图论专题第二次讲座

UESTC-XCPC-2024-暑假前集训

UESTC XCPC 集训队

2024 年 5 月 18 日

Overview

本次专题一共有二十六道题目，其中有二十四道题计入一血。

Overview

本次专题一共有二十六道题目，其中有二十四道题计入一血。

截至今日早上十点：

- 一共有 81 名同学报名参加本次专题，有 60 名同学通过了其中至少一道题目

Overview

本次专题一共有二十六道题目，其中有二十四道题计入一血。

截至今日早上十点：

- 一共有 81 名同学报名参加本次专题，有 60 名同学通过了其中至少一道题目
- 一共有 11 名同学抢到了一血

Overview

本次专题一共有二十六道题目，其中有二十四道题计入一血。

截至今日早上十点：

- 一共有 81 名同学报名参加本次专题，有 60 名同学通过了其中至少一道题目
- 一共有 11 名同学抢到了一血 （其中 yty 同学抢了六道题的一血，Orz 古希腊掌管流的神）

Overview

本次专题一共有二十六道题目，其中有二十四道题计入一血。

截至今日早上十点：

- 一共有 81 名同学报名参加本次专题，有 60 名同学通过了其中至少一道题目
- 一共有 11 名同学抢到了一血 （其中 yty 同学抢了六道题的一血，Orz 古希腊掌管流的神）
- 本次专题通过人数最高的题目是 H 题，一共有 50 人（包括打星）通过这道题目

A - 用户管理系统

题目大意

给你一个有根树，判断一个点是否为另一个点的父亲。

A - 用户管理系统

题目大意

给你一个有根树，判断一个点是否为另一个点的父亲。

- 本题不计一血
- 知识点：LCA

A - 用户管理系统

一个显然的想法就是求出两个点的 LCA，判断 LCA 是否为询问的点。

常见的 求 LCA 算法 有：倍增，Tarjan，欧拉序（DFS 序）+ RMQ，树链剖分……

以上尽量都掌握，特别推荐 DFS 序 + RMQ。

A - 用户管理系统

一个显然的想法就是求出两个点的 LCA，判断 LCA 是否为询问的点。

常见的 求 LCA 算法 有：倍增，Tarjan，欧拉序（DFS 序）+ RMQ，树链剖分……

以上尽量都掌握，特别推荐 DFS 序 + RMQ。

但其实显然也有 $O(n)$ 的做法：

考虑 DFS 序，另一个点 u 的 DFS 序为 $\texttt{dfn}_u$，点 u 的子树大小为 $\texttt{siz}_u$。

则只需要考虑 $\texttt{dfn}_u \leq \texttt{dfn}_v < \texttt{dfn}_u + \texttt{siz}_u$ 是否满足即可。

G - 模拟宇宙 · 其一

题目大意

题目大意：给定一张 DAG ，求以各个点为起点的最长路径长度。

G - 模拟宇宙 · 其一

题目大意

题目大意：给定一张 DAG，求以各个点为起点的最长路径长度。

- 一血：陈治臻
- 分享者：吕东阳
- 知识点：拓扑排序

G - 模拟宇宙 · 其一

我们可以发现每个点的答案可完全由其后继节点决定，因此可以使用反向存储边后用拓扑排序的方式求得各个点的答案。

在原图中，出度为 0 的节点答案显然为 1 ，而其他所有节点的答案则是其后继节点中答案的最大值加一。于是我们从出度为 0 的节点开始，更新其所有前驱节点的答案后将其删去，然后再寻找出度为 0 的节点，重复上述流程，直到所有节点被删去。

但是这样我们需要从后继节点寻找前驱节点，因此我们可以反向存储边，然后按照拓扑排序的方式进行更新，

H - 救援行动

题目大意

给定 n 个点及任意两点之间的距离，要求找出最小生成树。

H - 救援行动

题目大意

给定 n 个点及任意两点之间的距离，要求找出最小生成树。

- 一血: 陈治臻
- 分享者: 龚锦鹏
- 知识点: 最小生成树

H - 救援行动

最小生成树模版题。在算法竞赛中，常用的最小生成树算法有

Prim 算法

从一个结点开始，不断加点，每次要选择距离最小的一个结点，以及用新的边更新其他结点的距离。

Kruskal 算法

从小到大的枚举边，用并查集判断这条边的两个端点是否连通，如果未连通，则把这条边加入最小生成树中。

M - 不要等到失去……

题目大意

给定 n 点，m 边无向图，求割点数量，点双连通分量数量，割边数量，边双连通分量数量。

M - 不要等到失去……

题目大意

给定 n 点，m 边无向图，求割点数量，点双连通分量数量，割边数量，边双连通分量数量。

- 本题不计一血
- 知识点：各种连通图

M - 不要等到失去……

Tarjan 算法的模版题。

相关资料：OI Wiki

P - 利萨斯陷落

题目大意

求字典序最小的无向图欧拉路径。

P - 利萨斯陷落

题目大意

求字典序最小的无向图欧拉路径。

- 一血：伍柏霖
- 分享者：幸旭鑫
- 知识点：无向图最小字典序欧拉路径

P - 利萨斯陷落

欧拉回路的定义

- 欧拉回路：通过图中每条边恰好一次的回路
- 欧拉通路：通过图中每条边恰好一次的通路
- 欧拉图：具有欧拉回路的图
- 半欧拉图：具有欧拉通路但不具有欧拉回路的图

P - 利萨斯陷落

欧拉回路的定义

- 欧拉回路：通过图中每条边恰好一次的回路
- 欧拉通路：通过图中每条边恰好一次的通路
- 欧拉图：具有欧拉回路的图
- 半欧拉图：具有欧拉通路但不具有欧拉回路的图

欧拉回路的判别

- 无向图是欧拉图当且仅当：
  - 非零度顶点是连通的
  - 顶点的度数都是偶数

P - 利萨斯陷落

欧拉回路的定义

- 欧拉回路：通过图中每条边恰好一次的回路
- 欧拉通路：通过图中每条边恰好一次的通路
- 欧拉图：具有欧拉回路的图
- 半欧拉图：具有欧拉通路但不具有欧拉回路的图

欧拉回路的判别

- 无向图是欧拉图当且仅当：
  - 非零度顶点是连通的
  - 顶点的度数都是偶数
- 无向图是半欧拉图当且仅当：
  - 非零度顶点是连通的
  - 恰有 2 个奇度顶点

P - 利萨斯陷落

而找欧拉路径（回路）的方法也很简单：

遍历 + stack 加入 + 倒序输出即可

```
void dfs(int u) {
    for (int &i = cur[u]; i < e[u].size();) {
        // 获取该边信息
        i++; // cur[u] 指向下一条边
        // 判断该边是否已被遍历，是则 continue
        // 标记该边以及反向便已被遍历
        dfs(v); // DFS 下个点
    }
    stk.push_back(u);
}
```

V - 客户

题目大意

按照给定的需求顺序，求最大匹配，一旦失配就停止。

V - 客户

题目大意

按照给定的需求顺序，求最大匹配，一旦失配就停止。

- 一血：龚锦鹏
- 分享者：龚锦鹏
- 知识点：二分图最大匹配

V - 客户

本题为匈牙利算法的模版题。

匈牙利算法大致思路：从右向左尝试寻找一个未被匹配的点，如果找到就配对，没找到，就尝试更改之前已配对的点，如果更改成功就可以完成配对，否则就不能配对（建议网上学习，如 OI Wiki）。

C - 神奇王国（easy）

题目大意

给定 n $(2 \leq n \leq 2 \times 10^3)$ 件物品，每个物品有两个对应的状态值。
设 d 表示当前状态下任意两个物品所处状态值差的绝对值的最小值。
找到 d 的最大值。

C - 神奇王国（easy）

题目大意

给定 n $(2 \leq n \leq 2 \times 10^3)$ 件物品，每个物品有两个对应的状态值。
设 d 表示当前状态下任意两个物品所处状态值差的绝对值的最小值。
找到 d 的最大值。

- 一血：许庆山
- 分享者：许庆山
- 知识点：2-SAT

C - 神奇王国（easy）

首先考虑二分答案，对于当前的 mid 采用 2-SAT 进行判断。

如何判断，只需搞清楚不同状态之间的关系。

C - 神奇王国（easy）

首先考虑二分答案，对于当前的 mid 采用 2-SAT 进行判断。

如何判断，只需搞清楚不同状态之间的关系。

我们称第 i 件物品的 x 和 y 两个状态互为反状态。

枚举每一个状态，在 mid 的限制下，如果选择了当前状态 val ，那么区间 $(\text{val}-\text{mid}, \text{val}+\text{mid})$ 这个区间中的任何一个状态都不能再选择，那么我们可以向位于这个区间中的所有状态的反状态建一条有向边，表示如果我选择了当前状态，那么这些区间中状态的反状态也必须被选择。

C - 神奇王国（easy）

枚举每一个状态，在 mid 的限制下，如果选择了当前状态 val ，那么区间 $(\text{val} - \text{mid}, \text{val} + \text{mid})$ 这个区间中的任何一个状态都不能再选择，那么我们可以向位于这个区间中的所有状态的反状态建一条有向边，表示如果我选择了当前状态，那么这些区间中状态的反状态也必须被选择。

建完图之后，对整张图进行缩点求强连通分量，不难发现如果两个反状态出现在了一个强连通分量里，意味着这两个状态必须同时出现，这就与它们两个状态互为反状态相互矛盾，此时我们认为当前枚举的答案不合理。

