再谈图连通性相关算法

宁波市镇海中学 虞皓翔

摘 要

本文介绍了图论中经典的连通性问题，针对 k–点/边连通和 2–点/边连通给出了对应的算法和应用，并引入了 3–点/边连通问题，并对其进行了初步介绍。

引 言

图论是组合数学中一个历史悠久的分支，以图为研究对象。图的连通性是图论中基础且重要的理论，更是图论中的奠基石。

OI 中的图论主要以数据结构为主，对连通性等的考察并不多。笔者希望通过本文，结合图论、dfs 树、网络流等多种算法，使用了传统和较现代的图论理论，来对这些基本的问题做一个介绍。

1 相关定义

1.1 图

定义 1.1.1 (图)**.** 图是一个二元组 $G = (V, E)$，其中 V, E 为集合或多重集，分别表示图的**点集**和**边集**，V 中的元素称为**顶点**，E 中的元素称为**边**。每条边是一个二元组 (u, v)，其中 $u, v \in V$。若所有边均为无序二元组，则称 G 是**无向图**；若所有边均为有序二元组，则称 G 是**有向图**。

一般地，我们用 $V(G)$ 表示 G 的点集，$E(G)$ 表示 G 的边集。

定义 1.1.2 (自环, 重边, 简单图)**.** 对无向图 $G = (V, E)$，若 $e = (u, v) \in E$ 满足 $u = v$，则称 E 是**自环**；若 E 中存在相同元素 (u, v) *(注意* (u, v) 和 (v, u) 视为相同*)*，则称它们是一组重边；若 G 中不存在自环和重边，则称 G 是简单图。

定义 1.1.3 (关联, 相邻)**.** 在无向图 G 中，若点 v 是 e 的一个端点，则称 v, e 是**关联**的，若两个顶点 u, v 是同一条边 e 的两个端点，则称 u, v 是**相邻**的。

定义 1.1.4 (邻域, 度)**.** 对于一个顶点 $v \in V$，所有与之相邻的顶点的集合 *(多重集)* 称为 v 的**邻域**，用 $N(v)$ 表示；与 v 关联的边的条数称作该顶点的**度**，用 $d(v)$ 表示。对于无向图 G，记 $\delta(G) = \min\limits_{v \in G} d(v), \Delta(G) = \max\limits_{v \in G} d(v)$ 分别表示 G 的最小度和最大度。

定义 1.1.5 (子图)**.** 对无向图 $G = (V, E)$，若存在另一张无向图 $H = (V', E')$ 满足 $V' \subseteq V$ 且 $E' \subseteq E$，则称 H 是 G 的**子图**，记作 $H \subseteq G$。

定义 1.1.6 (导出子图)**.** 若 $H = (V', E')$ 是 $G = (V, E)$ 的子图，且满足对 $\forall u, v \in V'$，只要 $(u, v) \in E$ 就有 $(u, v) \in E'$，则称 H 是 G 的**导出子图**，也称诱导子图。

容易发现，一个图的导出子图仅仅由它的点决定，因此点集为 V' 的导出子图称为 (G 中) V' 导出的子图，记作 $G[V']$。

1.2 连通性

定义 1.2.1 (途径、迹和路径)**.**

- 对于无向图 G 中一个点和边的交错序列 $w = [v_0, e_1, v_1, e_2, v_2, \cdots, e_k, v_k]$，其中首尾是点，且 $e_i = (v_{i-1}, v_i)$，则称 w 是一条**途径** *(walk)*，简记作 $w = v_0 \to v_1 \to \cdots \to v_k$ 或 $w = v_0 \rightsquigarrow v_k$。
- 对于一条途径 w，若 $e_1, e_2, \cdots, e_k$ 两两不同，则称 w 是一条**迹** *(trail)*。
- 对于一条迹 w，若 $v_i = v_j$ $(i \neq j)$ 蕴含 $\{i, j\} = \{0, k\}$，则称 w 是一条**路径** *(path)*。也就是说，除了首末端点可以相同外，其余所有的点都不能相同。

定义 1.2.2 (回路和圈)**.** 闭合[1]的迹称为**回路** *(circuit)*，闭合的路径称为**圈** *(cycle)*，也称环。

定义 1.2.3 (连通性)**.** 对于一张无向图 $G = (V, E)$ 和 $u, v \in V$，若存在途径 $u \rightsquigarrow v$，则称 u, v 是**连通的**。若 G 中任意两个顶点之间都是连通的，则称 G 是**连通图**，或 G 的这一性质称作连通性。

定理 1.2.1. 对无向图 $G = (V, E)$ “连通”是 V 上的等价关系[2]。 □

根据定理 1.2.1，我们可以根据顶点之间是否“连通”的关系将 V 划分[3]成若干个等价类 $V_1, V_2, \cdots, V_n$。其中每个等价类被称作 G 的一个**连通分量**，也称连通块。

[1] 即 $v_0 = v_k$。

[2] 即具有自反性、对称性和传递性的二元关系。

[3] “划分”即两两交集为空，所有集合并集为 V。

1.3 割集和连通度

定义 1.3.1 (点割集)**.** 对于无向图 $G=(V,E)$，其中 $u,v\in V$ 满足 $u\neq v$ 且 $(u,v)\notin E$。若存在点集 $S\subseteq V\setminus\{u,v\}$ 使得 $G[V\setminus S]$ 中 u,v 不连通，则称 S 是 u,v 的 *(局部)* **点割集**。若 S 是**某对** (u,v) 的局部点割集，则称 S 是 G 的 *(全局)* **点割集**。

定义 1.3.2 (边割集)**.** 对于无向图 $G=(V,E)$ 和 $u,v\in V$ $(u\neq v)$，如果存在边集 $F\subseteq E$ 使得在 $G\setminus F$[4] 中 u,v 不连通，则称 F 是 u,v 的 *(局部)* **边割集**。若 F 是**某对** (u,v) 的局部边割集，则称 F 是 G 的 *(全局)* **边割集**。

定义 1.3.3 (点连通度)**.** 对于无向图 G 中一对不相邻的顶点 u,v，定义 u,v 的 *(局部)* **点连通度**为 u,v 的局部点割集大小的最小值，记为 $\kappa(u,v)$。若 G 是非完全图[5]，则定义 G 的 *(全局)* **点连通度**为所有全局点割集大小的最小值，记为 $\kappa(G)$。

定义 1.3.4 (边连通度)**.** 对于无向图 $G=(V,E)$ 和 $u,v\in V$ $(u\neq v)$，定义 u,v 的 *(局部)* **边连通度**为 u,v 的局部边割集大小的最小值，记为 $\lambda(u,v)$。若 $|G|\geq 2$，则定义 G 的 *(全局)* **边连通度**为所有全局边割集大小的最小值，记为 $\lambda(G)$。若 $|G|=1$，则定义 $\lambda(G)=+\infty$。

定义 1.3.5 (k–连通性)**.** 若 $\kappa(G)\geq k$ *(*或 $\lambda(G)\geq k$*)*，则称 G 是 k–点 *(*边*)* 连通图，G 的这一性质称作 k–点 *(*边*)* 连通性。

在大多数文献中，若没有特别注明是点连通还是边连通，则一般认为其是指**点连通**。

可以发现，点连通性对图是否是简单图**不敏感**，即去掉自环，将重边缩为一条不影响两点间的点连通性。

相反，边连通性对图是否是简单图是**敏感**的，即统计边连通度时需要计算重边的重数。

2 k–连通性及其性质[6]

2.1 Menger 定理

关于点连通度和边连通度，最重要的结论或许是下述定理：

定理 2.1.1 (Menger)**.**

- 对于无向图 G 中一对不相邻的顶点 u,v，$\kappa(u,v)\geq k$ 当且仅当存在 k 条从 u 到 v 的**路径**，两两之间的公共点只有 $\{u,v\}$。

