浅谈平面图相关算法

唐绍轩

山东省平邑第一中学

2022 年 1 月

定义 (平面图)

若图 G 能被画在平面上且不同的边仅在端点处相交，则称图 G 为平面图。画出的没有边相交的图称为 G 的平面表示或平面嵌入。

定义 (平面图)

若图 G 能被画在平面上且不同的边仅在端点处相交，则称图 G 为平面图。画出的没有边相交的图称为 G 的平面表示或平面嵌入。

定义 (面)

一个图的平面嵌入会将整个平面划分成若干个互不连通的区域，每个区域称为一个面。无界的区域称作外部面，有界的区域称作内部面。包围某个面的所有边构成了该面的边界。

定义

- 虽然外部面与内部面看起来差异很大，但它们本质上是一致的。我们可以通过球极投影实现平面和球面之间的转换，在球面上所有面的地位自然相等。

定义

- 虽然外部面与内部面看起来差异很大，但它们本质上是一致的。我们可以通过球极投影实现平面和球面之间的转换，在球面上所有面的地位自然相等。
- 这也给出了一个经典的平面图的例子：正多面体。正多面体被球极投影到平面上后，它的点和棱的投影构成了一个平面嵌入，它的面对应平面嵌入的面。

定义 (对偶图)

设 G 为平面图的一个平面嵌入，定义它的对偶图 G^* 为：

1. 在 G 中的每一个面取一个点，作为 G^* 的顶点；
2. 对于 G 中的每一条边 e，都对应一条 G^* 中的边 e^* 连接与 e 相邻的两个面中的顶点，且 e^* 在平面中穿过 e。

定义

- 对偶图给出了平面嵌入中面与面之间的关系。

定义

- 对偶图给出了平面嵌入中面与面之间的关系。
- 无论平面嵌入 G 是不是简单图，它的对偶图 G^* 都可能有自环和重边。具体来说，若 G 中有割边 e，则 G^* 中会有自环 e^*，若 G 中的某两个面的公共边界有多条边，则 G^* 中会有这两个面对应点之间的重边。

- 对偶图给出了平面嵌入中面与面之间的关系。
- 无论平面嵌入 G 是不是简单图，它的对偶图 G^* 都可能有自环和重边。具体来说，若 G 中有割边 e，则 G^* 中会有自环 e^*，若 G 中的某两个面的公共边界有多条边，则 G^* 中会有这两个面对应点之间的重边。
- 容易发现，对于一个连通的平面嵌入 G，它的对偶的对偶都是其本身，也即 $(G^*)^* = G$。

定理 (欧拉公式)

对于一个连通的平面嵌入 G，它的点数 V，边数 E，面数 F 有如下关系：

$$V - E + F = 2$$

定理 (欧拉公式)

对于一个连通的平面嵌入 G，它的点数 V，边数 E，面数 F 有如下关系：

$$V - E + F = 2$$

- 更一般地，我们有 $V - E + F = C + 1$，其中 C 为连通块数。

定理 (欧拉公式)

对于一个连通的平面嵌入 G，它的点数 V，边数 E，面数 F 有如下关系：

$$V - E + F = 2$$

- 更一般地，我们有 $V - E + F = C + 1$，其中 C 为连通块数。
- 欧拉公式给出了点数、边数和面数之间的关系，这在很多情况下都是非常有用的。

引理

对于一个简单的连通的平面嵌入 G，若 $V \geq 3$ 则有 $E \leq 3V - 6$。

引理

对于一个简单的连通的平面嵌入 G，若 $V \geq 3$ 则有 $E \leq 3V - 6$。

- 若 $E \leq 2$ 条件显然成立，否则由于图 G 为简单图，所以 G 中每个面的边界上至少有三条边，而每条边只与两个面相邻，故 $3F \leq 2E$，将欧拉公式带入即得。

定义

将无向图 $G(V,E)$ 的点集 V 划分成两个非空集合 V_1,V_2，则其中一个端点在 V_1，另一个端点在 V_2 的所有边构成了图 G 的一个割集。

定义

将无向图 $G(V,E)$ 的点集 V 划分成两个非空集合 V_1, V_2，则其中一个端点在 V_1，另一个端点在 V_2 的所有边构成了图 G 的一个割集。

引理

平面嵌入 G 的每个割集与对偶图 G^* 的每个简单环一一对应。特别地，G 的最小割对应了 G^* 中的最小环。

- 由于求某个满足性质的割集比求某个满足性质的环困难很多，我们可以用引理来简化问题。

- 由于求某个满足性质的割集比求某个满足性质的环困难很多，我们可以用引理来简化问题。
- 考虑一个面边界上的两点 s,t 之间的最小割问题。将 s,t 之间连一条边 e，把 f 分割成面 f_1 与面 f_2，再在对偶图上跑含有边 e^* 的最小环，也即去掉边 e^* 后 f_1 到 f_2 的最短路，就可以算出 s 和 t 之间的最小割。这相比使用最大流求出最小割在时间复杂度上有较大优越性。

问题 (网格图删边连通性[1])

给定一个 $n \times m$ 的网格图 G，进行 q 次操作，操作有以下两种：

1. 删除两个点 u, v 之间的边，保证在本次操作之前 u, v 之间有边。
2. 询问两个点 u, v 之间是否连通。

$1 \leq n \times m, q \leq 10^6$，操作强制在线。

[1] bzoj4423，题面有改动

- 若直接使用无向图动态连通性的算法，时间复杂度较高，无法通过此题。

- 若直接使用无向图动态连通性的算法，时间复杂度较高，无法通过此题。
- 若删边改变了连通性，那么必定有某个割集被完全删除。由引理知，此时对偶图中新形成了至少一个由删除的边构成的环。

- 若直接使用无向图动态连通性的算法，时间复杂度较高，无法通过此题。
- 若删边改变了连通性，那么必定有某个割集被完全删除。由引理知，此时对偶图中新形成了至少一个由删除的边构成的环。
- 网格图的对偶图容易求出，设对偶图中两个点之间有边 e^* 当且仅当原图中的边 e 被删除了。若对偶图中新形成了环就说明原图中删去这条边分成了两个连通块。

- 从 u, v 同时开始 BFS，每次两边分别扩展一个点，直到遍历完某个连通块为止。可以得到此处的总复杂度为 $O(nm \log nm)$。

- 从 u, v 同时开始 BFS，每次两边分别扩展一个点，直到遍历完某个连通块为止。可以得到此处的总复杂度为 $O(nm \log nm)$。
- 将遍历完的那个连通块染成一种新的颜色，那么查询时直接比较两个点的颜色是否相同即可。