如果经过 2-SAT 判断为 `true`，则增加 mid，否则减小 mid。

R - 你好世界

题目大意

进行 m 次操作，每次操作对一个区间内的所有点之间加上一条权值相同的边，求 m 次操作后点 1 与点 n 间的最短路。

R - 你好世界

题目大意

进行 m 次操作，每次操作对一个区间内的所有点之间加上一条权值相同的边，求 m 次操作后点 1 与点 n 间的最短路。

- 一血：张博轩
- 分享者：张博轩
- 知识点：最短路，建图

R - 你好世界

暴力建图建边数为 $O(mn^2)$，考虑如何优化建图。

R - 你好世界

暴力建图建边数为 $O(mn^2)$，考虑如何优化建图。

对于每一个操作新增一个点 x，区间内的点向 x 连一条值为 c 的边，再从 x 向区间内的边连一条值为 0 的边，这样建边数为 $O(mn)$。还能再优化吗？

R - 你好世界

暴力建图建边数为 $O(mn^2)$，考虑如何优化建图。

对于每一个操作新增一个点 x，区间内的点向 x 连一条值为 c 的边，再从 x 向区间内的边连一条值为 0 的边，这样建边数为 $O(mn)$。还能再优化吗？

~~考虑线段树优化建边~~，建边数为 $O(m \log n)$，卡卡常可以过。还能再优化吗？

R - 你好世界

暴力建图建边数为 $O(mn^2)$，考虑如何优化建图。

对于每一个操作新增一个点 x，区间内的点向 x 连一条值为 c 的边，再从 x 向区间内的边连一条值为 0 的边，这样建边数为 $O(mn)$。还能再优化吗？

~~考虑线段树优化建边~~，建边数为 $O(m \log n)$，卡卡常可以过。还能再优化吗？

对于每次操作建成的边，最后跑最短路时会用到的最多只有 1 条，有办法对每次操作只建 1 条边吗？

R - 你好世界

先将区间内的点连接起来，$\forall\ 1 < i \leq n$，建边 $< i, i-1 >$，长度为 0。每次操作建边 $< l, r >$，长度为 c。这样建边就能够满足题面所需，建边数为 $O(n+m)$。最后跑堆优化的 Dijkstra 即可。

Q - 田忌赛马

题目大意

构造出两个满足给定 $n \times m$ 个约束条件的数组。

Q - 田忌赛马

题目大意

构造出两个满足给定 $n \times m$ 个约束条件的数组。

- 一血: 冉泽宇
- 分享者: 冉泽宇
- 知识点: 差分约束

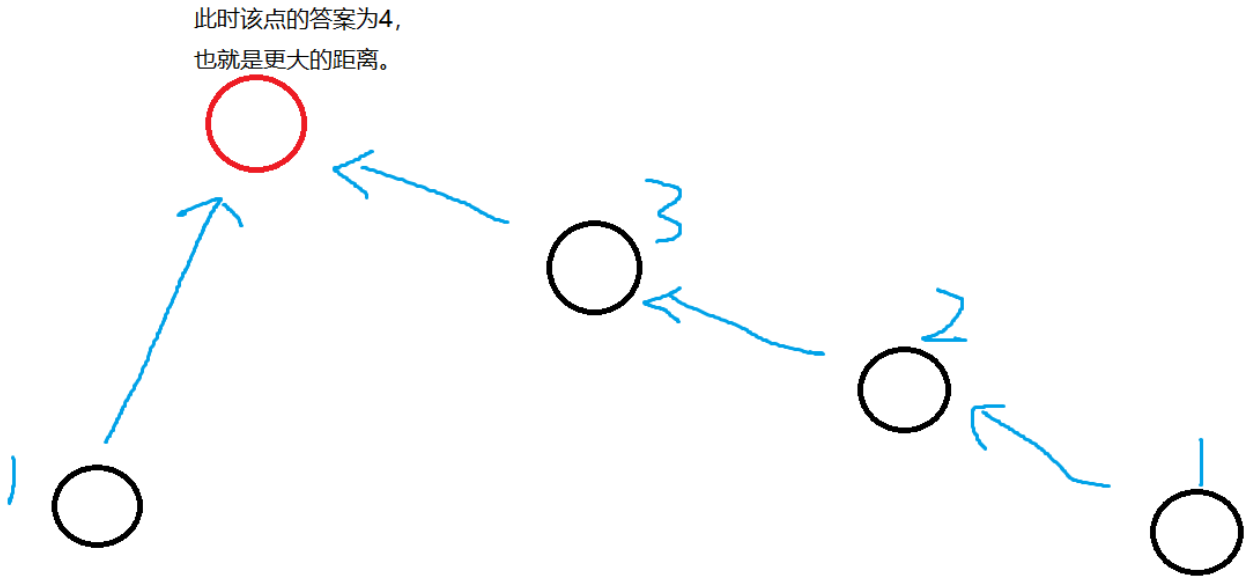
Q - 田忌赛马

对于一个约束条件 $a > b$，可以变形为 $a \geq b + 1$，有点类似于更新距离的过程。可以从 b 点向 a 点建一条有向边，再从 0 向所有点建一条有向边，从 0 开始，更新每个点的答案。

Q - 田忌赛马

对于一个约束条件 $a > b$，可以变形为 $a \geq b + 1$，有点类似于更新距离的过程。可以从 b 点向 a 点建一条有向边，再从 0 向所有点建一条有向边，从 0 开始，更新每个点的答案。

但是在这个过程中，并不是求最短距离，而是需要不断更新使得距离更大（如下图所示）。当图中出现一个环时，就认为不可能构造出一个满足约束条件的数据。



J - 惊魂夜

题目大意

给定一张无向连通图，求任意路径中割边最多的那一条有多少割边。

J - 惊魂夜

题目大意

给定一张无向连通图，求任意路径中割边最多的那一条有多少割边。

- 一血：冉泽宇
- 分享者：冉泽宇
- 知识点：树的直径，边双连通分量

J - 惊魂夜

显然割边只存在于边双连通分量之间，边双连通分量内对答案无贡献，因此我们可以将一个边双连通分量缩为一个点。

在缩点之后，我们可以发现整张图变为一颗树，而树上的所有边都是割边。因此只要找到树上最长的一条路径，即树的直径就可以了。

J - 惊魂夜

显然割边只存在于边双连通分量之间，边双连通分量内对答案无贡献，因此我们可以将一个边双连通分量缩为一个点。

在缩点之后，我们可以发现整张图变为一颗树，而树上的所有边都是割边。因此只要找到树上最长的一条路径，即树的直径就可以了。

我们从任意一点出发，进行第一次 dfs ，寻找离该节点最远的任意一个节点，它一定是某条直径的一端。然后以它为起点进行第二次 dfs ，离它最远的节点就是另一端了。它们之间的距离减一就是本题的答案。

注意本题重新建边时不要重复建边，否则会超时。

I - 鹤运物流

题目大意

给定一张无向图，每个点有一定数量的机巧鸟和一定容量的仓库，求使所有机巧鸟进入仓库的最短时间。

I - 鹤运物流

题目大意

给定一张无向图，每个点有一定数量的机巧鸟和一定容量的仓库，求使所有机巧鸟进入仓库的最短时间。

- 一血：冉泽宇
- 分享者：冉泽宇
- 知识点：Floyd，网络流，二分

I - 鹤运物流

我们可以发现即使某个点的仓库容量大于该点的机巧鸟数量，它们也不一定会选择该点的仓库，因为这样可能使其他点的鸟花费更长的时间寻找仓库。

我们发现本题的答案是让机巧鸟回到仓库的最大时间最小，这很容易让我们想到二分答案。

I - 鹤运物流

我们可以发现即使某个点的仓库容量大于该点的机巧鸟数量，它们也不一定会选择该点的仓库，因为这样可能使其他点的鸟花费更长的时间寻找仓库。

我们发现本题的答案是让机巧鸟回到仓库的最大时间最小，这很容易让我们想到二分答案。

在二分确定了最大时间之后，我们就可以知道机巧鸟的可能移动方案。我们可以用 Floyd 求出任意两点之间的最短路，只要两点之间的最短路时间小于最大时间，那么它就可能是机巧鸟的移动方案。

I - 鹤运物流

那怎么判断在此情况下能否使所有机巧鸟回到仓库呢？我们可以使用网络流。

I - 鹤运物流

那怎么判断在此情况下能否使所有机巧鸟回到仓库呢？我们可以使用网络流。

将每个点拆成机巧鸟 a 和仓库 b 两个部分，从超级源点 S 向各点 a 连接一条容量为该点机巧鸟数量的边，所有可能的移动方案之间从 a 到 b 连接容量为正无限的边，最后从各点 b 连接一条容量为仓库容量的边。

I - 鹤运物流

那怎么判断在此情况下能否使所有机巧鸟回到仓库呢？我们可以使用网络流。

将每个点拆成机巧鸟 a 和仓库 b 两个部分，从超级源点 S 向各点 a 连接一条容量为该点机巧鸟数量的边，所有可能的移动方案之间从 a 到 b 连接容量为正无限的边，最后从各点 b 连接一条容量为仓库容量的边。

如果该流量网络的最大流等于机巧鸟的数量，那么这个方案就是可行的。

D - 简单的数学题

题目大意

给定正整数 K，求 K 的倍数的最小数位和。

D - 简单的数学题

题目大意

给定正整数 K，求 K 的倍数的最小数位和。

- 一血：雷昊
- 分享者：雷昊
- 知识点：同余最短路，01 BFS

D - 简单的数学题

一个典型的错解：枚举正整数 K 的倍数，求每一个倍数的数位和。这么做不能保证在复杂度允许的范围内找到正确答案（而且这么做与图论无关）。

D - 简单的数学题

一个典型的错解：枚举正整数 K 的倍数，求每一个倍数的数位和。这么做不能保证在复杂度允许的范围内找到正确答案（而且这么做与图论无关）。

假设 K 各位数之和最小的倍数是 S，显然有 $S \equiv 0 \pmod{K}$，考虑如何得到这个正整数 S。

D - 简单的数学题

一个典型的错解：枚举正整数 K 的倍数，求每一个倍数的数位和。这么做不能保证在复杂度允许的范围内找到正确答案（而且这么做与图论无关）。

假设 K 各位数之和最小的倍数是 S，显然有 $S \equiv 0 \pmod{K}$，考虑如何得到这个正整数 S。

对于任一正整数，都可以通过数字 1 不断进行 $+1$ 和 $\times 10$ 两种操作得到。

D - 简单的数学题

一个典型的错解：枚举正整数 K 的倍数，求每一个倍数的数位和。这么做不能保证在复杂度允许的范围内找到正确答案（而且这么做与图论无关）。

假设 K 各位数之和最小的倍数是 S，显然有 $S \equiv 0 \pmod{K}$，考虑如何得到这个正整数 S。

对于任一正整数，都可以通过数字 1 不断进行 $+1$ 和 $\times 10$ 两种操作得到。

对于 $f(x)$，我们有

$$f(x+1) = f(x) + 1$$
$$f(x \times 10) = f(x)$$

D - 简单的数学题

$$f(x+1) = f(x) + 1$$
$$f(x \times 10) = f(x)$$

当然，在个位为 9 时第一个式子并不成立，但这并不会影响接下来的推导。

D - 简单的数学题

$$f(x+1) = f(x) + 1$$
$$f(x \times 10) = f(x)$$

当然，在个位为 9 时第一个式子并不成立，但这并不会影响接下来的推导。

通过上式，我们可以对一个正整数 x 连两条边：$< x, (x+1) \bmod K >$ 边权为 1 ，$< x, (10 \times x) \bmod K >$ 边权为 0，答案即为 1 到 0 的最短路。