[4] 即图 $(V,E\setminus F)$ 的简记。

[5] 对于完全图 K_n，由于不存在全局点割集，因此上述定义无效，此时常见的定义由两种：$\kappa(K_n)=n-1$ 或 $\kappa(K_n)=n$，需根据上下文和相关文献合理选择。

[6] 本节内容及复杂度分析中默认点和边不带权，在边连通的分析中重边可以模拟边权的相关性质。

- 对于无向图 G 中的一对顶点 u,v，$\lambda(u,v)\geq k$ 当且仅当存在 k 条从 u 到 v 的**路径**，两两之间没有公共边。 □

事实上，Menger 定理是最大流最小割定理 [1] 的不带权版本，其中点连通和边连通分别对应于限制点容量和边容量。

2.2 连通度的界

关于，图的连通度有非常多的结论，下面给出一个最基本的结论：

定理 2.2.1. 对于无向图 G 中一对不相邻的顶点 u,v，有 $\kappa(u,v)\leq\lambda(u,v)\leq\min\{d(u),d(v)\}$。从而，对于任意非完全图的图 G，则 $\kappa(G)\leq\lambda(G)\leq\delta(G)$。

证明. 设 F 是 u,v 的局部边割集。

则对于 F 中的每一条边，至少存在一个端点不是 u 或 v。我们对所有 $|F|$ 条边都取这样一个点，得到一个点集 S，显然 $|S|\leq|F|$ 且 S 是 u,v 的局部点割集，故 $\kappa(u,v)\leq\lambda(u,v)$。不妨设 $d(u)\leq d(v)$，则与 u 关联的所有边构成 G 的一个边割集，故 $\lambda(u,v)\leq\min\{d(u),d(v)\}$。

由全局连通度和最小度的定义知 $\kappa(G)\leq\lambda(G)\leq\delta(G)$。 □

结合握手定理，有 $|V|\delta(G)\leq\sum_{v\in V}d(v)=2|E|$，得

推论 2.2.1. 对于非完全图 $G=(V,E)$，有 $\kappa(G)\leq\lambda(G)\leq\dfrac{2|E|}{|V|}$。 □

在图中添加或删除一条点和边，对图的连通度的影响如下：

定理 2.2.2.

- 点连通 *(以下默认点连通有意义，即图不是完全图).*

  设 $\kappa(G)=A$，则：

  对于任意 $v\in G$，$\kappa(G\setminus\{v\})\geq A-1$。[7]

  对于任意 $e\in G$，$\kappa(G\setminus\{e\})\leq A$。

- 边连通.

  设 $\lambda(G)=B<+\infty$，则：

  对于任意 $e\in G$，$B-1\leq\lambda(G\setminus\{e\})\leq B$。 □

这些结论可以通过定义不难得到。

[7]对于点集 $S\subseteq V(G)$，用 $G\setminus S$ 表示导出子图 $G[V(G)\setminus S]$，即删去图中若干个点。

2.3 局部 k–边连通性

2.3.1 k–边连通关系

下面我们来讨论一下局部 k–边连通性。

对于无向图 G 和正整数 k，我们在 V 上定义**二元关系** ρ_k，其中 $(a,b) \in \rho_k$ 当且仅当 $\lambda(a,b) \geq k$[8]，称该关系为图的 k–**边连通关系**。则：

定理 2.3.1. *ρ_k 是 V 上的等价关系。*

证明. 显然 ρ_k 具有自反性和对称性。只需证明 ρ_k 具有传递性。

设 $(a,b),(b,c) \in \rho_k$，即 $\lambda(a,b) \geq k, \lambda(b,c) \geq k$。

设 F 是 a,c 的局部边割集，则 $G \setminus F$ 中 a,c 不连通，于是由连通的传递性知，b 至少和其中之一不连通。不妨设 a 和 b 不连通。

于是 F 也是 a,b 的局部边割集，由定义知 $\lambda(a,c) \geq \lambda(a,b) \geq k$，即 $(a,c) \in \rho_k$。 □

2.3.2 k–边连通分量

由定理 2.3.1，ρ_k 将 V 划分为若干个等价类 $V_1, V_2, \cdots, V_n$。称其中每个等价类为 G 的一个 k–**边连通分量**。

当 $k \geq 3$ 时，一个 k–边连通分量的导出子图并不一定是连通的，如图 1 中 $\{1,2\}$ 是一个 3–边连通分量。

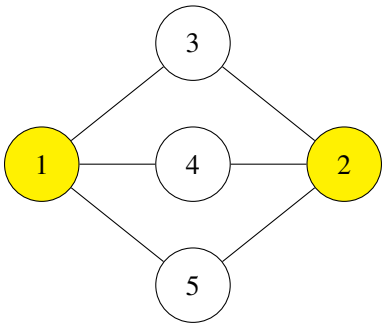

图 1

2.3.3 k–边连通判定

由定理 2.1.1，判断两个点 u,v 是否 k–边连通，只需要判断是否存在从 u 到 v 大小 $\geq k$ 的流即可。

因此，如果只需要判定一对顶点，那么使用最大流算法是一个不错的选择。

[8]特别地，当 $a = b$ 时人为规定 $\lambda(a,b) = +\infty$。

由于最大流算法具有很多不同的实现 [2]，需要合理衡量算法的时间复杂度和实现难度，因此以下用 $O(\text{Flow}(v,e))$ 表示在一张 v 个点 e 条边的 (单位容量) 图做最大流的时间复杂度。

2.3.4 k–边连通分量划分

接下来考虑如何求出 G 的 k–边连通分量划分。

一个最简单的实现是对每一个点对之间做最大流，时间复杂度 $O\left(|V|^2\,\text{Flow}(|V|,|E|)\right)$。

但实际上并不需要做那么多次最大流。

考虑取定一对顶点 u,v，我们可以在 $O(\text{Flow}(|V|,|E|))$ 的时间内判断是否有 $\lambda(u,v)\geq k$。

- 若 $\lambda(u,v)\geq k$，则 u 和 v 应在同一个 k–边连通分量中，因此以下过程中无需再考虑 v 点。
- 若 $\lambda(u,v)<k$，则说明存在一个局部边割集 F，满足 u,v 在 $G\setminus F$ 中不连通。

  于是，令 C_u 为 u 所在的连通分量，$C_v=V\setminus C_u$。此时，对于 $\forall u'\in C_u,v'\in C_v$，$F$ 也是 u' 和 v' 的局部边割集，即 $(u',v')\notin\rho_k$。

  这说明，C_u 是若干个 k–边连通分量的并，C_v 也是若干个 k–边连通分量的并。也就是说，我们将一个问题转化为了其子问题。

综上，每次操作要么“删去”某个集合中的一个顶点，要么将一个已知的集合拆分成若干个集合。当每个集合中的点都删得只剩下 1 个时，我们就得到了 G 的 k–边连通分量划分。

于是我们只需要完成 $O(|V|)$ 次最大流，时间复杂度 $O(|V|\,\text{Flow}(|V|,|E|))$。

2.4 边连通度

2.4.1 局部边连通度

判定是否为 k–边连通的一个自然推广就是计算取任意点对之间的局部边连通度。

在介绍局部边连通度的算法前，再介绍最小边割集 (最小割) 的一个性质：

定理 2.4.1. *设 $G=(V,E)$，有 $u,v\in V$。设 F 是 u,v 的最小割，设 u,v 所在的连通分量为 S,T。则对于 $\forall u'\in S,v'\in T$，均有 $\lambda(u',v')\leq\lambda(u,u')$。*