- 从 u, v 同时开始 BFS，每次两边分别扩展一个点，直到遍历完某个连通块为止。可以得到此处的总复杂度为 $O(nm \log nm)$。
- 将遍历完的那个连通块染成一种新的颜色，那么查询时直接比较两个点的颜色是否相同即可。
- 时间复杂度 $O(q + nm \log nm)$。

- 从 u, v 同时开始 BFS，每次两边分别扩展一个点，直到遍历完某个连通块为止。可以得到此处的总复杂度为 $O(nm \log nm)$。
- 将遍历完的那个连通块染成一种新的颜色，那么查询时直接比较两个点的颜色是否相同即可。
- 时间复杂度 $O(q + nm \log nm)$。
- 此题存在 $O(q + nm)$ 做法，详见 [1]。

对偶图的求法

- 给定一张 n 个点的连通平面嵌入 G。即给定 G 中每个点的坐标以及若干连接两点的线段。我们需要求出它的对偶图 G^*。

- 给定一张 n 个点的连通平面嵌入 G。即给定 G 中每个点的坐标以及若干连接两点的线段。我们需要求出它的对偶图 G^*。
- 将每条无向边 (u,v) 拆成两条有向边 $u \to v$ 和 $v \to u$，并对每个点的出边极角排序。

- 给定一张 n 个点的连通平面嵌入 G。即给定 G 中每个点的坐标以及若干连接两点的线段。我们需要求出它的对偶图 G^*。
- 将每条无向边 (u,v) 拆成两条有向边 $u \to v$ 和 $v \to u$，并对每个点的出边极角排序。
- 从某条边 $e: u \to v$ 出发，每次找出以 v 为中心将边 e 顺时针旋转后碰到的第一条边 e'，再从 e' 继续往下找，直到形成了一个环。

- 可以看出，这个环就对应 G 的一个面 f，且面 f 在有向边的左侧。

对偶图的求法

- 可以看出，这个环就对应 G 的一个面 f，且面 f 在有向边的左侧。
- 找出所有面后，对于图 G 中的边 $e:(u,v)$，将 e 两侧的面在对偶图上对应的点之间连边即可。

- 可以看出，这个环就对应 G 的一个面 f，且面 f 在有向边的左侧。
- 找出所有面后，对于图 G 中的边 $e:(u,v)$，将 e 两侧的面在对偶图上对应的点之间连边即可。
- 时间复杂度 $O(n\log n)$，瓶颈在于给边极角排序。

点定位

- 给定一张 n 个点的连通平面嵌入 G。有 m 个平面上的点，需要快速求出这些点在 G 上的哪个面。

- 给定一张 n 个点的连通平面嵌入 G。有 m 个平面上的点，需要快速求出这些点在 G 上的哪个面。
- 首先跑上述找对偶图的算法。接着对横坐标扫描线，在每个时刻 t 维护 $x = t$ 这条直线穿过的边的相对位置。

点定位

- 给定一张 n 个点的连通平面嵌入 G。有 m 个平面上的点，需要快速求出这些点在 G 上的哪个面。
- 首先跑上述找对偶图的算法。接着对横坐标扫描线，在每个时刻 t 维护 $x = t$ 这条直线穿过的边的相对位置。
- 使用平衡树维护一下即可，这里的平衡树可以使用 `set` 实现。

- 给定一张 n 个点的连通平面嵌入 G。有 m 个平面上的点，需要快速求出这些点在 G 上的哪个面。
- 首先跑上述找对偶图的算法。接着对横坐标扫描线，在每个时刻 t 维护 $x = t$ 这条直线穿过的边的相对位置。
- 使用平衡树维护一下即可，这里的平衡树可以使用 set 实现。
- 时间复杂度 $O((n+m)\log n)$。

问题 (WC2013 平面图[2])

给定一张 n 个点的联通平面嵌入 G，边带权。有 q 次查询，每次给定两个平面上的点 x, y，请求出从 x 走到 y 穿过的边的边权最大值最小是多少，走的过程中不能经过外部面。
$1 \leq n, q \leq 10^5$，坐标范围 $[0, 10^7]$。

[2] https://uoj.ac/problem/57

点定位

- 首先跑点定位找出每次查询的点的位置。

点定位

- 首先跑点定位找出每次查询的点的位置。
- 只有外部面中围绕它的边为顺时针，于是使用叉积算面积的方式即可判断是否为外部面。

点定位

- 首先跑点定位找出每次查询的点的位置。
- 只有外部面中围绕它的边为顺时针，于是使用叉积算面积的方式即可判断是否为外部面。
- 剩下的事情就是一张图上查询两点间路径中最大边权的最小值，这是经典问题，不再赘述。

点定位

- 首先跑点定位找出每次查询的点的位置。
- 只有外部面中围绕它的边为顺时针，于是使用叉积算面积的方式即可判断是否为外部面。
- 剩下的事情就是一张图上查询两点间路径中最大边权的最小值，这是经典问题，不再赘述。
- 时间复杂度 $O((n+q)\log n)$。

- 考虑怎样判定一个给定的图是不是平面图。我们有以下熟知的定理。

- 考虑怎样判定一个给定的图是不是平面图。我们有以下熟知的定理。

定理 (库拉托夫斯基定理)

一个简单图是平面图当且仅当它并不包含一个是 K_5 或 $K_{3,3}$ 的分割的子图。

- 其中一个图的分割指在这个图的边上插入若干个二度顶点。

- 考虑怎样判定一个给定的图是不是平面图。我们有以下熟知的定理。

定理 (库拉托夫斯基定理)

一个简单图是平面图当且仅当它并不包含一个是 K_5 或 $K_{3,3}$ 的分割的子图。

- 其中一个图的分割指在这个图的边上插入若干个二度顶点。
- 如果直接按照该定理进行判定，所需的时间复杂度甚至是指数级的，这显然不可接受。所以我们需要更高效的算法解决这一问题。

- 设要判定的图为 $G(V,E)$。注意到自环和重边对图的可平面性没有影响，且每个联通块可以分开判定，所以不妨设 G 为简单连通图。

- 设要判定的图为 $G(V,E)$。注意到自环和重边对图的可平面性没有影响，且每个联通块可以分开判定，所以不妨设 G 为简单连通图。
- 若 G 有割点 v，则可以将 G 从割点 v 分割成若干个子图 $G_1, G_2, \cdots, G_k$，注意点 v 会出现在每个子图 G_i 中。

