，找到所有 B 类点即可。

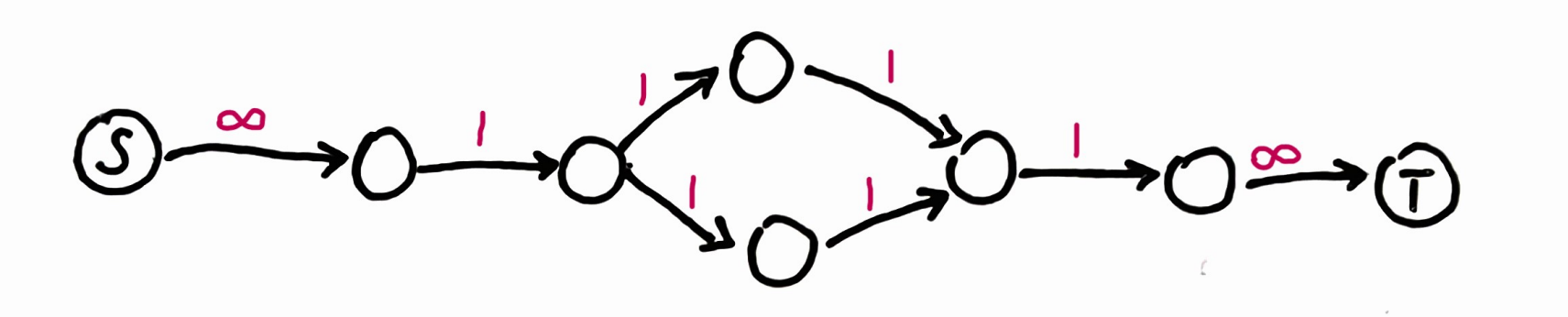
简要题意

给定一个 n 个点 m 条边的无向图，初始所有边的边权为 0，你可以将 k 条边的边权变为 1，问从 1 到 n 的最短路最大是多少。

一血：丁柏宇
分享者：丁柏宇
知识点：分层图 / 最小割。

二分答案，设二分出的答案是 d，那么需要判定让 $1 \to n$ 的最短路为 d 的最小操作次数是否 $\leq k$。

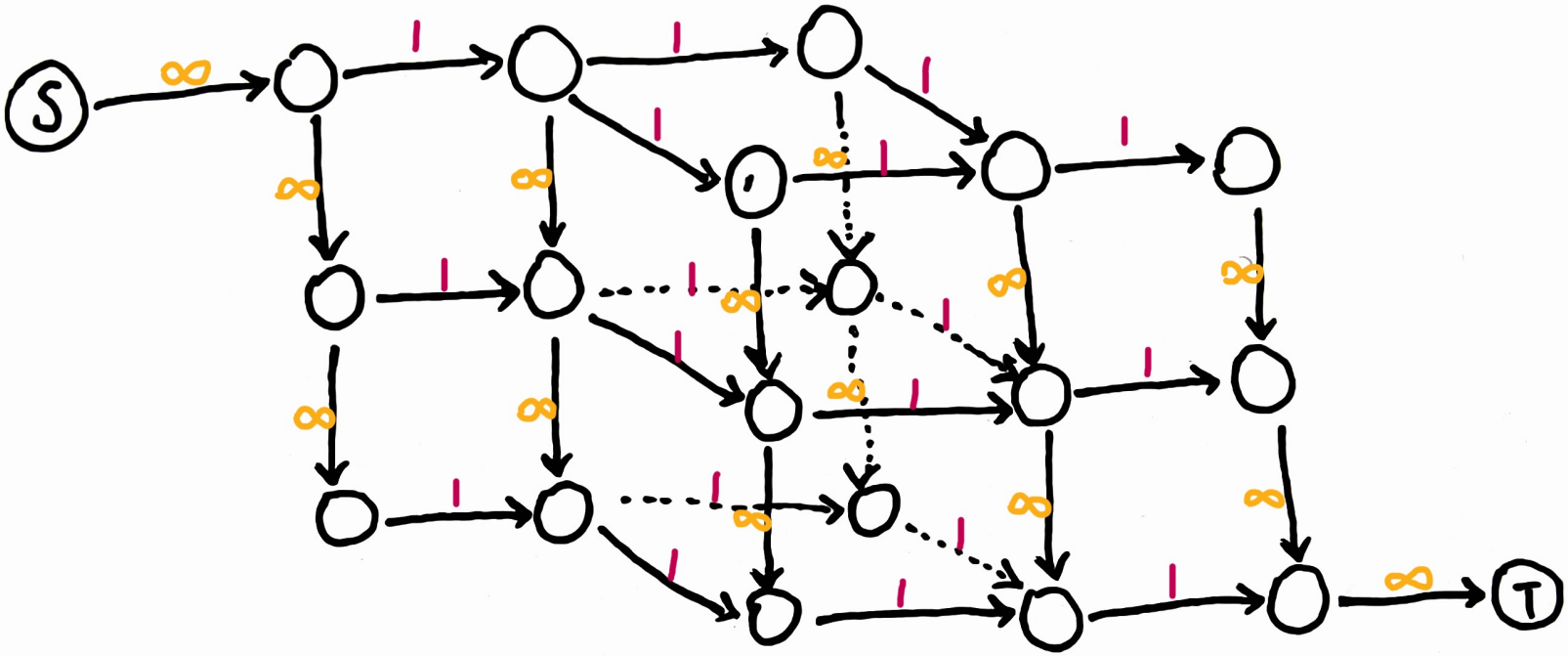
当 $d = 1$ 时，问题等价于选定一个最小边集，使得任何一条从 $1 \to n$ 的路径都无可避免地经过这个边集中的边。换句话说，边集中的边将原图分割成了两部分，即最小割。