证明. *设 G 为 u,u' 的最小割，其中 u,u' 所在的连通分量为 P,Q，如图 2。*

由最小割的 Submodular 性质知可以不妨设 $T\subseteq P$ 或 $T\subseteq Q$。

- *若 $T\subseteq P$，则 G 也是 u',v' 的边割集，因此 $\lambda(u',v')\leq|G|=\lambda(u,u')$。*

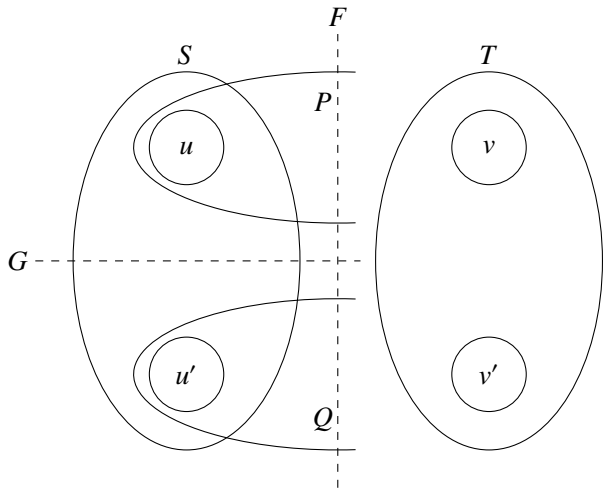

图 2

- 若 $T \subseteq Q$，则 G 也是 u, v 的边割集，因此 $\lambda(u,v) \leq |G| = \lambda(u,u')$，而 F 也是 u', v' 的边割集，因此 $\lambda(u',v') \leq |F| = \lambda(u,v)$，即 $\lambda(u',v') \leq \lambda(u,u')$。

综上，有 $\lambda(u',v') \leq \lambda(u,u')$。 □

推论 2.4.1. 条件同*定理 2.4.1*，有 $\lambda(u',v') \leq \min\{\lambda(u',u), \lambda(u,v), \lambda(v,v')\}$。 □

又由定理 2.3.1 知，$\lambda(a,c) \geq \min\{\lambda(a,b), \lambda(b,c)\}$，结合推论 2.4.1 知

定理 2.4.2. 条件同*定理 2.4.1*，有 $\lambda(u',v') = \min\{\lambda(u',u), \lambda(u,v), \lambda(v,v')\}$。 □

根据定理 2.4.2，不难得到一个计算任意点对的局部边连通度的算法：

1. 定义函数 $\mathrm{EFT}(U)$，其中 $U \subseteq V$，返回一棵带边权的树。

   $\mathrm{EFT}(U)$ 的具体过程如下：

2. 若 $|U| = 1$，**返回树** $(U, \varnothing)$。

3. 否则，任取 U 中不同两点 u, v，使用一次最大流算法得到 $\lambda(u,v)$ 以及对应的局部边割集，设对应的两个连通分量为 S^*, T^*，其中 $u \in S^*, v \in T^*$。

4. 令 $S = S^* \cap U, T = T^* \cap U$。

5. 令 $(S, E_S) = \mathrm{EFT}(S), (T, E_T) = \mathrm{EFT}(T)$。

6. 最后令边 $e = (u,v)$，边权为 $\lambda(u,v)$，函数**返回树** $(U, E_S \cup E_T \cup e)$。

要注意一点的是，运行最大流算法时一定要在原图 G 上运行，不能取导出子图 (见 2.3.2)。

上述过程中得到的树被称为**等价流树** (Equivalent-flow tree)，下面证明等价流树的主要性质：

定理 2.4.3. *对于 $\forall u, v \in V\ (u \neq v)$，$\lambda(u,v)$ 等于 $\mathrm{EFT}(T)$ 中路径 $u \rightsquigarrow v$ 中边权最小的边的权值。*

证明. *固定 G，(依照算法过程) 对 U 归纳证明。*

设 $\mathrm{EFT}(S), \mathrm{EFT}(T)$ 满足定理结论，则对于 $\mathrm{EFT}(U)$ 中的点对，只需考虑 $u' \in S, v' \in T$ 中的情形。

由 定理 2.4.2，知 $\lambda(u',v') = \min\{\lambda(u',u), \lambda(u,v), \lambda(v,v')\}$。

由归纳假设和树的性质知，$\lambda(u',v')$ 恰好等于路径 $u' \rightsquigarrow u \to v \rightsquigarrow v'$ 中边权最小的边的权值，命题成立。 □

不难证明这个算法的时间复杂度也是 $O(|V|\,\mathrm{Flow}(|V|,|E|))$，且预处理完毕后，我们将一次局部边连通度的询问转化为树上权值 min，以达到 $O(\log|V|)$ 甚至 $O(1)$ 询问。

2.4.2 Gusfield 算法

Gusfield 算法是上述算法的一种实现，由于它不需要递归，因此显得较有优势。

该算法的流程如下：

1. 任取 $r \in V$，令 $R = V \setminus \{r\}, E_T = \varnothing, B_r = R, B_v = \varnothing\ (v \neq r)$。
2. 若 $R = \varnothing$，则**返回树** (V, E_T)。
3. 取 $v \in R$，令 $R = R \setminus \{v\}$。
4. 设 u 满足 $v \in B_u$，使用最大流算法得到 $\lambda(u,v)$ 以及对应的局部边割集，设对应的两个连通分量为 S^*, T^*，其中 $u \in S^*, v \in T^*$。
5. 令 $B_v = B_u \cap T^*, B_u = B_u \cap S^*$。
6. 最后令边 $e = (u,v)$，边权为 $\lambda(u,v)$，令 $E_T = E_T \cup \{e\}$，回到步骤 2。

读者不难证明该算法和之前给出的算法是等价的。Gusfield 算法的时间复杂度仍为 $O(|V|\,\mathrm{Flow}(|V|,|E|))$。

2.4.3 全局边连通度

对于一张图，它的全局边连通度等于所有局部边连通度的最小值，因此它可以在 $O(|V|\,\mathrm{Flow}(|V|,|E|))$ 的时间内得到。

然而，如果我们只需要知道它的全局边连通度的话，有专门的算法来解决此类问题，其中比较著名的有确定性的 Stoer-Wagner 算法 [4] (时间复杂度 $O\left(|V|^2\log|V|+|V||E|\right)$) 和基于随机化的 Karger 算法 [3] (时间复杂度 $O\left(|V|^2\log^3|V|\right)$)。限于篇幅，这里不对这两种算法进行展开，感兴趣的读者可以见相关文献。

2.5 点连通度

与边连通度相对应的一个问题就是点连通度。

由图 3 可知，k–点连通性没有传递性。($\kappa(1,2)=\kappa(2,3)=2,\kappa(1,3)=1$)。

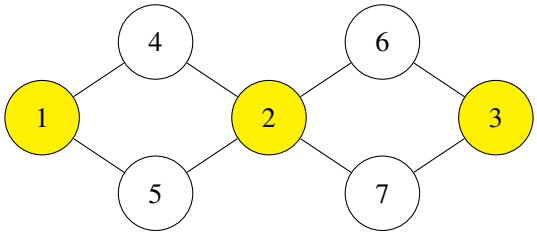

图 3

从而处理点连通度会比处理边连通度略显麻烦。

2.5.1 局部点连通度

计算两个点的点连通是容易的，由定理 2.1.1 知我们只需计算在限制点容量为 1 的条件下从 u 到 v 的最大流的大小。

对于限制了点容量，我们可以用拆点的思想，将一个点 v 拆成两个点 v,v'，中间连一条 $v\to v'$ 的容量为 1 的边。对于原来的无向边 (u,v)，转化成有向边 $u'\to v$ 和 $v'\to u$，最后使用一般的最大流算法即可。