D - 简单的数学题

K 的范围为 10^7，使用一般的最短路会超时。注意到边权只有 0 和 1，可以使用 **01bfs** 在 $O(K)$ 的复杂度实现，具体方法为，通过边权为 0 拓展的点插入队首，通过边权为 1 拓展的点插入队尾，这样便保证了队列的单调性。

T - 模拟宇宙 · 其二

题目大意

给定一张有向图，至多逆行一条边，以一号点为起始点至多可以到达多少个点。

T - 模拟宇宙 · 其二

题目大意

给定一张有向图，至多逆行一条边，以一号点为起始点至多可以到达多少个点。

- 一血：余天羽
- 分享者：余天羽
- 知识点：强连通（缩点）拓扑排序

T - 模拟宇宙 · 其二

首先可以想到如果可以到达强连通分量中的任意一个节点，那么该强连通分量中所有节点均可到达。

因此我们不难想到将一个强连通分量缩成一个点，该点对答案的贡献即为强连通分量内的节点个数。

T - 模拟宇宙 · 其二

缩完点之后，整张图变为一张 DAG ，我们可以把接下来的过程分成两种情况：

T - 模拟宇宙 · 其二

缩完点之后，整张图变为一张 DAG ，我们可以把接下来的过程分成两种情况：

- 不逆行，这样答案就是一号点所在的强连通分量内的节点个数，因为一旦我们出去就回不来了。因此该情况只出现在缩完点后只剩一个点时。

T - 模拟宇宙 · 其二

缩完点之后，整张图变为一张 DAG ，我们可以把接下来的过程分成两种情况：

- 不逆行，这样答案就是一号点所在的强连通分量内的节点个数，因为一旦我们出去就回不来了。因此该情况只出现在缩完点后只剩一个点时。
- 逆行，整个过程可以划分为三个部分，从一号点到某点 u ，然后从点 u 逆行至某点 v，最后从点 v 到一号点。由于该图为 DAG ，我们可以用拓扑排序或其他方式求出一号点到点 i 的最多可经过节点数 $\texttt{max1}_i$，以及点 i 到一号点的最长路 $\texttt{max2}_i$ 。那么答案即为 $\max_{u,v}\{\texttt{max1}_u + \texttt{max2}_v\}$ - 一号点所在强连通分量内的节点个数，其中 v 存在一条路通往 u 。注意一号点的重复部分需要减去。

T - 模拟宇宙 · 其二

缩完点之后，整张图变为一张 DAG ，我们可以把接下来的过程分成两种情况：

- 不逆行，这样答案就是一号点所在的强连通分量内的节点个数，因为一旦我们出去就回不来了。因此该情况只出现在缩完点后只剩一个点时。
- 逆行，整个过程可以划分为三个部分，从一号点到某点 u ，然后从点 u 逆行至某点 v，最后从点 v 到一号点。由于该图为 DAG ，我们可以用拓扑排序或其他方式求出一号点到点 i 的最多可经过节点数 $\texttt{max1}_i$，以及点 i 到一号点的最长路 $\texttt{max2}_i$ 。那么答案即为 $\max_{u,v}\{\texttt{max1}_u + \texttt{max2}_v\}$ - 一号点所在强连通分量内的节点个数，其中 v 存在一条路通往 u 。注意一号点的重复部分需要减去。

最后我们可以考虑一下为什么其他部分不会被重复计算：如果存在这样一个点，既可以从一号点到它，也可以从它到一号点，那么它一定在一号点所在强连通分量内。

S - 宇宙市场趋势

题目大意

给定一张图，从中选出若干个点以及连接它们的边，使边权减点权的差最大。

S - 宇宙市场趋势

题目大意

给定一张图，从中选出若干个点以及连接它们的边，使边权减点权的差最大。

- 一血：余天羽
- 分享者：余天羽
- 知识点：最大权闭合子图

S - 宇宙市场趋势

最大权闭合子图，即给定一张有向图，每个点都有一个权值，你需要选择一个权值和最大的子图，使得子图中每个点的后继都在子图中。

我们把本题抽象为成这样一个有向图模型：原图的每条边作为一个结点 u 分别向其两个端点 p, q 连有向边。u 的点权为原图中的边权；p, q 的点权为原图中点权的相反数 (表示代价)。如果选择 u 则必须选择 p, q。

这样就将本题转化为了求这个有向图的最大权闭合子图。

S - 宇宙市场趋势

求有向图的最大权闭合子图有一个通用的构造方法：

S - 宇宙市场趋势

求有向图的最大权闭合子图有一个通用的构造方法：

1. 建立超级源点 S，向所有正权点 u 连接一条容量为 u 点权的边；
2. 建立超级汇点 T，从所有负权点 p, q 分别向 T 连接一条容量为 p, q 点权绝对值的边；
3. 有向图中原有的边容量为正无限。然后求出该图的最小割，用所有正权点 u 的权值之和减去最小割，就是最大权值和。

S - 宇宙市场趋势

证明:

S - 宇宙市场趋势

证明：

最小割所去除的边一定与 S 和 T 其中一者相连。因为原有边的边权是正无限，不可能成为最小割。

S - 宇宙市场趋势

证明：

最小割所去除的边一定与 S 和 T 其中一者相连。因为原有边的边权是正无限，不可能成为最小割。

流量网络中的每一个割都对应原图中的一个闭合子图。一个割将网络分为两部分，一部分与 S 相连，另一部分与 T 相连。与 S 相连的那一部分就是该闭合子图。该闭合子图权值和 $=$ 所有正权点的权值之和 - 我们未选择的正权点的权值之和 $+$ 我们选择的负权点的权值之和。

S - 宇宙市场趋势

流量网络中的每一个割都对应原图中的一个闭合子图。一个割将网络分为两部分，一部分与 S 相连，另一部分与 T 相连。与 S 相连的那一部分就是该闭合子图。该闭合子图权值和 $=$ 所有正权点的权值之和 - 我们未选择的正权点的权值之和 $+$ 我们选择的负权点的权值之和。

S - 宇宙市场趋势

流量网络中的每一个割都对应原图中的一个闭合子图。一个割将网络分为两部分，一部分与 S 相连，另一部分与 T 相连。与 S 相连的那一部分就是该闭合子图。该闭合子图权值和 $=$ 所有正权点的权值之和 - 我们未选择的正权点的权值之和 $+$ 我们选择的负权点的权值之和。

当我们不选择一个正权点时，其与 S 的连边会被断开；当我们选择一个负权点时，其与 T 的连边会被断开。因此断开的边的边权之和即为割的容量。于是上述式子转化为：权值和 $=$ 所有正权点的权值之和 - 割的容量。

S - 宇宙市场趋势

流量网络中的每一个割都对应原图中的一个闭合子图。一个割将网络分为两部分，一部分与 S 相连，另一部分与 T 相连。与 S 相连的那一部分就是该闭合子图。该闭合子图权值和 $=$ 所有正权点的权值之和 - 我们未选择的正权点的权值之和 $+$ 我们选择的负权点的权值之和。

当我们不选择一个正权点时，其与 S 的连边会被断开；当我们选择一个负权点时，其与 T 的连边会被断开。因此断开的边的边权之和即为割的容量。于是上述式子转化为：权值和 $=$ 所有正权点的权值之和 - 割的容量。

于是得出结论，最大权值和 $=$ 所有正权点的权值之和 - 最小割。

F - 时间穿梭

题目大意

给你 n 点 m 边的无向连通图，q 次询问，每次询问给定 l, r，求使点编号在 $l \sim r$ 连通所需要用到的边的编号最大值最小是多少。

$2 \leq n \leq 10^5$

$1 \leq m, q \leq 2 \times 10^5$

F - 时间穿梭

题目大意

给你 n 点 m 边的无向连通图，q 次询问，每次询问给定 l, r，求使点编号在 $l \sim r$ 连通所需要用到的边的编号最大值最小是多少。

$2 \leq n \leq 10^5$

$1 \leq m, q \leq 2 \times 10^5$

- 一血：梁育诚
- 分享者：吕书武
- 知识点：Kruskal 重构树，LCA，ST 表

F - 时间穿梭

我们视边编号为边的权值，即是求使得一个区间的点连通所需要的边的权值最大值最小。

F - 时间穿梭

我们视边编号为边的权值，即是求使得一个区间的点连通所需要的边的权值最大值最小。

考虑到这个特征，使用 Kruskal 重构树。

F - 时间穿梭

我们视边编号为边的权值，即是求使得一个区间的点连通所需要的边的权值最大值最小。

考虑到这个特征，使用 Kruskal 重构树。

OI Wiki 描述的构建过程：

> 首先新建 n 个集合，每个集合恰有一个节点，点权为 0。
> 每一次加边会合并两个集合，我们可以新建一个点，点权为加入边的边权，同时将两个集合的根节点分别设为新建点的左儿子和右儿子。然后我们将两个集合和新建点合并成一个集合。将新建点设为根。
> 不难发现，在进行 $n-1$ 轮之后我们得到了一棵恰有 n 个叶子的二叉树，同时每个非叶子节点恰好有两个儿子。这棵树 Kruskal 重构树。

F - 时间穿梭

Kruskal 重构树的性质

在 Kruskal 重构树上，以下三个值相等：

F - 时间穿梭

Kruskal 重构树的性质

在 Kruskal 重构树上，以下三个值相等：

- 原图中两个点之间所有简单路径的最大边权的最小值
- 最小生成树上两个点之间的简单路径的最大值（瓶颈生成树）
- Kruskal 重构树上两点之间的 LCA 权值