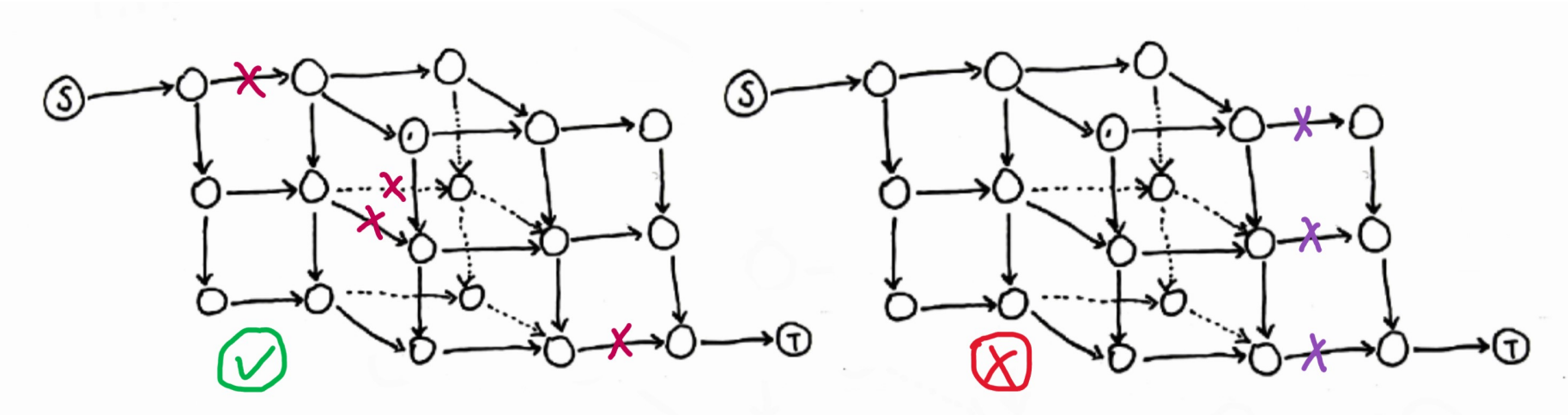
标注出的数字代表流量。

当 $d > 1$ 时，我们能通过建模将问题转化为一个分层图最小割。

考虑同样的图，但 $d = 3$：

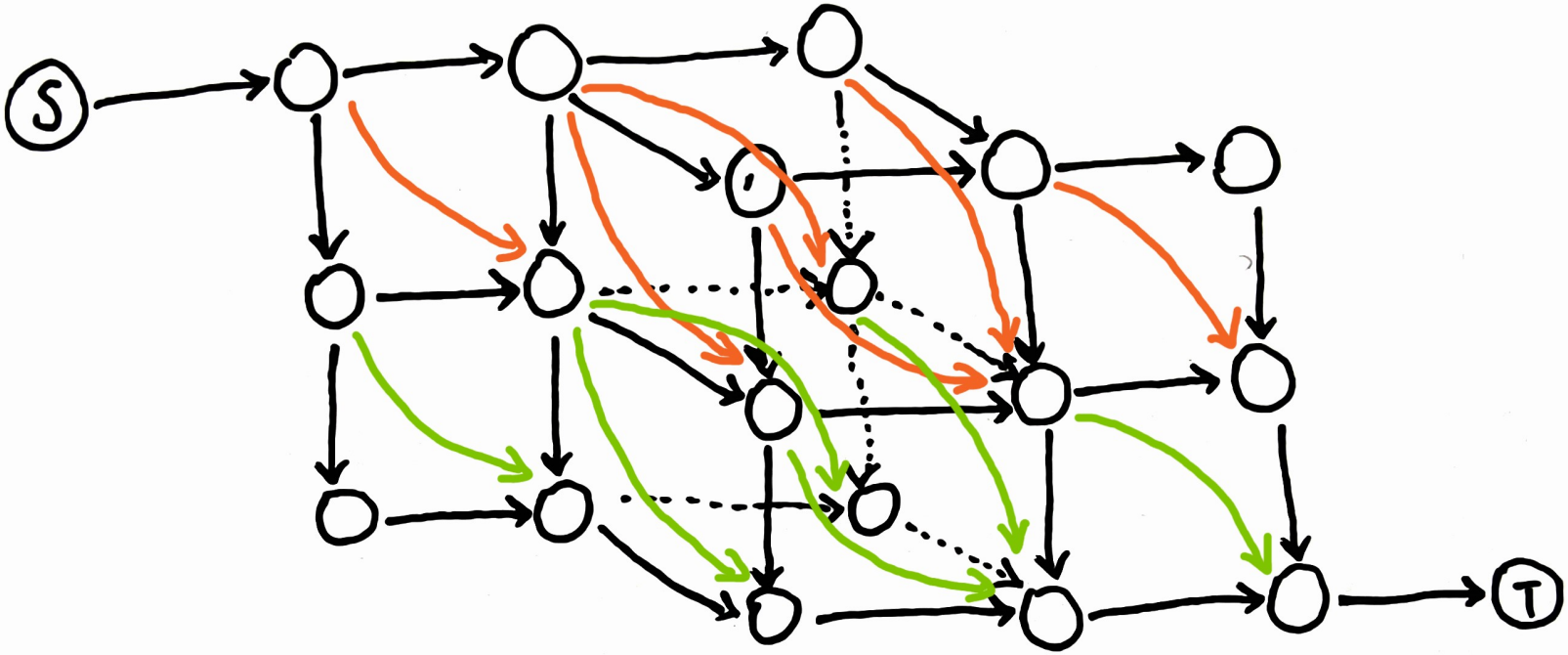


这只是一个分层图的框架，当我们需要让 $1 \to n$ 的最短路为 3 时，会出现一些问题。



左图是我们期望的：分层图的每一层都选定了一些边，且每一层选定的边都组成了原图的一个最小割。把它们割掉，原图不连通，这样一个 d 层的分层图的最小割就是让 $1 \to n$ 的最短路为 d 的最小操作次数。但能这么做的前提是每一层选定的边集互不相同，右图中我们只选定了三条边就能让图不连通，显然不合法。

W. Ciallo



解决方案是对于每一条存在于原图中的边 (u, v)，$\forall i \in [1, d)$，从第 i 层的 u 向第 $i+1$ 层的 v 连边。这样选中不同层的同一条边就没有了意义。

时间复杂度 $O(\mathsf{MaxFlow}(n^2, mn + n^2) \log n)$。

S. 唱跳 rap,?

简要题意

将 n 个球跟 m 个筐配对，要求每个球跟一个筐配对并且每个筐配对的球个数不超过 3，且配对数量小于等于 1 的筐数目最大。

一血：丁柏宇
分享者：丁柏宇
知识点：一般图匹配。

S. 唱跳 rap,?

首先每个筐可以放至多三个球，可以考虑把筐拆成 3 个点，那么题中的条件就变成了球代表的点向这三个点各连一条边并且筐对应的 3 个点互相连边。

- 如果跟球匹配为 0，那么对答案贡献为 1，筐子内部的 3 个点的最大匹配为 1。
- 如果跟球匹配为 1，那么对答案贡献为 1，此时筐子内部 3 个点和该球的最大匹配为 2，减去球的贡献 1 即可。
- 如果跟球匹配为 2，那么对答案的贡献为 2，此时筐内部无贡献，这时候减去球的贡献 2 即可。
- 如果跟球的匹配为 3，对答案的贡献也为 3，也需要减去球的贡献 3。

因此答案即为这个新图的最大匹配减去球的个数。

S. 唱跳 rap,?

但该图不是二分图 (不能黑白染色)，如何处理？

我们有带花树这个算法可以求解一般图上的匹配问题。

考虑一下二分图和一般图的最大区别（或者说唯一的区别在哪里)?