- 设要判定的图为 $G(V,E)$。注意到自环和重边对图的可平面性没有影响，且每个联通块可以分开判定，所以不妨设 G 为简单连通图。
- 若 G 有割点 v，则可以将 G 从割点 v 分割成若干个子图 $G_1, G_2, \cdots, G_k$，注意点 v 会出现在每个子图 G_i 中。
- 若这些图都是可平面的，则可将这些图中的点 v 都放到外部面，再把点 v 拼接起来即可得到 G 的一个平面嵌入。

- 设要判定的图为 $G(V,E)$。注意到自环和重边对图的可平面性没有影响，且每个联通块可以分开判定，所以不妨设 G 为简单连通图。
- 若 G 有割点 v，则可以将 G 从割点 v 分割成若干个子图 $G_1, G_2, \cdots, G_k$，注意点 v 会出现在每个子图 G_i 中。
- 若这些图都是可平面的，则可将这些图中的点 v 都放到外部面，再把点 v 拼接起来即可得到 G 的一个平面嵌入。
- 以下假设 G 为点双连通图。

DMP 算法

- 我们的想法是逐步确定点和边在平面上的位置。

- 我们的想法是逐步确定点和边在平面上的位置。
- 那么固定住某个子图 $H(V', E')$，G 可以分成互不干扰的若干小块，称之为段。

- 我们的想法是逐步确定点和边在平面上的位置。
- 那么固定住某个子图 $H(V', E')$，G 可以分成互不干扰的若干小块，称之为段。
- 接下来每次找出一个段，将它的某条路径嵌入至平面中，直到产生矛盾或者嵌入完成。

- 我们的想法是逐步确定点和边在平面上的位置。
- 那么固定住某个子图 $H(V',E')$，G 可以分成互不干扰的若干小块，称之为段。
- 接下来每次找出一个段，将它的某条路径嵌入至平面中，直到产生矛盾或者嵌入完成。
- 接下来，我们将更准确地叙述这一过程。

定义 (段)

定义图 G 被子图 H 划分的段为：

- 若边 $e=(u,v)$ 满足 $u\in H, v\in H, e\notin H$，则点 u,v 与边 e 构成了一个段。
- 考虑 G 删去 H 中的点后构成的连通分量 $G_1,G_2,\cdots,G_k$，对于某个 G_i，G_i 连到 H 的边及其端点与 G_i 本身构成了一个段。

定义段的附着点为同时在该段和子图 H 中的点，段的边界为附着点组成的点集，若段中某条简单路径包含的附着点恰为它的起点和终点，则称为附着路径。

- 由于每个段都是 G 的连通子图，所以每个段一定位于 H 的某个面内部，否则会导致交叉。所以，每个段的边界一定位于 H 的面上。

- 由于每个段都是 G 的连通子图，所以每个段一定位于 H 的某个面内部，否则会导致交叉。所以，每个段的边界一定位于 H 的面上。
- 记完全包含段 A 的边界的面的集合为 $F(A)$。算法如下：

- 由于每个段都是 G 的连通子图，所以每个段一定位于 H 的某个面内部，否则会导致交叉。所以，每个段的边界一定位于 H 的面上。
- 记完全包含段 A 的边界的面的集合为 $F(A)$。算法如下：

1. 开始时将 H 置为 G 中任意一个简单环，求出 G 的所有段。

- 由于每个段都是 G 的连通子图，所以每个段一定位于 H 的某个面内部，否则会导致交叉。所以，每个段的边界一定位于 H 的面上。
- 记完全包含段 A 的边界的面的集合为 $F(A)$。算法如下：

1. 开始时将 H 置为 G 中任意一个简单环，求出 G 的所有段。
2. 若存在某个段 A 满足 $|F(A)| = 0$，那么 G 不可平面，结束算法。

- 由于每个段都是 G 的连通子图，所以每个段一定位于 H 的某个面内部，否则会导致交叉。所以，每个段的边界一定位于 H 的面上。
- 记完全包含段 A 的边界的面的集合为 $F(A)$。算法如下：

1. 开始时将 H 置为 G 中任意一个简单环，求出 G 的所有段。
2. 若存在某个段 A 满足 $|F(A)|=0$，那么 G 不可平面，结束算法。
3. 若存在某个段 A 满足 $|F(A)|=1$，那么这个段所在的面 f 唯一确定，可以将 A 嵌入面 f 中。

4. 否则，任选一个段 A 以及它能嵌入的面 f。

4. 否则，任选一个段 A 以及它能嵌入的面 f。
5. 我们任意选定 A 中的一条附着路径 P 嵌入面 f，将它分割成两个面，接着更新 H 以及 G 被 H 划分出的段。

4. 否则，任选一个段 A 以及它能嵌入的面 f。
5. 我们任意选定 A 中的一条附着路径 P 嵌入面 f，将它分割成两个面，接着更新 H 以及 G 被 H 划分出的段。
6. 若 G 中没有任何段，那么 G 可平面，结束算法，否则回到 2. 继续执行。

DMP 算法

4. 否则，任选一个段 A 以及它能嵌入的面 f。
5. 我们任意选定 A 中的一条附着路径 P 嵌入面 f，将它分割成两个面，接着更新 H 以及 G 被 H 划分出的段。
6. 若 G 中没有任何段，那么 G 可平面，结束算法，否则回到 2. 继续执行。

- 直接实现这个算法，时间复杂度为 $O(n^3)$。

- 考虑优化一些实现上的细节。我们维护出每个段 A 对应的集合 $F(A)$，那么每次找出一个段 A 和面 f 的时间复杂度为 $O(n)$，接着会删除面 f，插入分割后的两个面 f_1, f_2，删除段 A，插入若干个新段。

- 考虑优化一些实现上的细节。我们维护出每个段 A 对应的集合 $F(A)$，那么每次找出一个段 A 和面 f 的时间复杂度为 $O(n)$，接着会删除面 f，插入分割后的两个面 f_1, f_2，删除段 A，插入若干个新段。
- 那么只需考虑以下两个问题：

- 考虑优化一些实现上的细节。我们维护出每个段 A 对应的集合 $F(A)$，那么每次找出一个段 A 和面 f 的时间复杂度为 $O(n)$，接着会删除面 f，插入分割后的两个面 f_1, f_2，删除段 A，插入若干个新段。
- 那么只需考虑以下两个问题：
  - 对于一个特定的面 f，对所有段 A 求出是否有 $f \in F(A)$。

- 考虑优化一些实现上的细节。我们维护出每个段 A 对应的集合 $F(A)$，那么每次找出一个段 A 和面 f 的时间复杂度为 $O(n)$，接着会删除面 f，插入分割后的两个面 f_1, f_2，删除段 A，插入若干个新段。
- 那么只需考虑以下两个问题：
  - 对于一个特定的面 f，对所有段 A 求出是否有 $f \in F(A)$。
  - 对于一个特定的段 A，对所有面 f 求出是否有 $f \in F(A)$。