如，图 3 拆点后就变成了图 4。

于是 $\kappa(u,v)$ 就等于从 u' 到 v 的最大流 (也等于从 v' 到 u 的最大流)。

该算法的时间复杂度为 $O(\mathrm{Flow}(|V|,|E|))$。

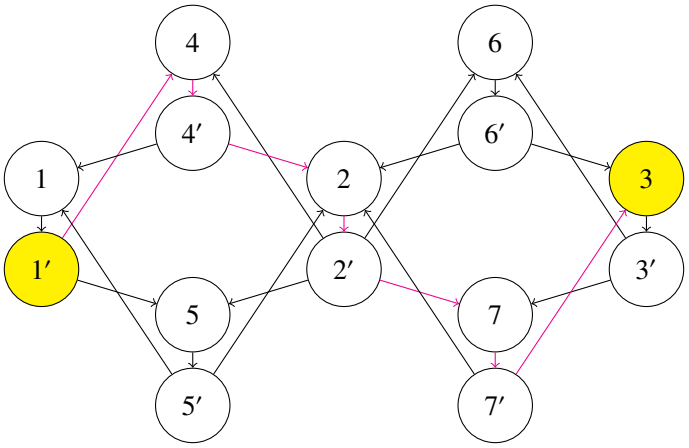

图 4

2.5.2 全局点连通度

由定义，

$$\kappa(G) = \min_{(u,v)\notin E} \kappa(u,v)$$

因此按照该定义直接计算，时间复杂度 $O\left(|V|^2 \operatorname{Flow}(|V|,|E|)\right)$。

和边连通度类似，我们并不需要运行 $O\left(|V|^2\right)$ 次最大流，因为有很多操作都是冗余的。

设 $G=(V,E)$ 是非完全图的连通无向简单图，$\kappa(G)=c$，则 $1 \le c \le |V|-2$。

于是，G 存在大小为 c 的全局点割集，设为 S。

设 $V=\{1,2,\cdots,|V|\}$。由于 $|S|=c$，因此必定存在元素 $v \in \{1,2,\cdots,c+1\}$ 满足 $v \notin S$。

又图 $G \setminus S$ 中有至少两个连通分量，从而存在 $u \in V \setminus S$ 且 u,v 在两个不同连通分量。显然 $(u,v) \notin E$。

于是 S 是 u,v 的一个局部点割集，从而 $\kappa(u,v)=c$。这说明：

定理 2.5.1. *若非完全图的连通无向简单图 $G=(V,E)$ 的点连通度为 c (即 $\kappa(G)=c$)，则对于 V 的任意一个 $c+1$ 元子集 X，一定存在 $v \in X, u \in V, (u,v) \notin E$ 满足 $\kappa(u,v)=c$。* □

于是我们得到了一个时间复杂度为 $O(\kappa(G)|V|\operatorname{Flow}(|V|,|E|))$ 的算法：

1. 不妨设 $V=\{1,2,\cdots,|V|\}$。
2. 令 $u=1, c=+\infty$。
3. 若 $u>c$，则**返回** $\kappa(G)=c$，程序结束。
4. 否则，令 $U=\{v \mid u<v \wedge (u,v) \notin E\}$，使用网络流计算

$$c_u = \min_{v\in U} \kappa(u,v)$$

5. 令 $c = \min\{c, c_u\}, u = u + 1$，回到步骤 2。

由推论 2.2.1，它的时间复杂度也等于 $O(|E|\,\mathrm{Flow}(|V|, |E|))$。

2.5.3 k–点连通分量

鉴于我们未对满足 $(u, v) \in E$ 的 u, v 定义 $\kappa(u, v)$。因此在本小节中，我们**额外给出**它的定义：

定义 2.5.1. *对于**无向简单图** $G = (V, E)$ 和 $(u, v) \in E$，定义 $\kappa(u, v)$ 为从 u 到 v 的路径条数的最大值 (含路径 $u \to v$)，满足两两之间的公共点只有 $\{u, v\}$ (也就是说，通过 Menger 定理来定义)。*

为了区分起见，我们将这个函数记为 κ^*，即

$$\kappa^*(u, v) = \begin{cases} \kappa(u, v) & (u, v) \notin E \\ u \text{ 到 } v \text{ 的路径条数的最大值} & (u, v) \in E \end{cases}$$

下面我们给出 k–点连通分量的定义。

定义 2.5.2 (k–点连通分量)**.** *对于无向简单图 $G = (V, E)$，若集合 $S \subseteq G$ 满足 $\forall u, v \in S\ (u \neq v)$，均有 $\kappa^*(u, v) \geq k$，且不存在 S 的一个真超集[9]满足上述性质，则称 S 是 G 的一个 **k–点连通分量**。*

我们考虑建立一张辅助图 $H = (V, E_H)$，其中 $E_H = \{(u, v) \mid u \neq v, \kappa^*(u, v) \geq k\}$。

当我们对边连通建图时，由于 k–边连通是等价关系，因此所得到的图为若干个不相交完全图的并，从而 G 中的一个边连通分量对应辅助图中的一个连通块。

而我们对点连通建图时，由于 k–点连通没有传递性，因此不同的点连通分量之间可能有交集，同样，当 $k \geq 3$ 时，一个 k–点连通分量并不一定能导出一个连通子图。

由定义知，G 的一个 k–点连通分量对应辅助图 H 中的一个**极大团**。因此 G 的 k–点连通分量的结构可以由 H 的极大团的结构来得到。

定理 2.5.2. *n 阶无向图至多有 n 个 k–点连通分量。* □

对于建立辅助图的过程，除了暴力对每一对点对使用局部点连通度的算法外，笔者尚未获得低于 $O\left(|V|^2\,\mathrm{Flow}(|V|, |E|)\right)$ 的算法，如果有优于此复杂度的做法，欢迎与笔者交流。

考虑 H 中两个极大团 A, B，它们的交集大小是有限制的。事实上，我们有如下结论：

定理 2.5.3. *对于 H 中任意两个不同的极大团为 A, B (对应 G 中两个不同 k–点连通分量)，则 $|A \cap B| \leq k - 1$。*

[9] 若 A 是 B 的真子集，则称 B 是 A 的真超集。

证明. 若 $|A\cap B|\geq k$，则由于 A,B 的极大性知 $A\not\subseteq B, B\not\subseteq A$，从而存在 $u\in A\setminus B, v\in B\setminus A$，满足 $(u,v)\notin H$。

由定义知 $\kappa^*(u,v)\leq k-1$，那么分两种情况讨论：

- $(u,v)\notin E$。

  于是存在大小不超过 $k-1$ 的集合 S 割掉 u,v，从而存在 $x\in(A\cap B)\setminus S$。

  因为 $\kappa^*(u,x)\geq k, \kappa^*(v,x)\geq k$。因此在图 $G[V\setminus S]$ 中 u,x 连通，v,x 连通，从而 u,v 连通，矛盾。

- $(u,v)\in E$。

  于是存在大小不超过 $k-2$ 的集合 S，使得 $G[V\setminus S]$ 中 (u,v) 是大小为 1 的边割集，从而存在 $x\in(A\cap B)\setminus S$。

  因为 $\kappa^*(u,x)\geq k$，因此 $G[V\setminus S]$ 中 $\kappa^*(u,x)\geq 2$，于是即使删去边 (u,v) 后 u,x 仍连通。同理 v,x 也连通，于是删去 (u,v) 后 u,v 连通，矛盾。

即任意两个极大团的交至多有 $k-1$ 个顶点。 □

2.6 总结

本节主要介绍了对一般的正整数 k，k–点 (边) 连通性的判断和点 (边) 连通分量的一些可行方法。

事实上，对于无向图边连通所建立的网络，每条边权均为 1，因此使用 Dinic 算法就有 $O(\text{Flow}(|V|,|E|))=O\left(|E|^{\frac{3}{2}}\right)$ [10]。当然，如果我们只需判定最大流是否 $\geq k$，第一项还可以对 k 取 min，即 $O\left(\min\left\{k,\sqrt{|E|}\right\}|E|\right)$。