二分图没有奇环（也就是长度为奇数的环)，而一般图是可以有的。

所以匈牙利算法中的寻找增广路然后路径取反的方法在一般图上就不适用了。

主要还是要解决奇环的问题。

S. 唱跳 rap,?

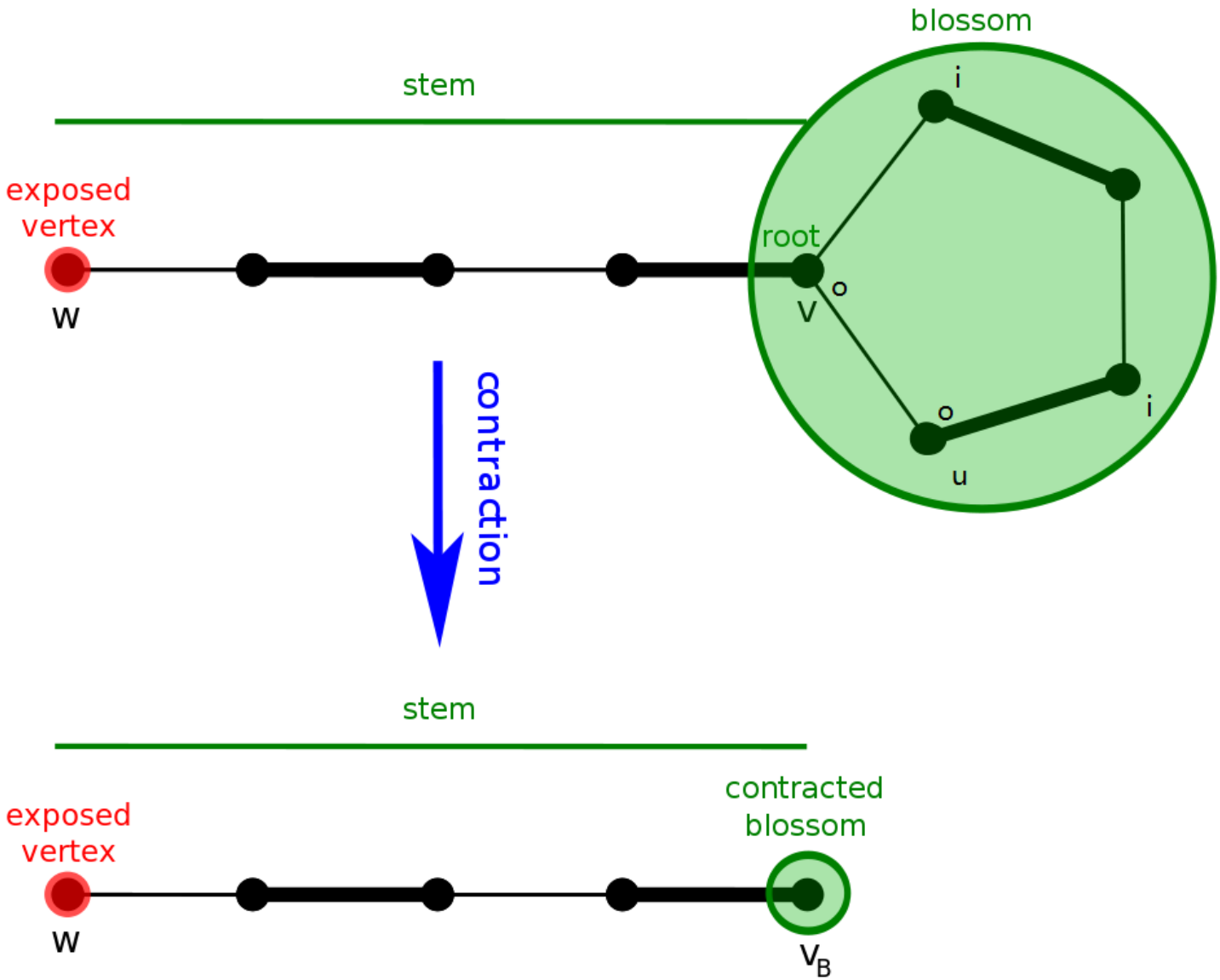
从二分图的角度出发，每次枚举一个未匹配点，设出发点为根，标记为「o」，接下来交错标记「o」和「i」，不难发现「i」到「o」这段边是匹配边。