F - 时间穿梭

现在考虑此题：

求一个区间，两两之间的路径权值最大值最小

F - 时间穿梭

现在考虑此题：

求一个区间，两两之间的路径权值最大值最小
$\Rightarrow$ 即是考虑一个区间内，所有两个点组合的 LCA

F - 时间穿梭

现在考虑此题：

求一个区间，两两之间的路径权值最大值最小
$\Rightarrow$ 即是考虑一个区间内，所有两个点组合的 LCA
$\Rightarrow$ 该区间所有点的 LCA

F - 时间穿梭

现在考虑此题：

求一个区间，两两之间的路径权值最大值最小
$\Rightarrow$ 即是考虑一个区间内，所有两个点组合的 LCA
$\Rightarrow$ 该区间所有点的 LCA
$\Rightarrow$ DFS 序最小的点和 DFS 序最大的点两点之间的 LCA。

最后一个转换是一个常见的结论。

F - 时间穿梭

现在考虑此题：

求一个区间，两两之间的路径权值最大值最小
$\Rightarrow$ 即是考虑一个区间内，所有两个点组合的 LCA
$\Rightarrow$ 该区间所有点的 LCA
$\Rightarrow$ DFS 序最小的点和 DFS 序最大的点两点之间的 LCA。

最后一个转换是一个常见的结论。

现在就维护一个区间的最大值和最小值即可转换为经典的 Kruskal 重构树问题，用 ST 表维护即可。

拓展 - 关于重构树

常见的重构树有：

1. 最小生成树
2. 虚树（保留有用点，构造新树）
3. 点分树（连接各个子树重心，子树大小和 $O(n \log n)$，可以乱搞）
4. Kruskal 重构树（利用最小生成树思想，新建点重构树）

拓展 - 关于重构树

Kruskal 重构树特点：

1. 二叉树，启发式合并
2. 每个点扩展到的连通块中的点编号是连续的，可以用数据结构维护。
3. 父亲结点权值递增，可以用倍增算法。

X - 魔法守护

题目大意

给定一张平面图，每条边有对应的权值，求在满足题设限制条件下从左上角到右下角的最小割。

X - 魔法守护

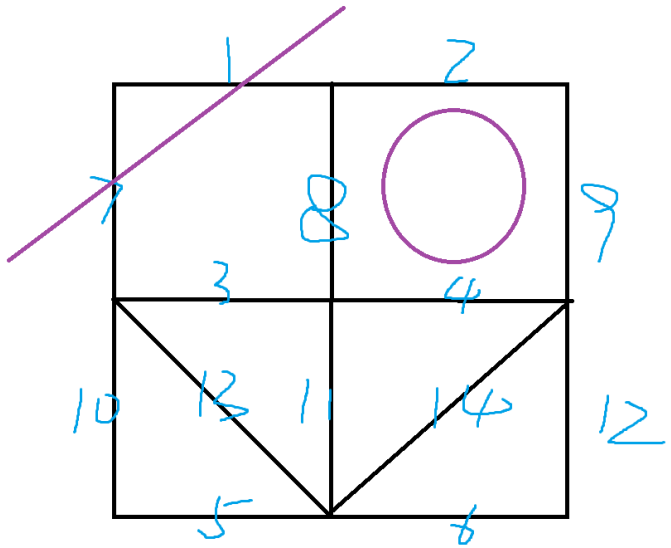
题目大意

给定一张平面图，每条边有对应的权值，求在满足题设限制条件下从左上角到右下角的最小割。

- 一血：幸旭鑫
- 分享者：幸旭鑫
- 知识点：平面图转对偶图，最短路

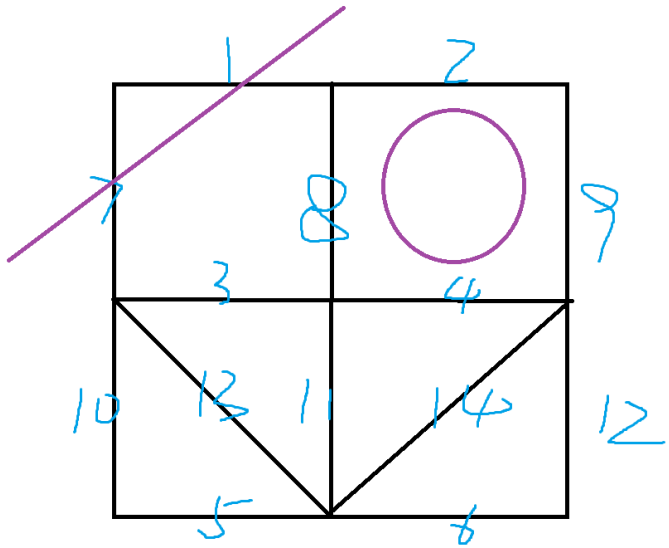
X - 魔法守护

直接求最小割显然不合理。观察样例



X - 魔法守护

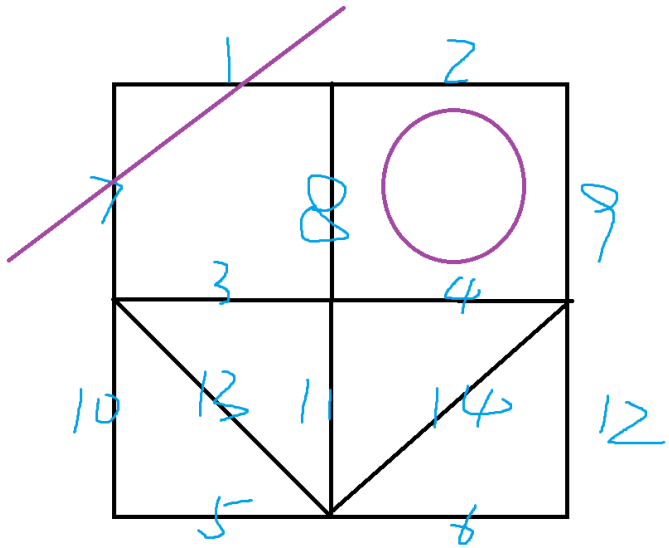
直接求最小割显然不合理。观察样例



发现这条隔断了左上角到右下角的线其实就是连接了左下角到右上角的一条线，而要求的最小割也类似于求左下角到右上角的最短距离。

X - 魔法守护

直接求最小割显然不合理。观察样例



发现这条隔断了左上角到右下角的线其实就是连接了左下角到右上角的一条线，而要求的最小割也类似于求左下角到右上角的最短距离。

从而将平面图转化为对偶图。

X - 魔法守护

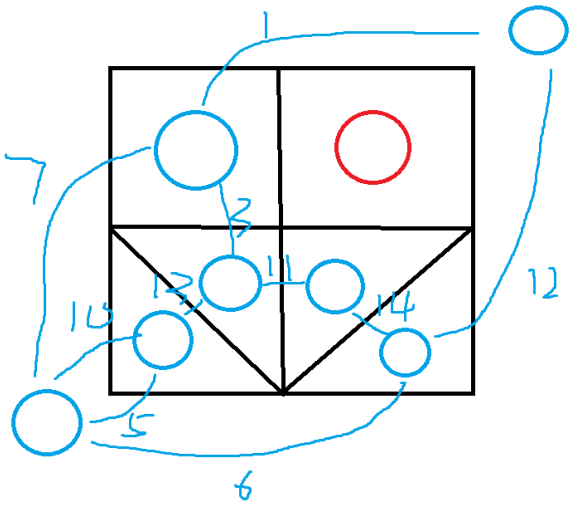
设 G 是一张平面图，构造图 G'：

1. 在 G 的每一个面 R_i 中放置一个顶点 v'_i。
2. 设 e 为 G 的一条边，若 e 在 G 的面 R_i 和 R_j 的公共边界上，做 G' 的边 e'，使得 e' 与 e 相交，且 e' 的顶点分别为 v_i' 和 v'_j、e' 的权值等于 e 的权值。

称 G' 是 G 的对偶图。

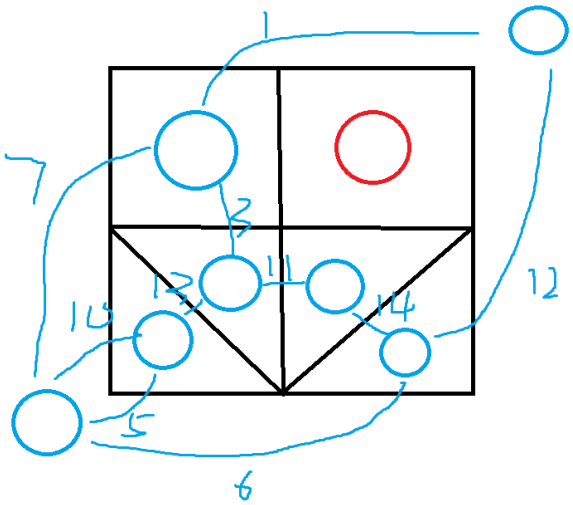
X - 魔法守护

则样例 1 变为:



X - 魔法守护

则样例 1 变为：



此时从左下角向右上角求单源最短路径即可。

Z - 认知偏差

题目大意

在一张无向连通图中，起点为 A，每次询问 B_i, C_i，判断是否有从 A 出发，经过 B_i，到达 C_i 的简单路径。

Z - 认知偏差

题目大意

在一张无向连通图中，起点为 A，每次询问 B_i, C_i，判断是否有从 A 出发，经过 B_i，到达 C_i 的简单路径。

- 一血：梁育诚
- 分享者：幸旭鑫
- 知识点：圆方树

Z - 认知偏差

回顾一下点双连通分量（下面简称点双）的定义：**任意两不同点之间都有至少两条点不重复的路径**。（一种特殊情况除外）

Z - 认知偏差

回顾一下点双连通分量（下面简称点双）的定义：**任意两不同点之间都有至少两条点不重复的路径**。（一种特殊情况除外）

将所有的点双缩点后原图便形成了一颗树，而圆方树就是一种通过点双建树的方法。

Z - 认知偏差

回顾一下点双连通分量（下面简称点双）的定义：**任意两不同点之间都有至少两条点不重复的路径**。（一种特殊情况除外）

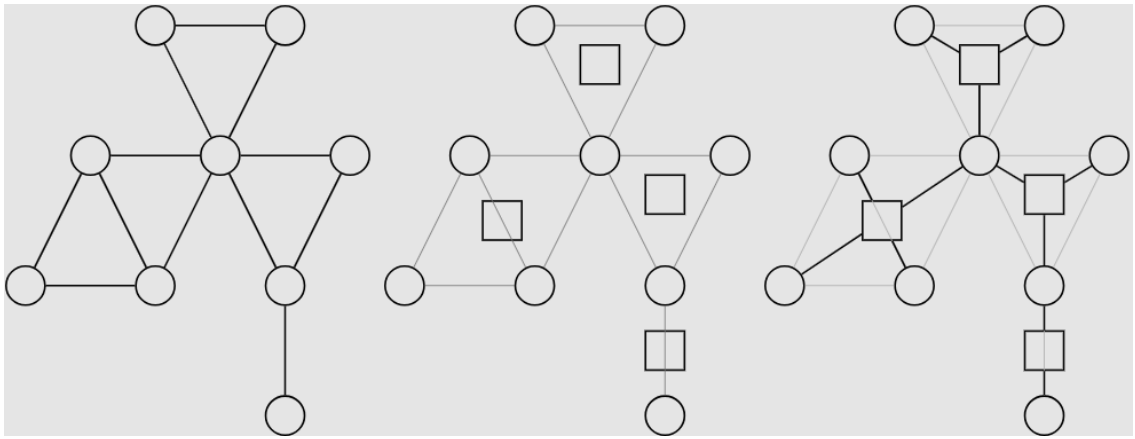
将所有的点双缩点后原图便形成了一颗树，而圆方树就是一种通过点双建树的方法。

Z - 认知偏差

圆方树的建树流程如下：通过 Tarjan 求点双（不会的快去做 M），对于每个点双新建一个**方点**，将这个方点与点双内的所有点相连。这样圆方树就建好了。~~（很简单吧）~~

Z - 认知偏差

圆方树的建树流程如下：通过 Tarjan 求点双（不会的快去做 M），对于每个点双新建一个**方点**，将这个方点与点双内的所有点相连。这样圆方树就建好了。~~（很简单吧）~~



Z - 认知偏差

点双有一个很好的性质：**点双中的两不同点 u, v 之间一定存在一条简单路径经过点双内的另一点 w。**

Z - 认知偏差

点双有一个很好的性质：**点双中的两不同点 u, v 之间一定存在一条简单路径经过点双内的另一点 w。**

将这个性质拓展到圆方树上就能得到这题的结论：存在一条从 A 出发，经过 B，到达 C 的简单路径 $\Leftrightarrow$ 圆方树上 A, C 的路径间存在一个与 B 点相连的方点。

Z - 认知偏差

点双有一个很好的性质：**点双中的两不同点 u, v 之间一定存在一条简单路径经过点双内的另一点 w。**

将这个性质拓展到圆方树上就能得到这题的结论：存在一条从 A 出发，经过 B，到达 C 的简单路径 $\Leftrightarrow$ 圆方树上 A, C 的路径间存在一个与 B 点相连的方点。

将建好的圆方树与 A 为根，如果 C 点在 $fa[B]$ 的子树内，所求的简单路径就一定存在。预处理圆方树的 DFS 序，每个询问就能 $O(1)$ 求解。

B - 阻止悲剧

题目大意

一个二维平面，有 n 个点，每个点可以用 (x_i, y_i) 表示，只允许从纵坐标大的点连向纵坐标小的点，该边的花费为两点的曼哈顿距离。

B - 阻止悲剧

题目大意

一个二维平面，有 n 个点，每个点可以用 (x_i, y_i) 表示，只允许从纵坐标大的点连向纵坐标小的点，该边的花费为两点的曼哈顿距离。

- 一血：余天羽
- 分享者：余天羽
- 知识点：费用流

B - 阻止悲剧

观察到 n 很小，考虑网络流，具体来说——费用流。

B - 阻止悲剧

观察到 n 很小，考虑网络流，具体来说——费用流。

对于原图中的每个点，拆成两个点，分别为入点和出点。新建起点和终点，起点向所有入点连一条容量为 2，费用为 0 的边，代表每个点最多两个儿子（二叉树）；所有出点向终点连一条容量为 1，费用为 0 的边，代表每个点最多有一个父亲。

B - 阻止悲剧

观察到 n 很小，考虑网络流，具体来说——费用流。

对于原图中的每个点，拆成两个点，分别为入点和出点。新建起点和终点，起点向所有入点连一条容量为 2，费用为 0 的边，代表每个点最多两个儿子（二叉树）；所有出点向终点连一条容量为 1，费用为 0 的边，代表每个点最多有一个父亲。

对于每对 i, j，$1 \leq i, j \leq n$，若满足 $y_i > y_j$，考虑连一条边，起点为 i 对应的入点，终点为 j 对应的出点，容量为 1，费用为两点的曼哈顿距离。

B - 阻止悲剧

观察到 n 很小，考虑网络流，具体来说——费用流。

对于原图中的每个点，拆成两个点，分别为入点和出点。新建起点和终点，起点向所有入点连一条容量为 2，费用为 0 的边，代表每个点最多两个儿子（二叉树）；所有出点向终点连一条容量为 1，费用为 0 的边，代表每个点最多有一个父亲。

对于每对 i, j，$1 \leq i, j \leq n$，若满足 $y_i > y_j$，考虑连一条边，起点为 i 对应的入点，终点为 j 对应的出点，容量为 1，费用为两点的曼哈顿距离。

最后跑费用流即可。

W - 意外伤害

题目大意

有一棵有 $n\,(1 \le n \le 3 \times 10^5)$ 的树，树上第 i 个点有一个权值 a_i。这些点组成了一个全集 T。你需要找到满足以下条件的点集 S 的数量：

- S 是 T 的非空子集
- 令 G 是 S 在 T 中的导出子图，则 G 形成了一棵树，且树的所有度数为 1 的节点都拥有相同的权值。

W - 意外伤害

题目大意

有一棵有 $n\,(1 \le n \le 3 \times 10^5)$ 的树，树上第 i 个点有一个权值 a_i。这些点组成了一个全集 T。你需要找到满足以下条件的点集 S 的数量：

- S 是 T 的非空子集
- 令 G 是 S 在 T 中的导出子图，则 G 形成了一棵树，且树的所有度数为 1 的节点都拥有相同的权值。

- **一血：梁育诚**
- **分享者：王振波**
- **知识点：虚树**

W - 意外伤害

我们发现对于每一种合法的方案的叶子结点，我们只关注具有相同权值的结点。这启发我们使用虚树。

虚树即是一颗只取关键结点重构形成的树，在有预处理情况下可以 $O(n)$ 构建，且结点数量也是 $O(n)$ 级别。

W - 意外伤害

我们发现对于每一种合法的方案的叶子结点，我们只关注具有相同权值的结点。这启发我们使用虚树。

虚树即是一颗只取关键结点重构形成的树，在有预处理情况下可以 $O(n)$ 构建，且结点数量也是 $O(n)$ 级别。

枚举当前讨论的权值，对具有该权值的结点构建一颗虚树：

W - 意外伤害

我们发现对于每一种合法的方案的叶子结点，我们只关注具有相同权值的结点。这启发我们使用虚树。

虚树即是一颗只取关键结点重构形成的树，在有预处理情况下可以 $O(n)$ 构建，且结点数量也是 $O(n)$ 级别。

枚举当前讨论的权值，对具有该权值的结点构建一颗虚树：

令 f_u 代表以 u 为根结点的子树在不考虑根节点（除非只有单个根节点）的情况下叶子节点都为目标权值的方案数量。

W - 意外伤害

我们发现对于每一种合法的方案的叶子结点，我们只关注具有相同权值的结点。这启发我们使用虚树。

虚树即是一颗只取关键结点重构形成的树，在有预处理情况下可以 $O(n)$ 构建，且结点数量也是 $O(n)$ 级别。

枚举当前讨论的权值，对具有该权值的结点构建一颗虚树：

令 f_u 代表以 u 为根结点的子树在不考虑根节点（除非只有单个根节点）的情况下叶子节点都为目标权值的方案数量。

令 v_i 代表 u 的第 i 个儿子，则很显然的想法是：$f_u = \prod(f_v + 1)$。

W - 意外伤害

我们发现对于每一种合法的方案的叶子结点，我们只关注具有相同权值的结点。这启发我们使用虚树。

虚树即是一颗只取关键结点重构形成的树，在有预处理情况下可以 $O(n)$ 构建，且结点数量也是 $O(n)$ 级别。

枚举当前讨论的权值，对具有该权值的结点构建一颗虚树：

令 f_u 代表以 u 为根结点的子树在不考虑根节点（除非只有单个根节点）的情况下叶子节点都为目标权值的方案数量。

令 v_i 代表 u 的第 i 个儿子，则很显然的想法是：$f_u = \prod(f_v + 1)$。

但若**当前结点不为目标权值**时，多算了自己单独点的情况，因此需要 $f_u = f_u - 1$。

W - 意外伤害

考虑在虚树上每个点统计答案：

W - 意外伤害

考虑在虚树上每个点统计答案：

1. 当前点为目前权值 $\texttt{ans} = \texttt{ans} + f_u$

W - 意外伤害

考虑在虚树上每个点统计答案：

1. 当前点为目前权值 $\texttt{ans} = \texttt{ans} + f_u$
2. 当前点不为目标权值
   此时发现若 u 度数为 1 时（即只连接了一个儿子）答案不合法，容斥一下，减去所有儿子的贡献即可。
   $\texttt{ans} = \texttt{ans} + f_u - \sum f_v$