而对于点连通所建立的网络流，不难发现形如 x 的点只有一条出边，形如 x' 的点只有一条入边，因此此时使用 Dinic 算法就有 $O(\text{Flow}(|V|,|E|))=O\left(\sqrt{|V|}\cdot|E|\right)$。同样，如果只需判定是否 $\geq k$，复杂度可降为 $O\left(\min\left\{k,\sqrt{|V|}\right\}|E|\right)$。

下面将上述所有的算法列成一张表，其中假设所有的网络流使用最常见的 Dinic 算法：

(注 1：对于其中的某些问题，学术界可能存在更优的做法，不过由于实现过于复杂或实际意义不大等原因这里就不给出了。本节讨论的算法都是相比之下较为实用的算法)

(注 2：对于某些问题，需要通过具体的 $k,|V|,|E|$ 来得出复杂度，因为对应的界比较多 (如 $\kappa=O\left(\frac{|E|}{|V|}\right)$, $\text{Flow}(|V|,|E|)=O\left(\min\left\{k,\sqrt{|V|}\right\}|E|\right)$，等等)。)

[10]特别地，如果原无向图是简单图，则还有一个更紧的上界：$O\left(\min\left\{|V|^{\frac{2}{3}},\sqrt{|E|}\right\}|E|\right)$。

算法	点连通	边连通	其它算法
局部 k–连通判定 (单次)	$O\left(\min\left\{k, \sqrt{\lvert V\rvert}\right\}\lvert E\rvert\right)$	$O\left(\min\left\{k, \sqrt{\lvert E\rvert}\right\}\lvert E\rvert\right)$	-
局部连通度 (单次)	$O\left(\sqrt{\lvert V\rvert}\cdot\lvert E\rvert\right)$	$O\left(\lvert E\rvert^{\frac{3}{2}}\right)$	-
全局 k–连通判定	$O\left(k\min\left\{k, \sqrt{\lvert V\rvert}\right\}\lvert V\rvert\lvert E\rvert\right)$	$O\left(\min\left\{k, \sqrt{\lvert E\rvert}\right\}\lvert V\rvert\lvert E\rvert\right)$	[边, 确定性] $O\left(\lvert V\rvert^2\log\lvert V\rvert+\lvert V\rvert\lvert E\rvert\right)$ [边, 随机化] $O\left(\lvert V\rvert^2\log^3\lvert V\rvert\right)$
全局连通度	$O\left(\sqrt{\lvert V\rvert}\cdot\lvert E\rvert^2\right)$	$O\left(\lvert V\rvert\lvert E\rvert^{\frac{3}{2}}\right)$	
k–连通分量划分	-	$O\left(\min\left\{k, \sqrt{\lvert E\rvert}\right\}\lvert V\rvert\lvert E\rvert\right)$	-
局部 k–连通判定 (多次)	$O\left(\min\left\{k, \sqrt{\lvert V\rvert}\right\}\lvert V\rvert^2\lvert E\rvert\right)-O(1)$	$O\left(\min\left\{k, \sqrt{\lvert E\rvert}\right\}\lvert V\rvert\lvert E\rvert\right)-O(1)$	-
局部连通度 (多次)	-	$O\left(\lvert V\rvert\lvert E\rvert^{\frac{3}{2}}\right)-O(1)$	-

3 1–连通与 2–连通

接下来我们着重讨论 k 较小的情形。

本文将讨论 $k=1,2,3$ 的情形。对于 $k=4$ 的情形，虽然学术界有对应的算法，不过由于实用性不大，而且实际表现并**不见得**比之前一般 k 的算法优秀，这里就暂且略去了。

$k=3$ 的情形相较于 $k=1,2$ 更为复杂，因此本节先讨论 $k=1,2$ 的情形。

3.1 1–连通

由定义立即可知，对于无向图 G，G 是连通图当且仅当 G 是 1–点连通图，当且仅当 G 是 1–边连通图。

因此问题转化为连通性的相关问题，使用 dfs、bfs 或并查集的技巧即可完成，这里不再赘述。

3.2 2–连通

2–连通分为两种：2–点连通和 2–边连通。在讨论 2–连通之前，我们有必要提及一下图的 dfs 生成树 (dfs 树)。

定义 3.2.1 (dfs 生成树)**.** *对于一张连通无向图 $G=(V,E)$，任取一个点 r 为根进行深度优先搜索 (dfs)，则每个顶点 $v\in V\setminus\{r\}$ 在 dfs 上都有一个前趋顶点 (记为 p_v)，所有形如 (v,p_v) 的边构成一棵树，称为图 G (以 r 为根) 的***一棵 dfs 生成树***，简称 dfs 树，记为 T_r。dfs 树上的边称为***树边***，其它的边称为***非树边***。*

dfs 树可以看成无根树也可以看成有根树，不过为了方便起见一般看成以 r 为根的有根树。通常情况下，dfs 树可以看成外向树 (根指向叶子)、内向树 (叶子指向根) 和无向树，具体要根据上下文决定。

对于不连通的图 G，我们需要对每个连通分量进行 dfs，从而可以类似得到一个 dfs 森林。

下面的这个定理，是无向图 dfs 树的重要性质：

定理 3.2.1. *设 T_r 连通无向图 G 的一棵 dfs 树，则所有的非树边都连接 T_r 的某个顶点和其祖先顶点。* □

因为这个原因，我们也把无向图的非树边称为**返祖边** (返回边，回边，back-edge)。

3.3 2–边连通[11]

首先来看 2–边连通，2–边连通也称为**边双连通**，2–边连通分量也称为**边双连通分量**。

3.3.1 2–边连通分量和桥边

定义 3.3.1 (桥边)**.** *若单边集 $\{e\}$ 为连通图 G 的某个边割集，则称 e 是 G 的***桥边*** (bridge)，也称割边。*

2–边连通有如下独有的性质：

定理 3.3.1. *若 B 是连通图 G 的一个 2–边连通分量，则 $G[B]$ 是 2–边连通图。*

证明. *取 $u,v \in B$，由 $\kappa(u,v) \geq 2$ 知存在从 u 到 v 两条边不相交的路径。*

于是我们可以得到经过 u,v 的有向回路，从而回路上的所有点两两 2–边连通。于是 u,v 在 $G[B]$ 中连通且 2–边连通。 □

推论 3.3.1. *B 是 G 的 2–边连通分量当且仅当 $G[B]$ 是 G 的极大 2–边连通子图。* □

考虑连通图 G 的 dfs 树，对于每条非树边 e，由定理 3.2.1 知它是返祖边，连接某个点 v 和其祖先顶点 u。对于这种情形，我们称 e ***覆盖***了路径 $u \rightsquigarrow v$ 上所有的树边。