假设当前点是 v，相邻点为 u，可以分为以下两种情况：

1. u 未拜访过，当 u 是未匹配点，则找到增广路径，否则从 u 的配偶找增广路。
2. u 已拜访过，遇到标记「o」代表需要缩花，否则代表遇到偶环，跳过。

遇到偶环的情况，将他视为二分图解决，故可忽略。缩花后，再新图中继续找增广路。

S. 唱跳 rap,?

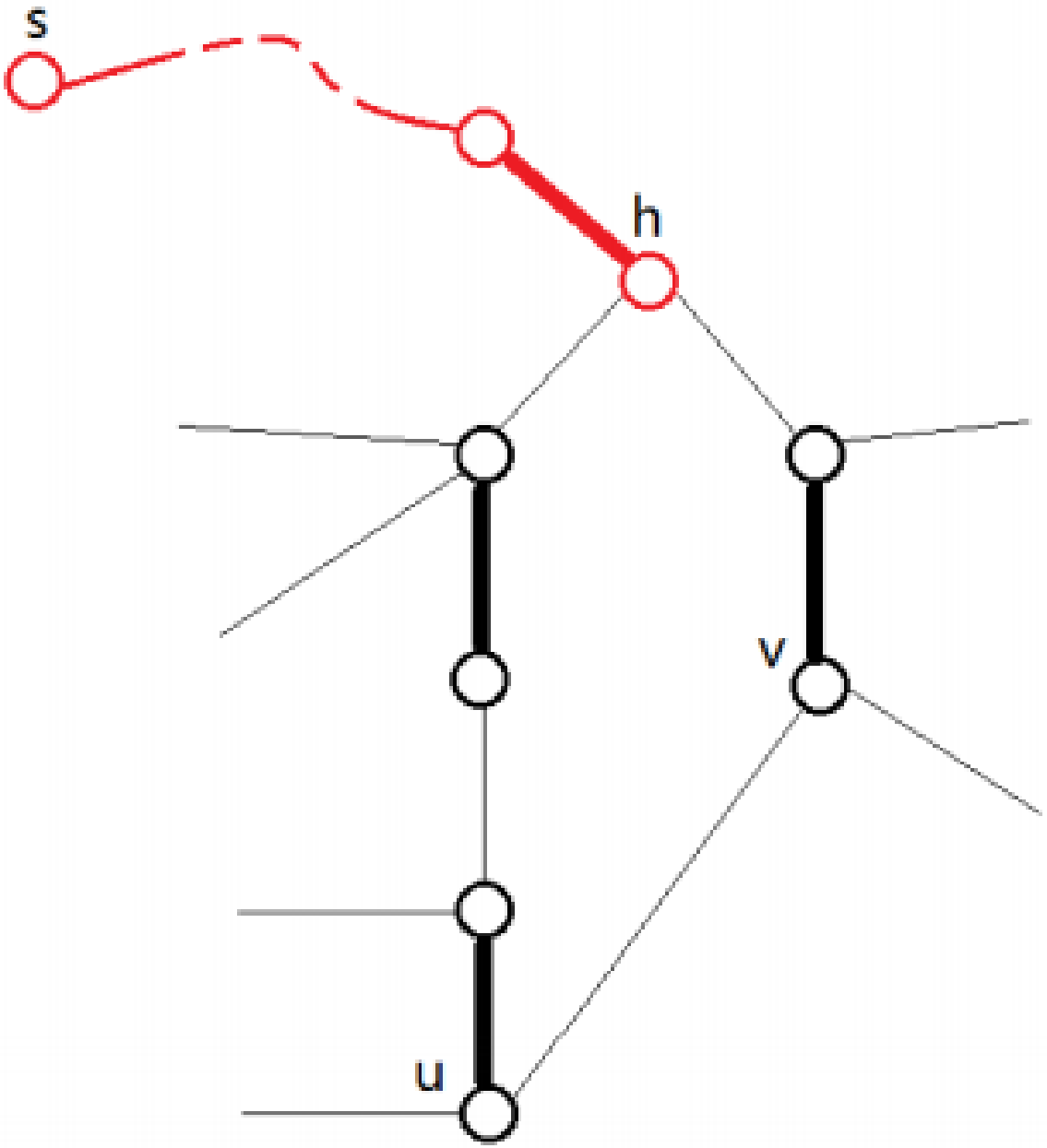

blossom
stem
i
exposed
vertex
root
o
V
w
i
o
u
contraction
stem
contracted
blossom
exposed
vertex
w
V_B