- 注意到面上的总点数和段的总附着点数都为 $O(n)$，那么以上两个问题也可以在 $O(n)$ 的时间内解决。

- 注意到面上的总点数和段的总附着点数都为 $O(n)$，那么以上两个问题也可以在 $O(n)$ 的时间内解决。
- 总共循环的次数为 $O(n)$，所以总时间复杂度为 $O(n^2)$。

- 注意到面上的总点数和段的总附着点数都为 $O(n)$，那么以上两个问题也可以在 $O(n)$ 的时间内解决。
- 总共循环的次数为 $O(n)$，所以总时间复杂度为 $O(n^2)$。
- 算法的正确性证明见 [2]，此处只给出大致思路。

- 注意到面上的总点数和段的总附着点数都为 $O(n)$，那么以上两个问题也可以在 $O(n)$ 的时间内解决。
- 总共循环的次数为 $O(n)$，所以总时间复杂度为 $O(n^2)$。
- 算法的正确性证明见 [2]，此处只给出大致思路。
- 任取一个图 G 的平面嵌入作为目标。考虑可能有问题的只会是在对于所有 A，$|F(A)| \geq 2$ 时，我们任选了一个段和一个面进行嵌入。

- 注意到面上的总点数和段的总附着点数都为 $O(n)$，那么以上两个问题也可以在 $O(n)$ 的时间内解决。
- 总共循环的次数为 $O(n)$，所以总时间复杂度为 $O(n^2)$。
- 算法的正确性证明见 [2]，此处只给出大致思路。
- 任取一个图 G 的平面嵌入作为目标。考虑可能有问题的只会是在对于所有 A，$|F(A)| \geq 2$ 时，我们任选了一个段和一个面进行嵌入。
- 若一个段 A 的某种嵌入方式会影响到另一个段 B 嵌入某个面，我们称 A 和 B 为冲突段。

引理

对于冲突段 S, T，满足 $|F(S)| \geq 2, |F(T)| \geq 2$，则有 $F(S) = F(T)$ 且 $|F(S)| = 2$。

引理

对于冲突段 S,T，满足 $|F(S)| \geq 2, |F(T)| \geq 2$，则有 $F(S) = F(T)$ 且 $|F(S)| = 2$。

引理

定义图 C 的点集为所有段，边集为所有冲突段构成的无序对。当所有段 A 均满足 $|F(A)| \geq 2$ 时，图 C 是二分图。

引理

对于冲突段 S,T，满足 $|F(S)|\geq 2, |F(T)|\geq 2$，则有 $F(S)=F(T)$ 且 $|F(S)|=2$。

引理

定义图 C 的点集为所有段，边集为所有冲突段构成的无序对。当所有段 A 均满足 $|F(A)|\geq 2$ 时，图 C 是二分图。

- 若对于段 A，我们任选的面与它应该嵌入的面不同，则可以取出段 A 在 C 中的连通分量，将它们所在的面均修改至另一个面，由引理可知，此时的目标平面嵌入仍合法。

引理

对于冲突段 S, T，满足 $|F(S)| \geq 2, |F(T)| \geq 2$，则有 $F(S) = F(T)$ 且 $|F(S)| = 2$。

引理

定义图 C 的点集为所有段，边集为所有冲突段构成的无序对。当所有段 A 均满足 $|F(A)| \geq 2$ 时，图 C 是二分图。

- 若对于段 A，我们任选的面与它应该嵌入的面不同，则可以取出段 A 在 C 中的连通分量，将它们所在的面均修改至另一个面，由引理可知，此时的目标平面嵌入仍合法。
- 所以若给定的图为平面图，那么 DMP 算法一定能给出它的一个平面嵌入。

- 求一张图的完美匹配计数是很困难的，甚至对于二分图而言，这个问题都是 #P-complete 的。

- 求一张图的完美匹配计数是很困难的，甚至对于二分图而言，这个问题都是 #P-complete 的。
- 由于平面图的特殊性，我们可以使用 FKT 算法在多项式时间复杂度内算出完美匹配个数。

- 求一张图的完美匹配计数是很困难的，甚至对于二分图而言，这个问题都是 #P-complete 的。
- 由于平面图的特殊性，我们可以使用 FKT 算法在多项式时间复杂度内算出完美匹配个数。
- FKT 算法的主要思想是，将完美匹配计数转化成计算一个反对称矩阵的 pf，而后者可以转化为计算行列式，从而得到一个高效的算法。

- 求一张图的完美匹配计数是很困难的，甚至对于二分图而言，这个问题都是 #P-complete 的。
- 由于平面图的特殊性，我们可以使用 FKT 算法在多项式时间复杂度内算出完美匹配个数。
- FKT 算法的主要思想是，将完美匹配计数转化成计算一个反对称矩阵的 pf，而后者可以转化为计算行列式，从而得到一个高效的算法。
- 首先我们需要找到给定平面图的一个平面嵌入 G。实际上，DMP 算法显式地给出了一个平面嵌入的方法。

- 求一张图的完美匹配计数是很困难的，甚至对于二分图而言，这个问题都是 #P-complete 的。
- 由于平面图的特殊性，我们可以使用 FKT 算法在多项式时间复杂度内算出完美匹配个数。
- FKT 算法的主要思想是，将完美匹配计数转化成计算一个反对称矩阵的 pf，而后者可以转化为计算行列式，从而得到一个高效的算法。
- 首先我们需要找到给定平面图的一个平面嵌入 G。实际上，DMP 算法显式地给出了一个平面嵌入的方法。
- 记 G 的点数为 $2n$。若 G 不连通，则对每个连通块分别执行以下算法，下设 G 连通。

- 定义 G 的邻接矩阵 $\{G_{i,j}\}$，若 i 和 j 之间有边则 $G_{i,j}=1$，否则 $G_{i,j}=0$。那么 G 的完美匹配数 $\mathrm{PerfMatch}(G)$ 为

$$\mathrm{PerfMatch}(G)=\frac{1}{2^n n!}\sum_{p}\prod_{i=1}^{n}G_{p_{2i-1},p_{2i}}$$

- 定义 G 的邻接矩阵 $\{G_{i,j}\}$，若 i 和 j 之间有边则 $G_{i,j}=1$，否则 $G_{i,j}=0$。那么 G 的完美匹配数 $\mathrm{PerfMatch}(G)$ 为

$$\mathrm{PerfMatch}(G)=\frac{1}{2^n n!}\sum_{p}\prod_{i=1}^{n}G_{p_{2i-1},p_{2i}}$$