N - 机器学习

题目大意

在一张 n 个点 m 条边的带权图中，求遍历所有的点恰好一次的最少用时，你可以花费 a_i 的时间传送至第 i 层。

N - 机器学习

题目大意

在一张 n 个点 m 条边的带权图中，求遍历所有的点恰好一次的最少用时，你可以花费 a_i 的时间传送至第 i 层。

- 一血: 梁育诚
- 分享者: 余天羽
- 知识点: 费用流 / 最小费用可行流

N - 机器学习 - 做法一

把点 i 拆成点 i_x 和点 i_y，原图的边 $< u, v >$ 转化为 $< u_x, v_y >$，这样就建成了一个二分图。

N - 机器学习 - 做法一

把点 i 拆成点 i_x 和点 i_y，原图的边 $< u, v >$ 转化为 $< u_x, v_y >$，这样就建成了一个二分图。

我们将源点 s 与所有 i_x 相连，汇点 t 与所有 i_y 相连。

N - 机器学习 - 做法一

把点 i 拆成点 i_x 和点 i_y，原图的边 $< u, v >$ 转化为 $< u_x, v_y >$，这样就建成了一个二分图。

我们将源点 s 与所有 i_x 相连，汇点 t 与所有 i_y 相连。

考虑不使用传送的情况，为了使每个点都访问次数不超过一次，将 $< s, i_x >$ 和 $< i_y, t >$ 的容量设为 1。

N - 机器学习 - 做法一

把点 i 拆成点 i_x 和点 i_y，原图的边 $< u, v >$ 转化为 $< u_x, v_y >$，这样就建成了一个二分图。

我们将源点 s 与所有 i_x 相连，汇点 t 与所有 i_y 相连。

考虑不使用传送的情况，为了使每个点都访问次数不超过一次，将 $< s, i_x >$ 和 $< i_y, t >$ 的容量设为 1。

同理，原图上每条边最多只会经过一次，将 $< u_x, v_y >$ 的容量设为 1，费用为边的权值。

N - 机器学习 - 做法一

把点 i 拆成点 i_x 和点 i_y，原图的边 $< u, v >$ 转化为 $< u_x, v_y >$，这样就建成了一个二分图。

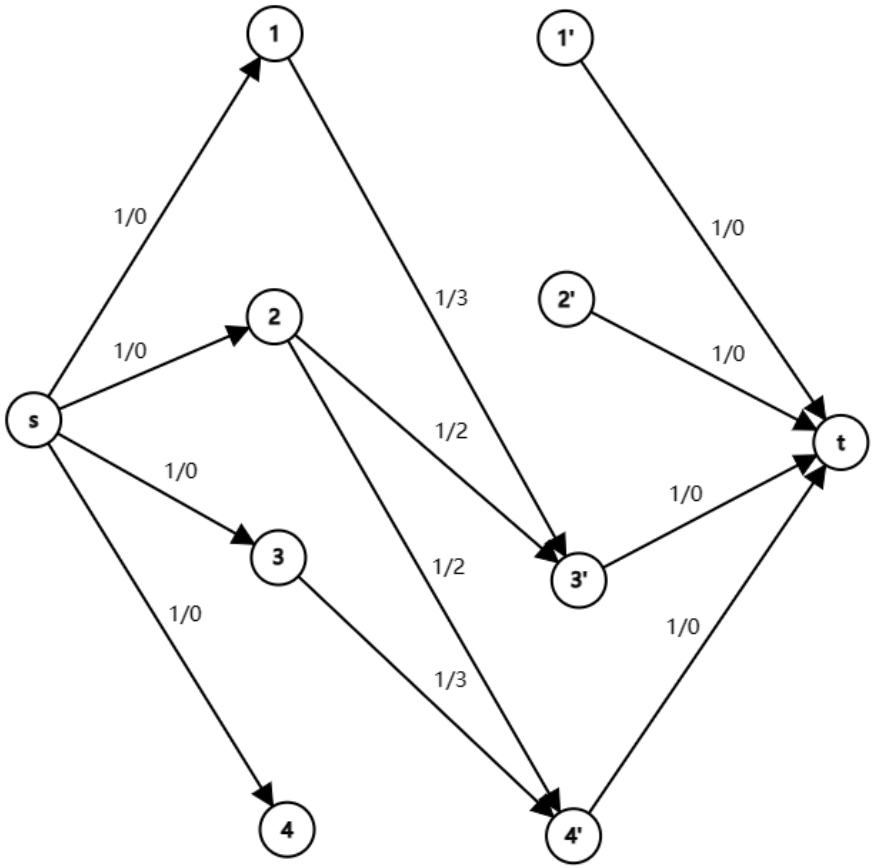
我们将源点 s 与所有 i_x 相连，汇点 t 与所有 i_y 相连。

考虑不使用传送的情况，为了使每个点都访问次数不超过一次，将 $< s, i_x >$ 和 $< i_y, t >$ 的容量设为 1。

同理，原图上每条边最多只会经过一次，将 $< u_x, v_y >$ 的容量设为 1，费用为边的权值。

从源点到汇点跑一遍最小费用最大流即可求出答案。

N - 机器学习 - 做法一



N - 机器学习 - 做法一

但由于原图是一个 DAG，在不使用传送不一定能访问所有点，接下来思考使用传送的情况。

N - 机器学习 - 做法一

但由于原图是一个 DAG，在不使用传送不一定能访问所有点，接下来思考使用传送的情况。

添加边 $< s, i_y >$ （思考为什么不是 i_x），容量为 1，费用为 a_i。

N - 机器学习 - 做法一

但由于原图是一个 DAG，在不使用传送不一定能访问所有点，接下来思考使用传送的情况。

添加边 $< s, i_y >$ （思考为什么不是 i_x），容量为 1，费用为 a_i。

最后跑一遍最小费用最大流这道题就解决啦。

N - 机器学习 - 做法二

把点 i 拆成入点 i_{in} 和出点 i_{out}，建边 $< i_{in}, i_{out} >$，为了使每个点恰好经过一次，给这条边的流量上界和流量下界都定为 1，费用为 0。

N - 机器学习 - 做法二

把点 i 拆成入点 i_{in} 和出点 i_{out}，建边 $< i_{in}, i_{out} >$，为了使每个点恰好经过一次，给这条边的流量上界和流量下界都定为 1，费用为 0。

原图的边 $< u, v >$ 转化为 $< u_{out}, u_{in} >$，容量为 inf，费用为边权。

N - 机器学习 - 做法二

把点 i 拆成入点 i_{in} 和出点 i_{out}，建边 $< i_{in}, i_{out} >$，为了使每个点恰好经过一次，给这条边的流量上界和流量下界都定为 1，费用为 0。

原图的边 $< u, v >$ 转化为 $< u_{out}, u_{in} >$，容量为 inf，费用为边权。

考虑传送，新建一个点 x，建边 $< i_{out}, x >$，容量为 inf，费用为 0。再建边 $< x, i_{in} >$，容量为 inf，费用为 a_i。

N - 机器学习 - 做法二

把点 i 拆成入点 i_{in} 和出点 i_{out}，建边 $< i_{in}, i_{out} >$，为了使每个点恰好经过一次，给这条边的流量上界和流量下界都定为 1，费用为 0。

原图的边 $< u, v >$ 转化为 $< u_{out}, u_{in} >$，容量为 inf，费用为边权。

考虑传送，新建一个点 x，建边 $< i_{out}, x >$，容量为 inf，费用为 0。再建边 $< x, i_{in} >$，容量为 inf，费用为 a_i。

到此为止，费用流模型已经建好了，接下来只需套最小费用可行流的模板即可。

K - 最小生成树？

题目大意

给定一张图，对于每条边，求出这条边最大为多少时，还能出现在图的所有最小生成树中，如果一定会出现，输出 -1。

K - 最小生成树？

题目大意

给定一张图，对于每条边，求出这条边最大为多少时，还能出现在图的所有最小生成树中，如果一定会出现，输出 -1。

- 一血：邓雄方
- 分享者：邓雄方
- 知识点：最小生成树，数据结构

K - 最小生成树？

先求出一个原图的 MST。我们把所有边分为树边和非树边。

K - 最小生成树？

先求出一个原图的 MST。我们把所有边分为树边和非树边。

对于一条非树边 (u, v)，如果将它加入 MST 中，会形成一个简单环，为了让这条边出现在 MST 中，它的边权不能是环中的最大值，故只需将其修改为 u, v 之间路径上边权最大值 -1 。

K - 最小生成树？

先求出一个原图的 MST。我们把所有边分为树边和非树边。

对于一条非树边 (u, v)，如果将它加入 MST 中，会形成一个简单环，为了让这条边出现在 MST 中，它的边权不能是环中的最大值，故只需将其修改为 u, v 之间路径上边权最大值 -1 。

对于一条树边 (u, v)，如果修改后不在 MST 中，则说明其被一条非树边代替了。

K - 最小生成树？

我们设它被非树边 (u', v') 所代替，则有 $w_{u,v}$(修改后) $\geq w_{u',v'}$ 且 (u, v) 出现在 u', v' 之间的路径上。

等价于 $\forall u', v'$ 满足边 (u, v) 在 u', v' 之间的路径上，都有 $w_{u,v}$(修改后) $\geq w_{u',v'}$。所以只需对于所有的非树边 (u', v')，用 $w_{u',v'} - 1$ 对路径上所有的树边进行覆盖取最小值。最后就能得到所有树边修改后的值。

对于非树边的求解可以用树上倍增实现，对于树边的求解可以通过多种数据结构实现，如树剖、LCT、并查集、线段树。

L - 神奇王国（hard）

题目大意

给定 n $(2 \leq n \leq 10^4)$ 件物品，每个物品有两个对应的状态值。
设 d 表示当前状态下任意两个物品所处状态值差的绝对值的最小值。
找到 d 的最大值。

L - 神奇王国（hard）

题目大意

给定 n $(2 \leq n \leq 10^4)$ 件物品，每个物品有两个对应的状态值。
设 d 表示当前状态下任意两个物品所处状态值差的绝对值的最小值。
找到 d 的最大值。