S. 唱跳 rap,?

设原图为 G，缩花后的图为 G，我们只需要证明：

1. 若 G 存在增广路，G' 也存在。
2. 若 G' 存在增广路，G 也存在。

S. 唱跳 rap,?

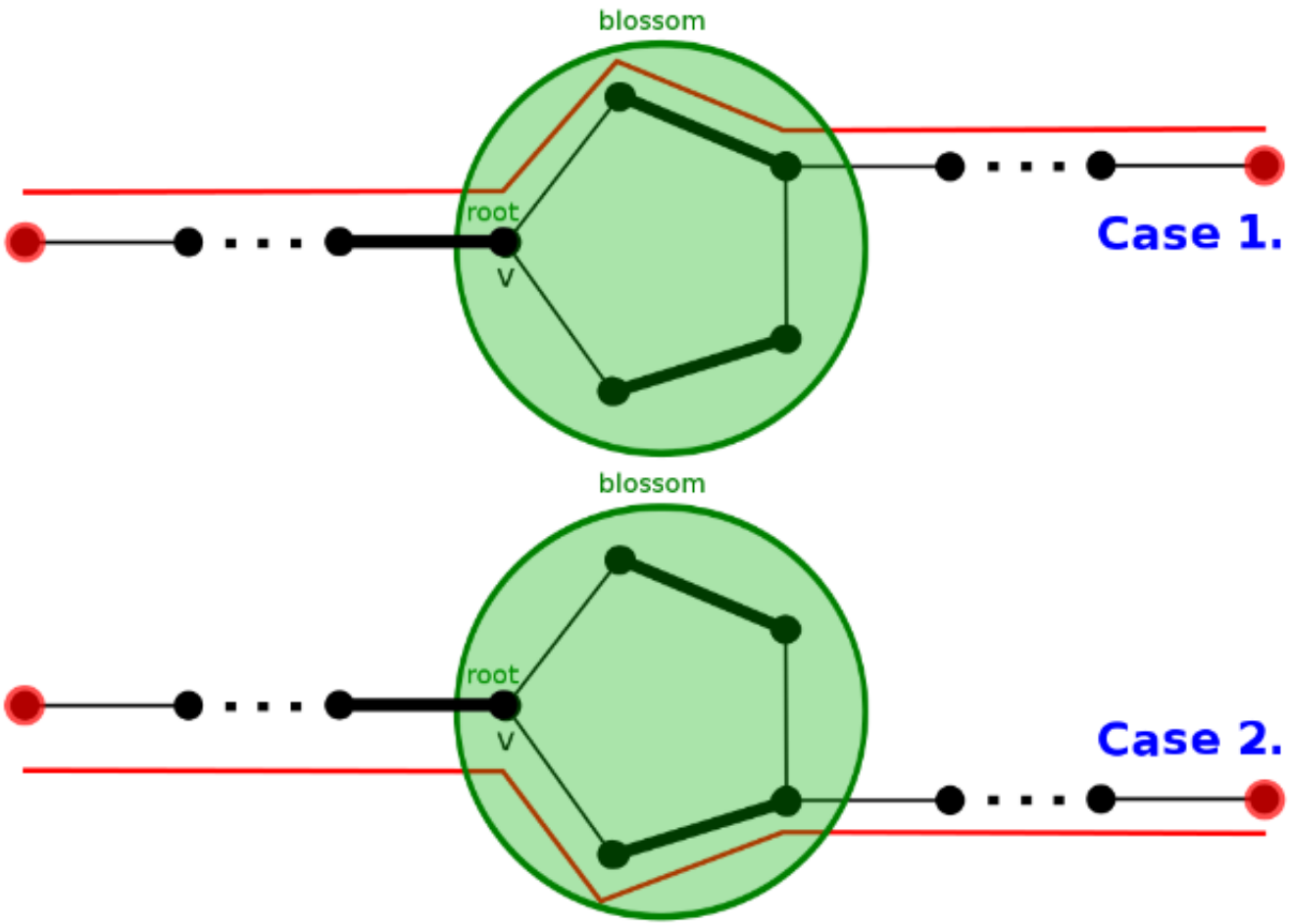


S. 唱跳 rap,?

设非树边（形成环的那条边）为 (u, v)，定义花根 $h = lca(u, v)$。奇环是交替的，有且仅有 h 的两条邻边类型相同，都是非匹配边。那么进入 h 的树边肯定是匹配边，环上除了 h 以外其他点往环外的边都是非匹配边。

观察可知，从环外的边出去有两种情况，顺时针或逆时针。

S. 唱跳 rap,?



G. 简单可满足性问题（二周目）

简要题意

给定一个 n 个变量，m 个子句的 CNF，满足每个正变量和反变量都至多出现一次。问是否存在一种变量赋值方案，使得每个子句中有且仅有一个文字的取值为 1。

一血：幸子豪
分享者：张诗杨
知识点：费用流。

G. 简单可满足性问题（二周目）

我们将每个变量 x_i 拆成 3 个点：x_i，x_i^+，x_i^-，一共得到 $3n$ 个点。

从源点 S 向每个 x_i 连一条容量为 1 的边，再从 x_i 向 x_i^+ 和 x_i^- 分别连一条容量为 1 的边，流量限制使得正文字和负文字只能有一个为真。

现在，我们再新建 m 个点（记为 p_1，p_2，$\ldots$，p_m），对应 m 个子句。如果变量 x_i 出现在子句 C_j 中，分两种情况：

1. 变量 x_i 以正文字形式在子句 C_j 中出现，此时由 x_i^+ 向 p_j 连容量为 1 的边。表示 $x_i = 1$ 会使子句 C_j 中的文字 x_i 为真。
2. 变量 x_i 以负文字形式在子句 C_j 中出现，此时由 x_i^- 向 p_j 连容量为 1 的边。表示 $x_i = 0$ 会使子句 C_j 中的文字 $\overline{x_i}$ 为真。

G. 简单可满足性问题（二周目）

最后，从每个子句对应的点 p_i 向汇点 T 连容量为 1 的边，这样保证了每个子句有且仅有一个文字为真。

此时跑一遍最大流，你可能会认为满流即是胜利，实则并不然。

因为如果对于一个变量 x_i，它即有正文字，又有负文字（即正文字和负文字都在该 CNF 中出现了），那么 $S \to x_i$ 这条边的流量必须为 1。我们将这类 x_i 称为特殊变量。

问题转化为如何让所有特殊变量满足限制。

G. 简单可满足性问题（二周目）

一种方法是对于所有特殊变量 x_i，在 $S \to x_i$ 这条边上加上 -1 的费用，使得该条边优先被流。

除这些边外，其它边费用设为 0，跑一遍最小费用最大流。

这样该 CNF 满足条件当且仅当满流且费用的绝对值等于特殊变量的个数。

遍历残余网络即可知道方案。

时间复杂度 $O(\mathsf{MaxFlowMinCost}(3n+m, 6n))$。

U. 离心率

简要题意

定义任意一个点 u 的离心率 $\epsilon(u) = \max\limits_{v \in V} dist(u, v)$。给定一张 n 个点 m 条边且每个点至多在一个环上的无向连通图，对每个点求离心率。