- 其中 p 取遍所有 $1\sim 2n$ 的排列。前面的系数是由于每一种完美匹配都恰好对应 $2^n n!$ 种不同的排列 p。

- 定义 G 的邻接矩阵 $\{G_{i,j}\}$，若 i 和 j 之间有边则 $G_{i,j}=1$，否则 $G_{i,j}=0$。那么 G 的完美匹配数 $\text{PerfMatch}(G)$ 为

$$\text{PerfMatch}(G)=\frac{1}{2^n n!}\sum_p \prod_{i=1}^{n} G_{p_{2i-1},p_{2i}}$$

- 其中 p 取遍所有 $1\sim 2n$ 的排列。前面的系数是由于每一种完美匹配都恰好对应 $2^n n!$ 种不同的排列 p。
- 对于一个矩阵 A，我们定义 $\text{pf}(A)$ 为

$$\text{pf}(A)=\frac{1}{2^n n!}\sum_p \text{sgn}(p)\prod_{i=1}^{n} A_{p_{2i-1},p_{2i}}$$

- 定义 G 的邻接矩阵 $\{G_{i,j}\}$，若 i 和 j 之间有边则 $G_{i,j}=1$，否则 $G_{i,j}=0$。那么 G 的完美匹配数 $\mathrm{PerfMatch}(G)$ 为

$$\mathrm{PerfMatch}(G)=\frac{1}{2^n n!}\sum_{p}\prod_{i=1}^{n}G_{p_{2i-1},p_{2i}}$$

- 其中 p 取遍所有 $1\sim 2n$ 的排列。前面的系数是由于每一种完美匹配都恰好对应 $2^n n!$ 种不同的排列 p。
- 对于一个矩阵 A，我们定义 $\mathrm{pf}(A)$ 为

$$\mathrm{pf}(A)=\frac{1}{2^n n!}\sum_{p}\mathrm{sgn}(p)\prod_{i=1}^{n}A_{p_{2i-1},p_{2i}}$$

- 设 $\sigma(p)$ 表示排列 p 的逆序数，那么定义 $\mathrm{sgn}(p)=(-1)^{\sigma(p)}$。

- 我们先将平面嵌入 G 的每条边定向，之后构造矩阵 A 为

$$A_{i,j} = \begin{cases} 1 & i \to j \in E' \\ -1 & j \to i \in E' \\ 0 & i \to j, j \to i \notin E' \end{cases}$$

- 我们先将平面嵌入 G 的每条边定向，之后构造矩阵 A 为

$$A_{i,j} = \begin{cases} 1 & i \to j \in E' \\ -1 & j \to i \in E' \\ 0 & i \to j, j \to i \notin E' \end{cases}$$

- 其中 E' 为图 G 的边集 E 定向以后的边集。容易发现矩阵 A 为反对称矩阵，即 $\forall i, j, A_{i,j} = -A_{j,i}$。

引理

若平面嵌入 $G(V,E)$ 的定向满足，对于任意一个内部面 f，f 的边界上都有奇数条顺时针方向的边，那么每一个完美匹配向 $\mathrm{pf}(A)$ 的贡献均相等，都为 1 或都为 -1。

引理

若平面嵌入 $G(V,E)$ 的定向满足，对于任意一个内部面 f，f 的边界上都有奇数条顺时针方向的边，那么每一个完美匹配向 $\mathrm{pf}(A)$ 的贡献均相等，都为 1 或都为 -1。

- 首先可以发现无论怎样交换匹配的顺序，排列 p 对 $\mathrm{pf}(A)$ 的贡献均不变。

引理

若平面嵌入 $G(V,E)$ 的定向满足，对于任意一个内部面 f，f 的边界上都有奇数条顺时针方向的边，那么每一个完美匹配向 $\mathrm{pf}(A)$ 的贡献均相等，都为 1 或都为 -1。

- 首先可以发现无论怎样交换匹配的顺序，排列 p 对 $\mathrm{pf}(A)$ 的贡献均不变。
- 那么对于一个完美匹配 M，它对应的所有排列的贡献均相等。

关键引理

- 两个完美匹配 M, M' 的对称差为若干个偶环的并。只需证相差一个偶环的情形，其余归纳即可。

- 两个完美匹配 M, M' 的对称差为若干个偶环的并。只需证相差一个偶环的情形，其余归纳即可。
- 记偶环为 $c : v_1 \to v_2 \to \cdots \to v_k \to v_1$，对于大小为 k 的边集 $E = (E_{i,0}, E_{i,1})$，记 $g(E) = \prod_{i=1}^{k} A_{E_{i,0},E_{i,1}}$。下证环 c 中顺时针的边数 a 必为奇数，即 $g(c) = -1$。

关键引理

- 两个完美匹配 M, M' 的对称差为若干个偶环的并。只需证相差一个偶环的情形，其余归纳即可。
- 记偶环为 $c : v_1 \to v_2 \to \cdots \to v_k \to v_1$，对于大小为 k 的边集 $E = (E_{i,0}, E_{i,1})$，记 $g(E) = \prod_{i=1}^{k} A_{E_{i,0},E_{i,1}}$。下证环 c 中顺时针的边数 a 必为奇数，即 $g(c) = -1$。
- 记严格在偶环内部的点数为 v，边数为 e，面数为 f。考虑对偶环内部（包括偶环自身）使用欧拉定理，可得 $(v+k)-(e+k)+(f+1)=2$，整理得 $v-e+f=1$。

- 两个完美匹配 M, M' 的对称差为若干个偶环的并。只需证相差一个偶环的情形，其余归纳即可。
- 记偶环为 $c: v_1 \to v_2 \to \cdots \to v_k \to v_1$，对于大小为 k 的边集 $E = (E_{i,0}, E_{i,1})$，记 $g(E) = \prod_{i=1}^{k} A_{E_{i,0}, E_{i,1}}$。下证环 c 中顺时针的边数 a 必为奇数，即 $g(c) = -1$。
- 记严格在偶环内部的点数为 v，边数为 e，面数为 f。考虑对偶环内部（包括偶环自身）使用欧拉定理，可得 $(v+k)-(e+k)+(f+1)=2$，整理得 $v-e+f=1$。
- 记 $C(u)$ 为面 u 内顺时针方向的边数，由条件知 $C(u) \equiv 1 \pmod 2$。考虑将所有偶环内部的面 u 的 $C(u)$ 求和，那么偶环中每条顺时针方向的边会有 1 的贡献，严格在偶环内部的每条边也会有 1 的贡献，所以 $\sum_u C(u) = a + e$。