定理 3.3.2. *对于连通无向图 G，e 是桥边当且仅当 e 是树边且 e 没有被任意一条返祖边覆盖。* □

[11] 本小节内容默认假设图是连通图。

于是可以使用树上差分的技巧计算出每条边是否被返祖边**覆盖**，来得到所有的桥边。

考虑所有**被覆盖的树边**构成的图，它们是一个森林。容易发现，这个森林中的每个连通分量对应 G 的一个 2–边连通分量。

从而找出所有桥边后，将它们去掉后，剩下的边 (或树边) 构成的连通分量，就是原图的 2–边连通分量。

上述算法的时间复杂度为 $O(|V|+|E|)$。

我们尝试发掘更深的性质。

考虑有根树上的一个连通子图 S，则 S **中有唯一的顶点具有最小的深度**。

于是，每个 2–边连通分量，对应到 dfs 树上是一个连通子图，于是它存在深度最小的顶点。

对于 $v \in G$，定义 low_v，表示以 v 为根的子树的点 u 中，通过返祖边 (u,w) 能到达的点 w 中深度最小者。

根据上述定义可知，每个 2–边连通分量都有且仅有一个点满足 $v = low_v$，且这个 v 就是深度最小的顶点。

为了快速完成这个任务，同时避免记录 (深度，标号) 二元组的麻烦，我们可以引入 dfs 序——dfs 时访问顶点的顺序。**下文用 seq_i 表示 dfs 时访问的第 i 个顶点，id_i 或 dfn_i 表示顶点 i 访问的时刻** (即第几个被访问)。容易发现，seq_i 和 id_i 互为逆置换的关系。

dfs 序的一个重要性质是：

定理 3.3.3. *若 u 是 dfs 树中 v 的祖先，则 $id_u \le id_v$。* □

注意到 low_v 一定是 v 的祖先，因此定义中可以被换成标号最小者。方便起见，不妨设 dfs 序就是 $1,2,\cdots,n$。

现在，考虑对每个 v 求出 low_v，可以使用**树形 DP**：

- 考虑和 v 关联的所有返祖边，对于每个 (v,u)，令 $low_v = \min\{low_v, u\}$。
- 考虑 v 的所有子节点，对于每个子节点 c，令 $low_v = \min\{low_v, low_c\}$。

当遇到 $v = low_v$ 的顶点时，将以 v 为根的子树取出来，作为一个 2–边连通分量，并将该子树删去。同时，如果 v 存在父节点 p_v，则边 (p_v, v) 是桥边。

这就是 **Tarjan 算法**，时间复杂度 $O(|V|+|E|)$。

3.3.2 应用——Robbins 定理

2–边连通图的一个实际应用是 Robbins 定理——2–边连通图的强连通定向。

定理 3.3.4 (Robbins). *无向简单图 G 能定向成强连通 (有向) 图的充分必要条件是，G 是 2–边连通图。*

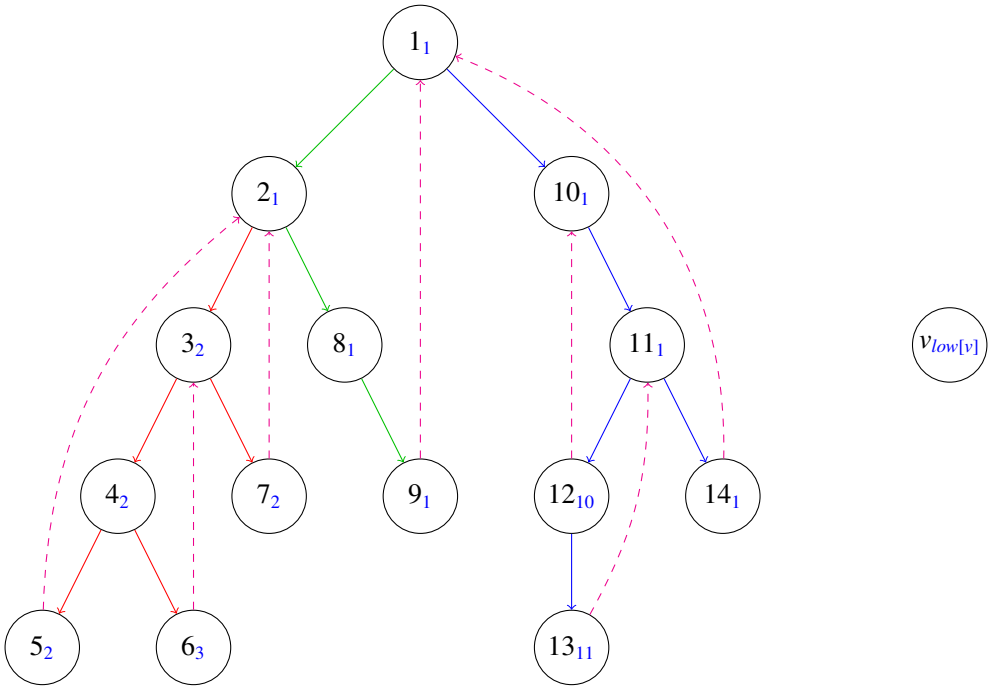

图 5

证明. 必要性：若 G 不是 2–边连通图，则 G 存在桥边 (u,v)，于是无论如何定向，都不可能同时存在 $u \rightsquigarrow v$ 的有向路径和 $v \rightsquigarrow u$ 的有向路径，矛盾。

充分性：设 G 是 2–边连通图，设它的一棵 *dfs* 树为 T，将所有树边按照外向树定向 *(*父节点指向子节点*)*，所有返祖边定向为“后代指向祖先”。下面证明这是一个强连通定向。

首先，由外向树的性质知根可以通向所有节点，因此只需要证明每个顶点 v 也可以通向根。

又因为 G 是 2–边连通图，因此在 *Tarjan* 算法中有 $\underbrace{low\,[low\,[\cdots low\,[v]\cdots]]}_{\infty} = 1$。[12]

而由 *Tarjan* 算法和定向规则可知，v 可以通向 low_v，从而可以不断通过此规则回到点 1 *(*即根节点*)*，证毕。 □

上述证明同时也给出了非常简单且容易实现的构造，时间复杂度 $O(|V|+|E|)$。

3.4 2–点连通[13]

和边连通类似，2–点连通也称为**点双连通**。

定义 3.4.1 (割点)**.** 若单点集 $\{v\}$ 为连通图 G 的某个点割集，则称 v 是 G 的割点 *(cut vertex)*。

[12]这里不妨设 dfs 序为 $1,2,\cdots,n$。

[13]本小节内容默认假设图是连通图。

3.4.1 2–点连通分量和点双连通分量

但是，由于对 2 阶完全图 K_2 点连通度的定义问题点双连通分量的定义会有少许区别。**为了方便起见，我们人为规定 K_2 是 2–点连通图，即 $\kappa(K_2)=2$。**

定义 3.4.2 (点双连通分量)**.** *对于连通无向图 $G=(V,E)$，对于 $B\subseteq V$，若 $G[B]$ 是 G 的极大 2–点连通子图，则称 B 是 G 的*点双连通分量。

实际上，当定义 $\kappa(K_2)=1$ 时，点双连通分量的定义和 2–点连通分量是完全一致的。而当 $\kappa(K_2)=2$ 时，所有的 2–点连通分量仍然是点双连通分量，多出来的点双连通分量均为二元集 $\{u,v\}$，满足 (u,v) 是 G 的桥边。当 $|V|>1$ 时，所有点双连通分量都至少有两个点。

在后面的过程中，会逐渐发现，这么定义点双连通分量对整个算法流程是有益的。

定理 2.5.2 告诉我们，n 个点的图至多有 n 个 2–点连通分量。事实上，可以证明，n 个点的图也至多有 n 个点双连通分量。

3.4.2 边的等价性

由于点连通对点来说不是等价关系，但 2–点连通对边来说是一种等价关系：

在 E 上定义二元关系 ρ_c，其中 $(e,f)\in\rho_c$ 当且仅当它们可以同时在一个圈中。

定理 3.4.1. *$(e,f)\in\rho_c$ 当且仅当 e,f 在同一个点双连通分量中。*

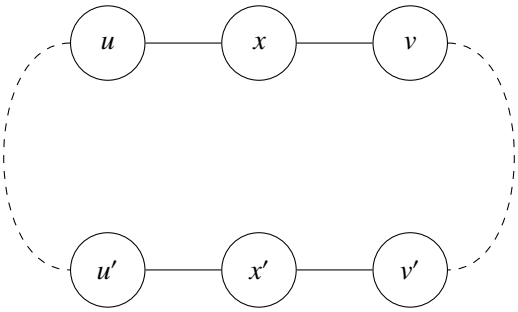

图 6

证明. *若 $(e,f)\in\rho_c$，则 e,f 同时在一个圈中，于是 e,f 的所有端点都是两两 2–点连通的，从而这些端点在同一个点双连通分量中。*