- 一血：冉泽宇
- 分享者：冉泽宇
- 知识点：2-SAT，线段树优化建图

L - 神奇王国（hard）

在 C 题解中给出建边方式的复杂度为 $O(n^2)$，不足以通过 hard version，需要考虑优化。

因为建边是对一个区间中的元素进行建边，所以考虑线段树。

L - 神奇王国（hard）

在 C 题解中给出建边方式的复杂度为 $O(n^2)$，不足以通过 hard version，需要考虑优化。

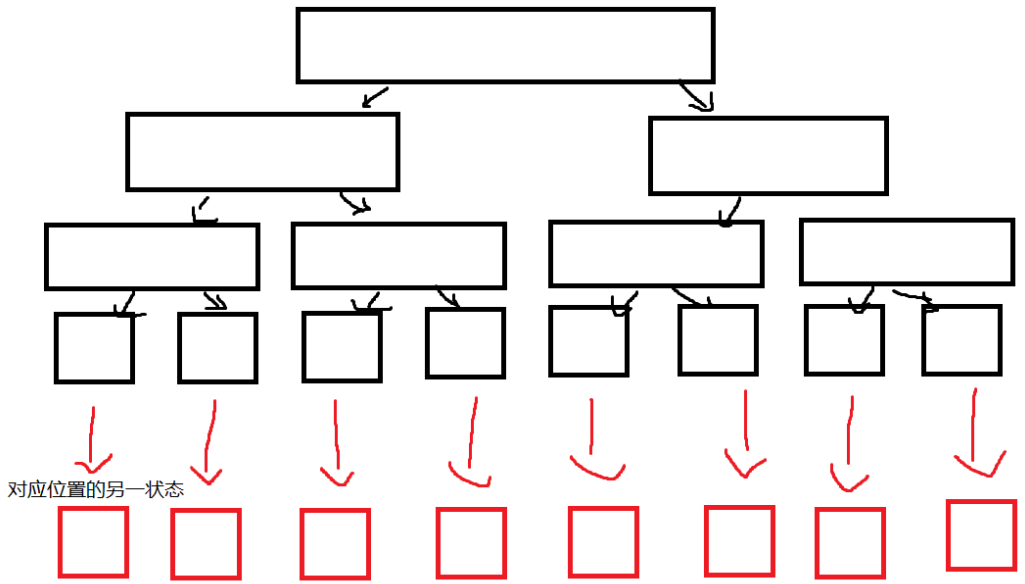
因为建边是对一个区间中的元素进行建边，所以考虑线段树。

首先对所有的 $2n$ 个值进行排序，假设当前枚举的位置为 i，二分找到需要区间的左端点和右端点 l, r，从 i 向 $(l, i-1)$ 和 $(i+1, r)$ 的另一个状态建边即可。

建一棵线段树，从每一个叶子节点建一条边指向排序后数组对应位置的元素的另一个状态。每个父亲节点建两条指向孩子节点的边。

C - 神奇王国（hard）

建一棵线段树，从每一个叶子节点建一条边指向排序后数组对应位置的元素的另一个状态。每个父亲节点建两条指向孩子节点的边。



E - 布尔公式

题目大意

给你一个完全量化的 2-CNF，即满足以下条件的布尔公式：

1. 该公式具有以下形式：$Q_1 x_1 ... Q_n x_n F(x_1, ..., x_n)$
2. 每个 Q_i 都是量词，意即 Q_i 代表 $\forall$（任意），或者 $\exists$（存在）
3. F 首先是一个合取范式，并且每个子句都是二元的，即类似于这种形式：$(s_1 \vee t_1) \wedge (s_2 \vee t_2) \ldots (s_n \vee t_n)$。$F$ 的子句有 m 个。

E - 布尔公式

题目大意

给你一个完全量化的 2-CNF，即满足以下条件的布尔公式：

1. 该公式具有以下形式：$Q_1 x_1 ... Q_n x_n F(x_1, ..., x_n)$
2. 每个 Q_i 都是量词，意即 Q_i 代表 $\forall$（任意），或者 $\exists$（存在）
3. F 首先是一个合取范式，并且每个子句都是二元的，即类似于这种形式：$(s_1 \vee t_1) \wedge (s_2 \vee t_2) \ldots (s_n \vee t_n)$。$F$ 的子句有 m 个。

- **一血**：余天羽
- **分享者**：余天羽
- **知识点**：2-SAT，思维

E - 布尔公式

由布尔公式很容易想到 2-SAT。

如何抛去量词，询问是否可满足，那就是裸的 2-SAT 模型。

E - 布尔公式

由布尔公式很容易想到 2-SAT。

如何抛去量词，询问是否可满足，那就是裸的 2-SAT 模型。

现在考虑加上量词的变化，先给出结论：

E - 布尔公式

由布尔公式很容易想到 2-SAT。

如何抛去量词，询问是否可满足，那就是裸的 2-SAT 模型。

现在考虑加上量词的变化，先给出结论：

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。

E - 布尔公式

由布尔公式很容易想到 2-SAT。

如何抛去量词，询问是否可满足，那就是裸的 2-SAT 模型。

现在考虑加上量词的变化，先给出结论：

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。

E - 布尔公式

由布尔公式很容易想到 2-SAT。

如何抛去量词，询问是否可满足，那就是裸的 2-SAT 模型。

现在考虑加上量词的变化，先给出结论：

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

条件 1 即是原始的 *2-SAT* 满足条件。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

条件 1 即是原始的 *2-SAT* 满足条件。

条件 2 确定了 $(\exists, \forall)$ 的正确性。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

条件 1 即是原始的 *2-SAT* 满足条件。
条件 2 确定了 $(\exists, \forall)$ 的正确性。
条件 3 确定了 $(\forall, \forall)$ 之间的正确性。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

条件 1 即是原始的 *2-SAT* 满足条件。
条件 2 确定了 $(\exists, \forall)$ 的正确性。
条件 3 确定了 $(\forall, \forall)$ 之间的正确性。
条件 2 和条件 3 保证了一个 $\forall$ 可以是 $\forall$ 的。

E - 布尔公式

1. 对于所有 i，x_i 和 $\overline{x_i}$ 不在同一个 SCC 中。
2. 对于所有 $i < j$ 并且 x_i 对应 $\exists$，y_i 对应 $\forall$，则 x_i 或 $\overline{x_i}$ 不会与 x_j 或 $\overline{x_j}$ 在同一个 SCC 中。
3. 对于所有 i, j，i, j 都对于 $\forall$，则不存在一条 x_i 或 $\overline{x_i}$ 到达 x_j 或 $\overline{x_j}$ 的路径。

首先 $(\exists, \exists)$ 不用考虑，$(\forall, \exists)$ 只要保证前者 $\forall$ 即可。

条件 1 即是原始的 *2-SAT* 满足条件。
条件 2 确定了 $(\exists, \forall)$ 的正确性。
条件 3 确定了 $(\forall, \forall)$ 之间的正确性。
条件 2 和条件 3 保证了一个 $\forall$ 可以是 $\forall$ 的。

理解以上可能需要对 2-SAT 的反对称性有一个清晰的理解。

Y - 精神助产术

题目大意

给定一张 n 个点 m 条边的无向连通图，判断该图所有生成树是否均同构。

Y - 精神助产术

题目大意

给定一张 n 个点 m 条边的无向连通图，判断该图所有生成树是否均同构。

- 一血：冉泽宇
- 分享者：冉泽宇
- 知识点：树哈希

Y - 精神助产术

解决本题首先需要知道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。

树哈希有很多方式，但想构造出一个不易产生冲突的哈希并不是那么简单，大家可以思考一下以下的这些哈希方法是否会产出冲突，如何构造使这些哈希产生冲突：

(令 v 是 u 的子节点，$f(u)$ 表示 u 子树的哈希值)

Y - 精神助产术

解决本题首先需要知道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。

树哈希有很多方式，但想构造出一个不易产生冲突的哈希并不是那么简单，大家可以思考一下以下的这些哈希方法是否会产出冲突，如何构造使这些哈希产生冲突：

（令 v 是 u 的子节点，$f(u)$ 表示 u 子树的哈希值）

1. $1+\Sigma f(v), \Pi f(v), 1+\oplus(f(v))$

Y - 精神助产术

解决本题首先需要知道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。

树哈希有很多方式，但想构造出一个不易产生冲突的哈希并不是那么简单，大家可以思考一下以下的这些哈希方法是否会产出冲突，如何构造使这些哈希产生冲突：

（令 v 是 u 的子节点，$f(u)$ 表示 u 子树的哈希值）

1. $1+\Sigma f(v), \Pi f(v), 1+\oplus(f(v))$
2. 对子树大小进行排序，然后计算 $\Sigma \texttt{size}_{vi} \times i$

Y - 精神助产术

解决本题首先需要知道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。

树哈希有很多方式，但想构造出一个不易产生冲突的哈希并不是那么简单，大家可以思考一下以下的这些哈希方法是否会产出冲突，如何构造使这些哈希产生冲突：

（令 v 是 u 的子节点，$f(u)$ 表示 u 子树的哈希值）

1. $1+\Sigma f(v), \Pi f(v), 1+\oplus(f(v))$
2. 对子树大小进行排序，然后计算 $\Sigma \texttt{size}_{vi} \times i$
3. 对子树哈希值进行排序，然后计算 $\texttt{size}_u + \Sigma f_{vi} \times \texttt{prime}^{i-1}$

Y - 精神助产术

我们回到这道题上。显然当 $m = n - 1$ 时生成树唯一，故同构；当 $m > n$ 时，该图一定包含至少两个简单环，我们可以先随意拆去其他环，然后根据这两个简单环是否有共同点进行讨论，具体证明需要分多种情况，此处略去。

Y - 精神助产术

我们回到这道题上。显然当 $m = n - 1$ 时生成树唯一，故同构；当 $m > n$ 时，该图一定包含至少两个简单环，我们可以先随意拆去其他环，然后根据这两个简单环是否有共同点进行讨论，具体证明需要分多种情况，此处略去。

接下来考虑 $m = n$ 的情况。该图一定只包含一个简单环，环上每个点可能还连接着一颗子树。我们求出以环上这些点为根的子树的哈希值，假设为 $a_1, a_2, a_3, ..., a_k$，它们构成一个环。现在我们只需要任意断开环中一条边，判断生成的链是否都相同即可。可以发现，只有两种情况符合要求：

Y - 精神助产术

我们回到这道题上。显然当 $m = n - 1$ 时生成树唯一，故同构；当 $m > n$ 时，该图一定包含至少两个简单环，我们可以先随意拆去其他环，然后根据这两个简单环是否有共同点进行讨论，具体证明需要分多种情况，此处略去。