关键引理

$$f \equiv \sum_u C(u) = a + e \equiv a + v + f - 1 \pmod 2$$

$$f \equiv \sum_{u} C(u) = a + e \equiv a + v + f - 1 \pmod 2$$

- 整理得

$$a \equiv 1 - v \pmod 2$$

关键引理

$$f \equiv \sum_u C(u) = a + e \equiv a + v + f - 1 \pmod 2$$

- 整理得

$$a \equiv 1 - v \pmod 2$$

- 由于完美匹配 M 交替选择了偶环中的边，而这分离了偶环的内部与外部，所以内部的点数 v 必须为偶数才能保证存在这样的完美匹配，故 a 为奇数。

关键引理

- 将 $v_1, v_2, \cdots, v_k$ 变为 $v_k, v_1, v_2, \cdots, v_{k-1}$ 会让 p 对应的符号 $\operatorname{sgn}(p)$ 乘上 -1。

- 将 $v_1, v_2, \cdots, v_k$ 变为 $v_k, v_1, v_2, \cdots, v_{k-1}$ 会让 p 对应的符号 $\mathrm{sgn}(p)$ 乘上 -1。
- 于是这两个 -1 会相互抵消，那么相差一个偶环贡献不变。

引理

给定平面嵌入 G 以及对偶图 G^*，对于任意一棵 G 的生成树 T_1，所有不在 T_1 中的边 e 的对偶边 e^* 以及 G^* 中所有点构成的子图 T_2 是 G^* 的一棵生成树。

引理

给定平面嵌入 G 以及对偶图 G^*，对于任意一棵 G 的生成树 T_1，所有不在 T_1 中的边 e 的对偶边 e^* 以及 G^* 中所有点构成的子图 T_2 是 G^* 的一棵生成树。

- 首先 T_2 无环，否则任取 T_2 的一个简单环。根据对偶图环与割对应的引理，这对应了图 G 中的一个割，与图 G 中含有的生成树 T_1 矛盾。

引理

给定平面嵌入 G 以及对偶图 G^*，对于任意一棵 G 的生成树 T_1，所有不在 T_1 中的边 e 的对偶边 e^* 以及 G^* 中所有点构成的子图 T_2 是 G^* 的一棵生成树。

- 首先 T_2 无环，否则任取 T_2 的一个简单环。根据对偶图环与割对应的引理，这对应了图 G 中的一个割，与图 G 中含有的生成树 T_1 矛盾。
- 其次 T_2 连通，否则任取两个连通块，它们之间的边均不在 T_2 中。类似地，根据引理，这对应了图 G 中的一个环，与图 G 中含有的生成树 T_1 矛盾。

引理

给定平面嵌入 G 以及对偶图 G^*，对于任意一棵 G 的生成树 T_1，所有不在 T_1 中的边 e 的对偶边 e^* 以及 G^* 中所有点构成的子图 T_2 是 G^* 的一棵生成树。

- 首先 T_2 无环，否则任取 T_2 的一个简单环。根据对偶图环与割对应的引理，这对应了图 G 中的一个割，与图 G 中含有的生成树 T_1 矛盾。
- 其次 T_2 连通，否则任取两个连通块，它们之间的边均不在 T_2 中。类似地，根据引理，这对应了图 G 中的一个环，与图 G 中含有的生成树 T_1 矛盾。
- 所以 T_2 是 G^* 的一棵生成树。

- 以下为算法流程：

具体流程

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。

具体流程

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。
2. 找出图 G 的任意一棵生成树 T_1，并对其中的边任意定向。

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。
2. 找出图 G 的任意一棵生成树 T_1，并对其中的边任意定向。
3. 找出 G 里所有不在 T_1 中的边，根据引理，它们的对偶边构成了 G^* 的生成树 T_2。

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。
2. 找出图 G 的任意一棵生成树 T_1，并对其中的边任意定向。
3. 找出 G 里所有不在 T_1 中的边，根据引理，它们的对偶边构成了 G^* 的生成树 T_2。
4. 找出一个 T_2 中的不代表外部面的叶子 v 以及它相连的边 e^*。

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。
2. 找出图 G 的任意一棵生成树 T_1，并对其中的边任意定向。
3. 找出 G 里所有不在 T_1 中的边，根据引理，它们的对偶边构成了 G^* 的生成树 T_2。
4. 找出一个 T_2 中的不代表外部面的叶子 v 以及它相连的边 e^*。
5. 注意到叶子 v 对应的面中仅含一条边 e 还未确定方向。若现在有奇数条顺时针的边，则将 e^* 定向为逆时针，否则将 e^* 定向为顺时针。

- 以下为算法流程：

1. 找出平面嵌入 G 的对偶图 G^*。
2. 找出图 G 的任意一棵生成树 T_1，并对其中的边任意定向。
3. 找出 G 里所有不在 T_1 中的边，根据引理，它们的对偶边构成了 G^* 的生成树 T_2。
4. 找出一个 T_2 中的不代表外部面的叶子 v 以及它相连的边 e^*。
5. 注意到叶子 v 对应的面中仅含一条边 e 还未确定方向。若现在有奇数条顺时针的边，则将 e^* 定向为逆时针，否则将 e^* 定向为顺时针。
6. 将 v 与 e^* 从 T_2 中删除。若 T_2 中仅含一个外部面对应的点，则结束算法，否则回到 4. 继续执行。

引理

对于一个反对称矩阵 A，有

$$\mathrm{pf}(A)^2 = \det(A)$$

引理

对于一个反对称矩阵 A，有

$$\mathrm{pf}(A)^2 = \det(A)$$

- 证明见 [3]。

引理

对于一个反对称矩阵 A，有

$$\mathrm{pf}(A)^2 = \det(A)$$

- 证明见 [3]。
- 由引理，$\mathrm{PerfMatch}(G) = |\mathrm{pf}(A)|$ 那么我们可以直接算出 A 的行列式，再开根即可。

具体流程

- 但我们通常是在 $\mathbb{F}_p$ 上做运算，而此时开根会得出两个解。我们无法区分这两个数在 $\mathbb{F}_p$ 下的正负性，这带来了一些麻烦。

- 但我们通常是在 $\mathbb{F}_p$ 上做运算，而此时开根会得出两个解。我们无法区分这两个数在 $\mathbb{F}_p$ 下的正负性，这带来了一些麻烦。
- 实际上，求解过程引入了两个正负号。