一血：鸣鸣鸣没人做
分享者：鸣鸣鸣没人做
知识点：基环树 / 换根。

U. 离心率

我们任选一个环，钦定他为整个图的「根」。同样约定 x 在 y 的子树中当且仅当 y 在从根到 x 的路径上。

定义 $F[x] = \max\limits_{y} dis(x, y)$，其中 y 在 x 的子树内。对于根来说，我们从根的每个点出发向子树内 bfs，可以 $O(n)$ 得到根上每个点的 $F[x]$。

U. 离心率

令 $Cir[x]$ 代表 x 所在的环，特别地，单点看作一个环。

定义 $G[x]$ 代表从 x 出发向 $Cir[x]$ 方向延伸出的最长距离，那么：

$$G[u] = \max_{v \in Cir[u]} (dis(u, v) + F[v])$$

这是一个典型的使用单调队列解决的式子。

具体地，拆环成链，单调队列维护最大的 $F[i] - i$，窗口大小不超过 $\frac{1}{2}$ 环长，顺时针逆时针扫两遍。

U. 离心率

现在，在知道一个环所有 $F[x]$ 的情况下，可以线性求出 $G[x]$。而对于每个点 x，答案就是 $\max(F[x], G[x])$。于是问题转化为如何把 F 从根转移到图上的其它点。

这依然是典型的换根问题，每次转移时的 UpValue 选择非该当前子树的最大深度，并与当前点的 G 取 $\max$。

时间复杂度 $O(n)$。

实现时有若干细节等待你的探索:)