考虑点双连通图中的两条边 $e=(u,v),f=(u',v')$。我们将 e 换成 (u,x) 和 (x,v)，f 换成 (u',x') 和 (x',v') (裂边)。

则新图仍然是点双连通的，于是至少两条从 x 到 x' 的路径，它们拼起来就可以构成一个圈。由 x 和 x' 的构造方式知，这个圈经过边 e 和 f。 □

现在让我们回到 dfs 树。根据上述结论知，dfs 树上的 $n-1$ 条树边可以根据 ρ_c 划分成若干个等价类。如图 5 中每种颜色的树边表示一个 ρ_c 的等价类。

考虑一个等价类，由之前的结论知它是一个点双连通分量中的所有树边，因此它必然对应 dfs 树上的一个连通块。

于是，我们对于每条返祖边 (v,u) (u 为 v 的祖先)，将这些路径上的所有边设为等价，最终树上每一种等价类对应的连通子图就是原图的一个点双连通分量。

对于一个点双连通分量 B，在 dfs 树上仍然是一个连通子图，且其中有唯一顶点具有最小深度。

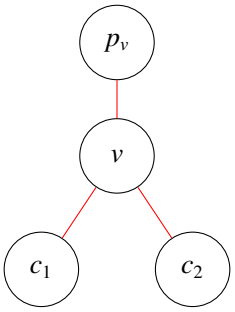

图 7

如图 7，设该顶点为 v，则 v 的子节点中只有**至多一个**属于 B。否则设 $c_1, c_2 \in B$，则边 (c_1,v) 和边 (c_2,v) 的等价，于是它们必然和 (v,p_v) 等价，这与 v 深度的最小性矛盾。

设满足该条件的唯一子节点为 u，则必然有 $v \le low_u$[14]。反之，如果一个顶点 v 和其子节点 u 满足 $v \le low_u$，则此时 u 所在的子树 (的所有边) 连同边 (u,v) 共同构成了 ρ_c 的等价类，即我们得到了一个点双连通分量。于是我们将以 u 为根的子树删去，然后重复运行即可。

这就是 Tarjan 求点双连通分量的算法，时间复杂度 $O(|V|+|E|)$。

3.4.3 割点

如何判断一个点 v 是否为割点呢？我们有如下结论：

定理 3.4.2. *点 v 不是割点，当且仅当和 v 关联的所有边在同一个 ρ_c 等价类中 (颜色相同)。* □

于是，如果这些边不全在同一个等价类中，则必然有一条边 (u,v) (u 为 v 的子节点) 与 (v,p_v) 不在一个等价类，从而存在 u 满足 $v \le low_u$。

当然，有一个例外，v 是根节点。对于根节点的情况，可以注意到在 dfs 树中，与根节点关联的所有边两两不 ρ_c 等价，因此对于根节点**只需要判断**它是否有两棵及以上子树即可。

该算法的时间复杂度仍为 $O(|V|+|E|)$。

[14] 当 (v,u) 为桥边时有 $v < low_u$，否则有 $v = low_u$，总之有 $v \le low_u$。

3.4.4 应用——块割树和圆方树

不同的点双连通分量可能有交，因此我们在求解时不得不使用边的等价类来处理。

那点双连通分量的主角还是点，对那些点来说又有哪些性质呢？

首先，由定理 2.5.3 知，两个点双连通分量的交集至多只有 1 个点。事实上，如果两个点双连通分量的有交点 v，则 v 必定是原图的割点。

定义 3.4.3 (块割树)**.** *对连通无向图 G，我们可以定义图 $H = (B \cup C, J)$，满足：*

- *B 是 G 中所有点双连通分量 (作为点集) 构成的集合。*
- *C 是 G 中所有割点构成的集合。*
- *$J = \{(b, c) \mid b \in B, c \in b \cap C\}$。*

*则称 H 为 G 的*块割树[15]。

对于图 8 所示的这张图，一共有 5 个 2–点连通分量和 7 个点双连通分量。它的块割树如图 9：

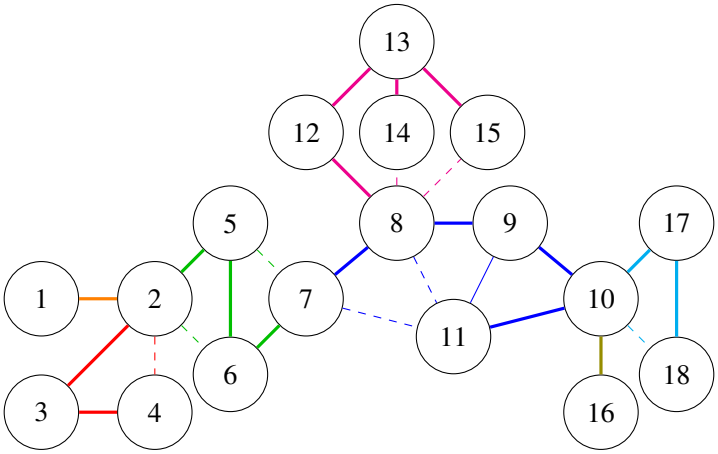

图 8

不难证明连通无向图的块割树一定是树。

得到一张图的块割树也是不难的。由于割点一定在块割树上，因此我们只需以任意一个割点为根进行 dfs，然后每次找到一个 (最浅点为 v 的) 点双连通分量 B 时，将 B 中的所有割点连一条边向 B，然后将 B 连向 v。