具体流程

- 但我们通常是在 $\mathbb{F}_p$ 上做运算，而此时开根会得出两个解。我们无法区分这两个数在 $\mathbb{F}_p$ 下的正负性，这带来了一些麻烦。
- 实际上，求解过程引入了两个正负号。
- 第一个在于我们不确定每个完美匹配给答案贡献的是 1 还是 -1，所以我们需要通过求解一般图完美匹配的一个解来判定贡献的正负号。

具体流程

- 但我们通常是在 $\mathbb{F}_p$ 上做运算，而此时开根会得出两个解。我们无法区分这两个数在 $\mathbb{F}_p$ 下的正负性，这带来了一些麻烦。
- 实际上，求解过程引入了两个正负号。
- 第一个在于我们不确定每个完美匹配给答案贡献的是 1 还是 -1，所以我们需要通过求解一般图完美匹配的一个解来判定贡献的正负号。
- 第二个在于算 pf 时我们通过引理得到了答案的平方，而开根时会带来正负号。实际上，我们可以通过 $\mathrm{pf}(BAB^T) = \det(B)\mathrm{pf}(A)$ 这一性质来导出一个类似消元的过程。

- 但我们通常是在 $\mathbb{F}_p$ 上做运算，而此时开根会得出两个解。我们无法区分这两个数在 $\mathbb{F}_p$ 下的正负性，这带来了一些麻烦。
- 实际上，求解过程引入了两个正负号。
- 第一个在于我们不确定每个完美匹配给答案贡献的是 1 还是 -1，所以我们需要通过求解一般图完美匹配的一个解来判定贡献的正负号。
- 第二个在于算 pf 时我们通过引理得到了答案的平方，而开根时会带来正负号。实际上，我们可以通过 $\mathrm{pf}(BAB^T) = \det(B)\mathrm{pf}(A)$ 这一性质来导出一个类似消元的过程。
- 时间复杂度 $O(n^3)$。

例题

- Alice 和 Bob 在下棋。有一个 $n \times m$ 的棋盘，棋盘的每个位置有一个权值 $a_{i,j}$，开始时棋盘中有 k 个格子上有一个棋子。Alice 和 Bob 每人轮流选择棋盘的一个没有棋子的格子，并放上棋子，Alice 先手，无棋可下的人输。Bob 选择的格子必须与 Alice 上次选择的格子有公共边。

例题

- Alice 和 Bob 在下棋。有一个 $n \times m$ 的棋盘，棋盘的每个位置有一个权值 $a_{i,j}$，开始时棋盘中有 k 个格子上有一个棋子。Alice 和 Bob 每人轮流选择棋盘的一个没有棋子的格子，并放上棋子，Alice 先手，无棋可下的人输。Bob 选择的格子必须与 Alice 上次选择的格子有公共边。
- 设一步棋的精彩度为这步棋所在位置的权值与对手上一步棋所在位置的权值之积，特别地，第一步棋的精彩度为 0。整局棋的精彩度为每步棋的精彩程度之和。

例题

- Alice 和 Bob 在下棋。有一个 $n \times m$ 的棋盘，棋盘的每个位置有一个权值 $a_{i,j}$，开始时棋盘中有 k 个格子上有一个棋子。Alice 和 Bob 每人轮流选择棋盘的一个没有棋子的格子，并放上棋子，Alice 先手，无棋可下的人输。Bob 选择的格子必须与 Alice 上次选择的格子有公共边。
- 设一步棋的精彩度为这步棋所在位置的权值与对手上一步棋所在位置的权值之积，特别地，第一步棋的精彩度为 0。整局棋的精彩度为每步棋的精彩程度之和。
- 如果两个人的策略都为随机选择一个合法位置，请计算所有可能的 Bob 胜利的局面中，整局棋精彩度的平均数。局面不同当且仅当在某一步棋走的不同。答案对 998244353 取模，保证 Bob 胜利的局面数不为 998244353 的倍数。

例题

- Alice 和 Bob 在下棋。有一个 $n \times m$ 的棋盘，棋盘的每个位置有一个权值 $a_{i,j}$，开始时棋盘中有 k 个格子上有一个棋子。Alice 和 Bob 每人轮流选择棋盘的一个没有棋子的格子，并放上棋子，Alice 先手，无棋可下的人输。Bob 选择的格子必须与 Alice 上次选择的格子有公共边。
- 设一步棋的精彩度为这步棋所在位置的权值与对手上一步棋所在位置的权值之积，特别地，第一步棋的精彩度为 0。整局棋的精彩度为每步棋的精彩程度之和。
- 如果两个人的策略都为随机选择一个合法位置，请计算所有可能的 Bob 胜利的局面中，整局棋精彩度的平均数。局面不同当且仅当在某一步棋走的不同。答案对 998244353 取模，保证 Bob 胜利的局面数不为 998244353 的倍数。
- $1 \leq n \times m \leq 400, 0 \leq k \leq nm$

例题

- 建立图 G，点由所有开始没有棋子的格子组成，若两个格子间有公共边，则将格子对应的点连边。设 G 的点数为 $2t$。

例题

- 建立图 G，点由所有开始没有棋子的格子组成，若两个格子间有公共边，则将格子对应的点连边。设 G 的点数为 $2t$。
- 考虑将 Bob 选择的格子与 Alice 上次选择的格子匹配，则 Bob 胜利时会给出图 G 的一个完美匹配。

例题

- 建立图 G，点由所有开始没有棋子的格子组成，若两个格子间有公共边，则将格子对应的点连边。设 G 的点数为 $2t$。
- 考虑将 Bob 选择的格子与 Alice 上次选择的格子匹配，则 Bob 胜利时会给出图 G 的一个完美匹配。
- 反过来，考虑一个完美匹配 $M = \{M_{i,0}, M_{i,1}\}$ 对答案产生的贡献。

例题

- 建立图 G，点由所有开始没有棋子的格子组成，若两个格子间有公共边，则将格子对应的点连边。设 G 的点数为 $2t$。
- 考虑将 Bob 选择的格子与 Alice 上次选择的格子匹配，则 Bob 胜利时会给出图 G 的一个完美匹配。
- 反过来，考虑一个完美匹配 $M = \{M_{i,0}, M_{i,1}\}$ 对答案产生的贡献。
- 首先，一个完美匹配会产生 $2^t t!$ 个不同的局面。