接下来考虑 $m = n$ 的情况。该图一定只包含一个简单环，环上每个点可能还连接着一颗子树。我们求出以环上这些点为根的子树的哈希值，假设为 $a_1, a_2, a_3, ..., a_k$，它们构成一个环。现在我们只需要任意断开环中一条边，判断生成的链是否都相同即可。可以发现，只有两种情况符合要求：

1. $a_1 = a_2 = a_3 = ... = a_k$

Y - 精神助产术

我们回到这道题上。显然当 $m = n - 1$ 时生成树唯一，故同构；当 $m > n$ 时，该图一定包含至少两个简单环，我们可以先随意拆去其他环，然后根据这两个简单环是否有共同点进行讨论，具体证明需要分多种情况，此处略去。

接下来考虑 $m = n$ 的情况。该图一定只包含一个简单环，环上每个点可能还连接着一颗子树。我们求出以环上这些点为根的子树的哈希值，假设为 $a_1, a_2, a_3, ..., a_k$，它们构成一个环。现在我们只需要任意断开环中一条边，判断生成的链是否都相同即可。可以发现，只有两种情况符合要求：

1. $a_1 = a_2 = a_3 = ... = a_k$
2. $a_1 = a_3 = a_5 = ... = a_{k-1}$ 且 $a_2 = a_4 = a_6 = ... = a_k$

U - 翘课高手

题目大意

有 n 个人，第 i 个人上课 d_i 天，然后旷课 d_i 天，然后上课 d_i 天，以此类推。每天能选择一个来上课的人，给他加 1 分，问所有人都达到 k 分至少需要多少天。

U - 翘课高手

题目大意

有 n 个人，第 i 个人上课 d_i 天，然后旷课 d_i 天，然后上课 d_i 天，以此类推。每天能选择一个来上课的人，给他加 1 分，问所有人都达到 k 分至少需要多少天。

- 一血：余天羽
- 分享者：邓雄方
- 知识点：Hall 定理，高维前缀和

U - 翘课高手

易证明答案在区间 $[nk, 2nk]$ 中，不妨二分答案，设需要 x 天。

U - 翘课高手

易证明答案在区间 $[nk, 2nk]$ 中，不妨二分答案，设需要 x 天。

建立二分图匹配模型。左边对于每个人，拆成 k 个点，共 nk 个点；右边每天对应一个点，共 x 个点。如果第 i 个人在第 j 天去上课，将第 i 人的 k 个点均与右边的 j 点连接。求一个二分图匹配，如果最大匹配数不为 nk，说明天数不够。

U - 翘课高手

易证明答案在区间 $[nk, 2nk]$ 中，不妨二分答案，设需要 x 天。

建立二分图匹配模型。左边对于每个人，拆成 k 个点，共 nk 个点；右边每天对应一个点，共 x 个点。如果第 i 个人在第 j 天去上课，将第 i 人的 k 个点均与右边的 j 点连接。求一个二分图匹配，如果最大匹配数不为 nk，说明天数不够。

这么的建边数是 $O(nkx)$ 级别的，完全跑不动，有方法优化吗？

U - 翘课高手

回顾二分图匹配的 Hall 定理：

Hall 定理

V_1, V_2 为二分图两侧的点集，若最大匹配的数量为 $|V_1|$，当且仅当对于任意的 $S_1 \subseteq V_1$ 满足 $|S_2| \geq |S_1|$，其中 S_2 是与 S_1 中的点有边相连的点的集合。

发现 n 的数据范围并不大，可以预处理每天来上课的人的状态。然后在二分答案的 check 中，枚举 2^n 个子集，通过 Hall 定理判断是否有完美匹配。

U - 翘课高手

若当前枚举的子集为 S，只需要比较 S 集合里的人出勤日的并集大小与 $k \times |S|$ 的大小关系。

U - 翘课高手

若当前枚举的子集为 S，只需要比较 S 集合里的人出勤日的并集大小与 $k \times |S|$ 的大小关系。

记 $f[i]$ 为第 $1 \sim x$ 天中状态为 i 的天数，通过高维前缀和得到新的 $f[i]$。这样得到的 $f[i]$ 则表示某些人没来的天数。例如 $f[10110]$ 表示 $1 \sim x$ 天中第 $1, 4$ 人没来的天数。

U - 翘课高手

若当前枚举的子集为 S，只需要比较 S 集合里的人出勤日的并集大小与 $k \times |S|$ 的大小关系。

记 $f[i]$ 为第 $1 \sim x$ 天中状态为 i 的天数，通过高维前缀和得到新的 $f[i]$。这样得到的 $f[i]$ 则表示某些人没来的天数。例如 $f[10110]$ 表示 $1 \sim x$ 天中第 $1, 4$ 人没来的天数。

用总天数 x 减去 S 集合里的人没来的天数就是 S 集合里的人出勤日的并集大小。

O - 碎片整理

题目大意

在一个 $n \times m$ 的 01 矩阵中，用最少数量的 $1 \times k$ 或 $k \times 1$ 矩形覆盖所有的数字 1，且没有矩形重叠。

O - 碎片整理

题目大意

在一个 $n \times m$ 的 01 矩阵中，用最少数量的 $1 \times k$ 或 $k \times 1$ 矩形覆盖所有的数字 1，且没有矩形重叠。

- 一血: 余天羽
- 分享者: 余天羽
- 知识点: 二分图最大匹配，konig 定理

O - 碎片整理

设矩阵中数字 1 的个数为 cnt，考虑对于每个数字 1 用一个 1×1 的矩形覆盖。

O - 碎片整理

设矩阵中数字 1 的个数为 cnt，考虑对于每个数字 1 用一个 1×1 的矩形覆盖。

这些 1×1 的矩形之间存在一些公共边，我们可以选择一些公共边让相邻的两个矩形合并为一个大矩形，且每选择一条边进行合并，所需要的矩形数量就减小 1。

O - 碎片整理

设矩阵中数字 1 的个数为 cnt，考虑对于每个数字 1 用一个 1×1 的矩形覆盖。

这些 1×1 的矩形之间存在一些公共边，我们可以选择一些公共边让相邻的两个矩形合并为一个大矩形，且每选择一条边进行合并，所需要的矩形数量就减小 1。

设选择了 x 条边进行合并，所需的矩形数量为 $cnt - x$，于是我们只需要求出最多能选择多少条边进行合并。

O - 碎片整理

设矩阵中数字 1 的个数为 cnt，考虑对于每个数字 1 用一个 1×1 的矩形覆盖。

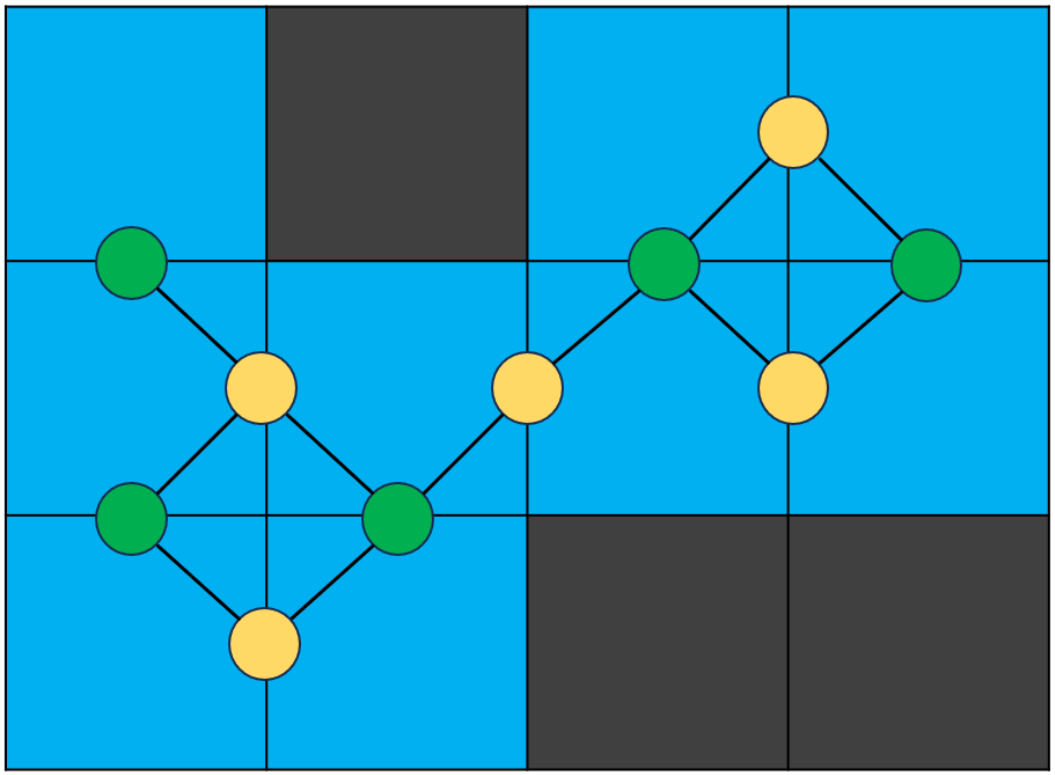
这些 1×1 的矩形之间存在一些公共边，我们可以选择一些公共边让相邻的两个矩形合并为一个大矩形，且每选择一条边进行合并，所需要的矩形数量就减小 1。

设选择了 x 条边进行合并，所需的矩形数量为 $cnt - x$，于是我们只需要求出最多能选择多少条边进行合并。

对于一个数字 1，他横方向的连边不能和纵方向的连边同时选择，否则合并后就不是满足条件的矩形了。

O - 碎片整理

考虑将公共边看作点，不能同时合并的边对应的点之间连一条边，如此一来就形成了一个二分图，接下来求**二分图最大独立集**即可。



O - 碎片整理

根据 König 定理，有：
二分图最大独立集 $= n -$ 最小点覆盖 $= n -$ 最大匹配数。

O - 碎片整理

根据 König 定理，有：
二分图最大独立集 $= n -$ 最小点覆盖 $= n -$ 最大匹配数。

剩下的就是求二分图最大匹配了，这里推荐用网络流来求。

没啦！

要好好补题哦