R. 时间线交织理论

简要题意

给一个 DAG，求最大权反链，并输出一组方案。

一血：幸子豪
分享者：幸子豪
知识点：Dilworth 定理 / 上下界网络流 / 拆点 / 最小割 / 有源汇最小流。

R. 时间线交织理论

根据 Dilworth 定理，最长反链 = 最小链覆盖。

但这里点有权值，即求的是最大权反链，我们同样能对应到带权最小路径覆盖问题。

具体的，相当于给定了一些带权的路径，要满足经过每个点的路径的权值和不小于它的点权，然后最小化所有路径的权值和。

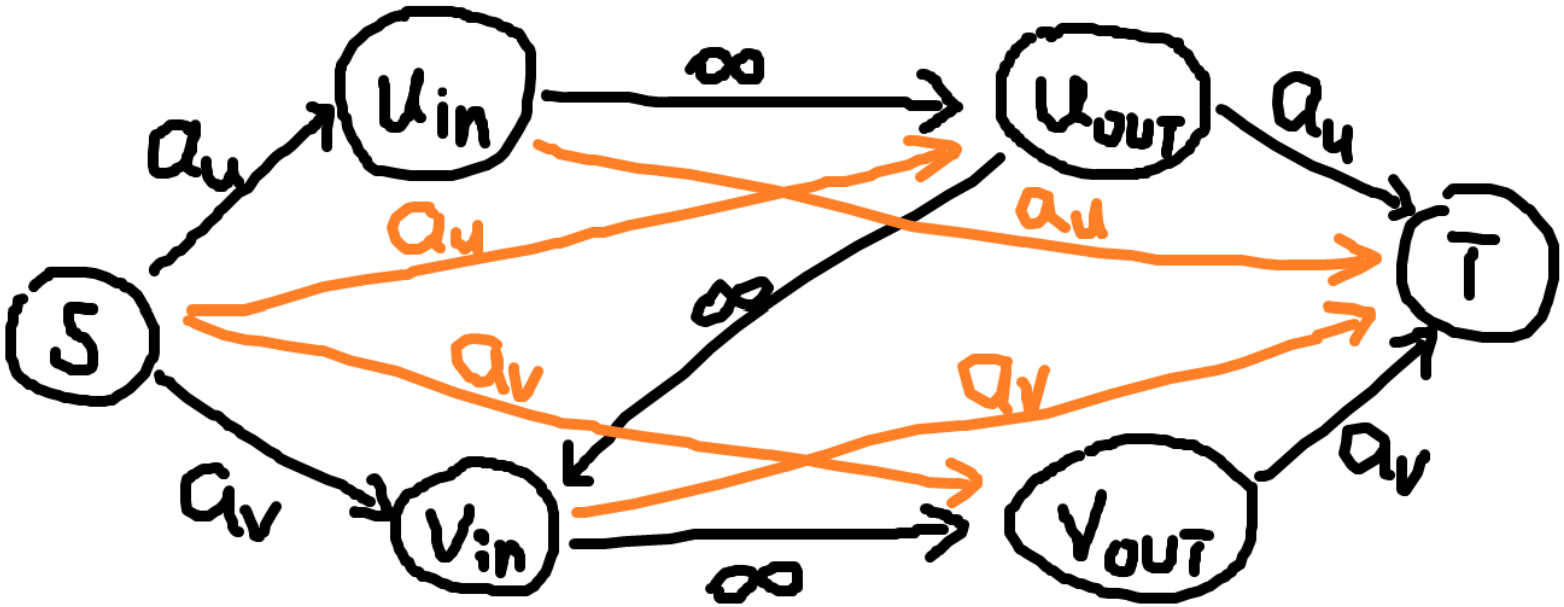
R. 时间线交织理论

用拆点解决点权问题：每个点 u 拆成入点 u_{in} 和出点 u_{out}，中间连一条下界为点权，上界正无穷的边。

接着再从源点 S 向所有 u_{in} 连边，所有 u_{out} 向汇点 T 连边，转换成一个有源汇最小流问题。

R. 时间线交织理论

按照上下界网络流的做法，将下界转换为辅助边，如下图：

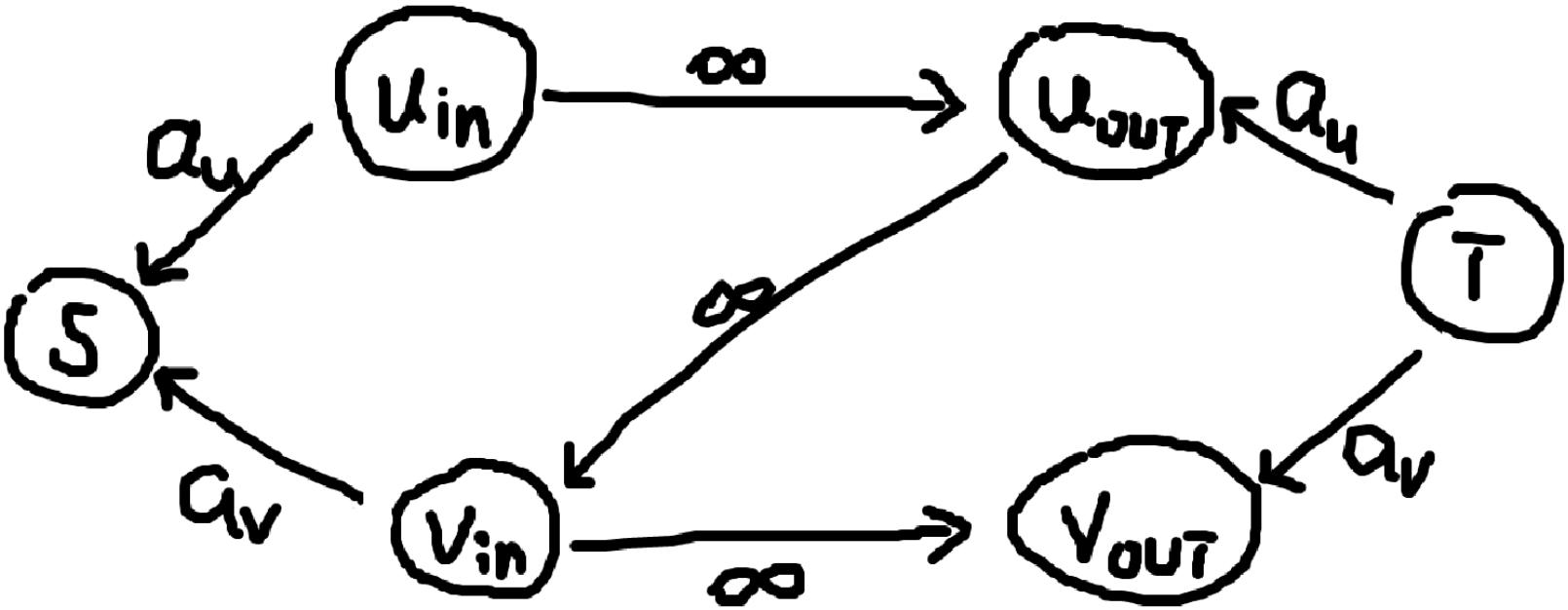


上图展示的是原图中有一条边 $u \to v$ 的情况。

我们惊奇的发现，这个图的最大流可以瞪眼瞪出来。对于每一个 u，有 $S \to u_{out} \to T$ 和 $S \to u_{in} \to T$ 两条满流路径，减去辅助边的流量后，最大流是 $\sum a_i$。

R. 时间线交织理论

接着因为要求最小流，所以从汇点 T 开始退流。注意到原图每一条边都流满，所以等价于对下面这幅图求最大流。



用 $\sum a_i$ 减去最大流就是答案。

R. 时间线交织理论

考虑如何输出方案。实际上退流时的模型也可以看作最小割，输出一组最小割的方案即可。

换一个角度想，退流相当于去掉一些不能选的点，而此时最小割相当于用最少的代价求出一组合法方案，对应到原题就是最优方案。

时间复杂度 $O(\mathsf{MaxFlow}(n, m))$。

没了。
要好好补题哦。
负责人天天盯着评测记录，如果你补题了，他会很开心的。
如对题解仍抱有疑问，欢迎私聊图论专题负责人。
预计图论专题完全结束后会放出 std。
感谢大家的支持！