例题

- 建立图 G，点由所有开始没有棋子的格子组成，若两个格子间有公共边，则将格子对应的点连边。设 G 的点数为 $2t$。
- 考虑将 Bob 选择的格子与 Alice 上次选择的格子匹配，则 Bob 胜利时会给出图 G 的一个完美匹配。
- 反过来，考虑一个完美匹配 $M = \{M_{i,0}, M_{i,1}\}$ 对答案产生的贡献。
- 首先，一个完美匹配会产生 $2^t t!$ 个不同的局面。
- 接着考虑每一手棋在这些局面下对答案的贡献，我们发现只需计算以下两个式子以及完美匹配数即可。

例题

$$\sum_{i=1}^{t} a_{M_{i,0}} a_{M_{i,1}}$$

$$\sum_{i=1}^{t} \sum_{j=i+1}^{t} (a_{M_{i,0}} + a_{M_{i,1}})(a_{M_{j,0}} + a_{M_{j,1}})$$

- 设 $U_i(x) = a_{M_{i,0}} a_{M_{i,1}} x + 1, V_i(x) = (a_{M_{i,0}} + a_{M_{i,1}})x + 1$，那么只需算 $[x^1] \prod_{i=1}^{t} U_i(x)$ 与 $[x^2] \prod_{i=1}^{t} V_i(x)$。

例题

$$\sum_{i=1}^{t} a_{M_{i,0}} a_{M_{i,1}}$$

$$\sum_{i=1}^{t} \sum_{j=i+1}^{t} (a_{M_{i,0}} + a_{M_{i,1}})(a_{M_{j,0}} + a_{M_{j,1}})$$

- 设 $U_i(x) = a_{M_{i,0}} a_{M_{i,1}} x + 1, V_i(x) = (a_{M_{i,0}} + a_{M_{i,1}})x + 1$，那么只需算 $[x^1]\prod_{i=1}^{t} U_i(x)$ 与 $[x^2]\prod_{i=1}^{t} V_i(x)$。
- 如果边带权，那么 FKT 能算出所有完美匹配边权乘积的和。

例题

$$\sum_{i=1}^{t} a_{M_{i,0}} a_{M_{i,1}}$$

$$\sum_{i=1}^{t} \sum_{j=i+1}^{t} (a_{M_{i,0}} + a_{M_{i,1}})(a_{M_{j,0}} + a_{M_{j,1}})$$

- 设 $U_i(x) = a_{M_{i,0}} a_{M_{i,1}} x + 1, V_i(x) = (a_{M_{i,0}} + a_{M_{i,1}})x + 1$，那么只需算 $[x^1]\prod_{i=1}^{t} U_i(x)$ 与 $[x^2]\prod_{i=1}^{t} V_i(x)$。
- 如果边带权，那么 FKT 能算出所有完美匹配边权乘积的和。
- 那么将边 (u, v) 的边权设为 $a_u a_v x + 1$，同时在 $\bmod x^2$ 的意义下求 pf，即可算出所有完美匹配中 $[x^1]\prod_{i=1}^{t} U_i(x)$ 的和，记为 p。同理，可以通过 FKT 计算 $[x^2]\prod_{i=1}^{t} V_i(x)$ 的和，记为 q。

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。

例题

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。
- 首先通过求行列式求出 pf 的平方。

例题

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。
- 首先通过求行列式求出 pf 的平方。
- 设完美匹配数为 u，那么第一个行列式算出的答案为 $(px+u)^2 \equiv 2pux + u^2 \pmod{x^2}$，虽然我们没法得到 p 和 u 分别是什么，但我们可以得知 $\frac{p}{u} = \frac{2pu}{2 \cdot u^2}$。

例题

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。
- 首先通过求行列式求出 pf 的平方。
- 设完美匹配数为 u，那么第一个行列式算出的答案为 $(px+u)^2 \equiv 2pux + u^2 \pmod{x^2}$，虽然我们没法得到 p 和 u 分别是什么，但我们可以得知 $\frac{p}{u} = \frac{2pu}{2 \cdot u^2}$。
- 类似地，第二个行列式算出的答案为 $(qx^2 + sux + u)^2 \equiv (2qu + s^2u^2)x^2 + 2su^2x + u^2 \pmod{x^3}$，其中 s 表示 G 中所有 a_i 的和，那么 $\frac{q}{u} = \frac{(2qu+s^2u^2)-s^2u^2}{2 \cdot u^2}$。

例题

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。
- 首先通过求行列式求出 pf 的平方。
- 设完美匹配数为 u，那么第一个行列式算出的答案为 $(px+u)^2 \equiv 2pux + u^2 \pmod{x^2}$，虽然我们没法得到 p 和 u 分别是什么，但我们可以得知 $\frac{p}{u} = \frac{2pu}{2 \cdot u^2}$。
- 类似地，第二个行列式算出的答案为 $(qx^2+sux+u)^2 \equiv (2qu+s^2u^2)x^2 + 2su^2x + u^2 \pmod{x^3}$，其中 s 表示 G 中所有 a_i 的和，那么 $\frac{q}{u} = \frac{(2qu+s^2u^2)-s^2u^2}{2 \cdot u^2}$。
- 可以发现，答案可以表示为 $c_1 \cdot \frac{p}{u} + c_2 \cdot \frac{q}{u}$，其中 c_1, c_2 为容易计算的常数。

例题

- 此时，我们又碰到了符号的问题。当然我们可以通过上述符号处理得出答案，但有些繁琐。
- 首先通过求行列式求出 pf 的平方。
- 设完美匹配数为 u，那么第一个行列式算出的答案为 $(px+u)^2 \equiv 2pux + u^2 \pmod{x^2}$，虽然我们没法得到 p 和 u 分别是什么，但我们可以得知 $\frac{p}{u} = \frac{2pu}{2 \cdot u^2}$。
- 类似地，第二个行列式算出的答案为 $(qx^2+sux+u)^2 \equiv (2qu+s^2u^2)x^2 + 2su^2x + u^2 \pmod{x^3}$，其中 s 表示 G 中所有 a_i 的和，那么 $\frac{q}{u} = \frac{(2qu+s^2u^2)-s^2u^2}{2 \cdot u^2}$。
- 可以发现，答案可以表示为 $c_1 \cdot \frac{p}{u} + c_2 \cdot \frac{q}{u}$，其中 c_1, c_2 为容易计算的常数。
- 时间复杂度 $O(n^3)$。

参考文献

[1] Łącki, J., & Sankowski, P. (2017). Optimal Decremental Connectivity in Planar Graphs. Theory Comput Syst 61, 1037–1053. doi: 10.1007/s00224-016-9709-x
[2] Kohnert, A. (2004). Algorithm of Demoucron, Malgrange, Pertuiset.
[3] Haber, H. E. (2015). Notes on antisymmetric matrices and the pfaffian.

Thanks

谢谢大家。