不过在实际应用中，我们不仅仅只要割点，其它的点也需要考虑进来。因此通常会使用块割树的广义版本——圆方树。

[15]block-cut tree, 见 [7]。

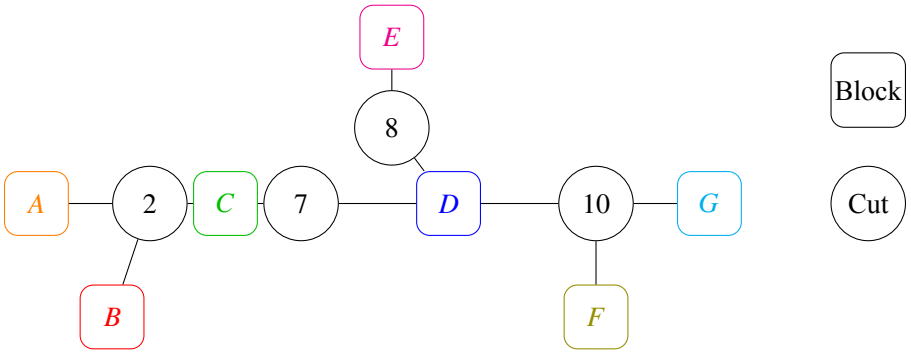

图 9

定义 3.4.4 (圆方树)**.** 对连通无向图 $G = (V, E)$，我们可以定义图 $H = (B \cup V, J)$，满足：

- B 是 G 中所有点双连通分量 *(作为点集)* 构成的集合。
- $J = \{(b, v) \mid b \in B, v \in b\}$。

则称 H 为 G 的**圆方树**。其中 B 中的点称为**方点** *(块点)*，V 中的点称为**圆点** *(普通点)*。

如，图 8 的圆方树是图 10。

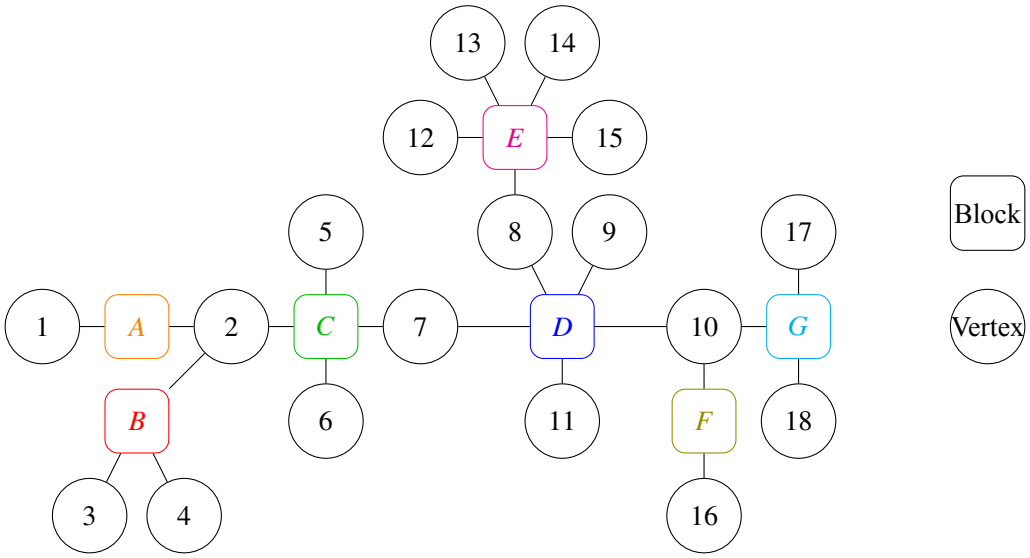

图 10

由定义可知，

定理 3.4.3. 块割树是圆方树中所有非叶节点的导出子图，它们的每条边连接一个圆点和一个方点。 □

和块割树类似，圆方树也可以通过一次 Tarjan 算法在 $O(|V|+|E|)$ 时间内得到。圆方树可以将许多和图上路径有关的问题转化为对应的树上问题，从而使用传统的树上算法处理，是一个化“图”为“树”的利器。

4 3–连通及其应用

3–连通是图论中较为现代的理论，同样分为 3–边连通和 3–点连通。同样它们各有各自的算法及应用。

4.1 3–边连通[16]

4.1.1 切边和切边对

定义 4.1.1 (切边, 切边对)**.** *对于 2–边连通的无向图 G，若某个二元边集 $\{e_1,e_2\}$ $(e_1\neq e_2)$ 是 G 的边割集，则称 (e_1,e_2) 是一对*切边对 *(cut pair)，其中 e_1,e_2 都被称为*切边 *(cut edge)。*

切边对具有如下性质：

定理 4.1.1. *对于 2–边连通无向图 $G=(V,E)$，$e_1,e_2\in E$ $(e_1,e_2$ 不是同一条边$)$。则 (e_1,e_2) 为切边对当且仅当对于 G 中的每个圈 C，要么 $e_1\in C\wedge e_2\in C$，要么 $e_1\notin C\wedge e_2\notin C$ $($即 $(e_1\in C)\Leftrightarrow(e_2\in C)$[17]$)$。*

证明. 充分性：若 e_1,e_2 满足对在所有圈中要么同时出现或同时不出现，这说明删去 e_1 后，e_2 不能在任何一个圈上。

从而 e_2 是 $G\setminus\{e_1\}$ 的桥边，即 (e_1,e_2) 是切边对。

必要性：若 (e_1,e_2) 是切边对，则 e_2 是 $G\setminus\{e_1\}$ 的桥边。

这说明，在 G 中通过 e_2 的圈必定通过 e_1，同理通过 e_1 的圈必定通过 e_2，即它们在所有圈中要么同时出现，要么同时不出现。 □

4.1.2 切边等价

从而我们可以引入**切边等价**的概念：

定义 4.1.2 (切边等价)**.** *对于 2–边连通无向图 $G=(V,E)$ 和 $e_1,e_2\in E$ $(e_1$ 和 e_2 不必不同$)$，如果对 G 中的每个圈 C，都有 $(e_1\in C)\Leftrightarrow(e_2\in C)$，则称 e_1,e_2 **切边等价**，记作 $e_1\overset{c}{\sim}e_2$。*

由定理 4.1.1 可知，当 $e_1\neq e_2$ 时，$e_1\overset{c}{\sim}e_2$ 当且仅当 (e_1,e_2) 是切边对。

[16]本小节内容默认假设图是 2–边连通图，允许有重边，但不允许有自环。

[17]$\Leftrightarrow$ 表示逻辑等价，见 [9]。

定理 4.1.2. *切边等价是等价关系。*

证明. 显然切边等价具有自反性和对称性。现在设 $e_1 \overset{c}{\sim} e_2, e_2 \overset{c}{\sim} e_3$，则对于 G 中的每个圈 C，都有 $(e_1 \in C) \Leftrightarrow (e_2 \in C), (e_2 \in C) \Leftrightarrow (e_3 \in C)$。由 $\Leftrightarrow$ 的传递性知 $(e_1 \in C) \Leftrightarrow (e_3 \in C)$，从而 $e_1 \overset{c}{\sim} e_3$。 □

于是，“切边等价”关系将边集 E 划分成若干个等价类 $E_1, E_2, \cdots, E_n$。

4.1.3 和 3–边连通的关系

定义 4.1.3 (收缩)**.** *对于无向图 $G = (V, E)$ 和边 $e = (u, v) \in E$，定义 G 对 e 收缩 (edge contraction) 所得的图为 $H = (V', E')$，其中：*

- *定义一个新的顶点 w，则 $V' = (V \setminus \{u, v\}) \cup \{w\}$。*
- *对于 E 中的边 (x, y)，若 $x \notin \{u, v\} \wedge y \notin \{u, v\}$，则 (x, y) 也在 E' 中；否则不妨设 $x \in \{u, v\}$，则将对应的边换成 (w, y)。*
- **最终去掉所得图中的所有自环。**[18]

G 对 e 收缩的图记为 $G \cdot e$ 或 G/e。

图 11 即为一次收缩操作的例子 (图来源 [10])。

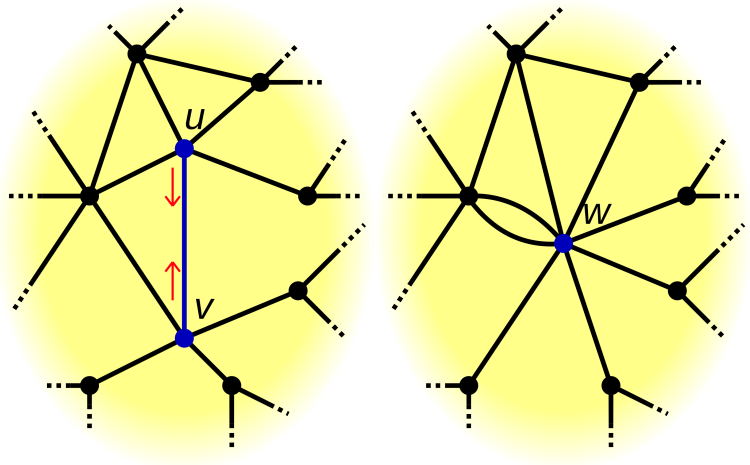

图 11

注意，简单图经过收缩后可能成为非简单图 (如图 11)。

切边等价和 3–边连通的关系由下面两个定理建立：

[18]这是 3–边连通所需要的特殊要求，也为了简化问题所考虑。

定理 4.1.3. 设 G 是 2–边连通无向图。若边 $e \in (u,v)$ 满足它所在的 $\overset{c}{\sim}$ 等价类大小为 1，则 u,v 在同一个 3–边连通分量中。

同时，令 $H = G \cdot e$，则 H 的 3–边连通分量划分就是将 G 的 3–边连通分量划分将 u,v 替换为 w 后的结果，且 H 和 G 中对应边具有相同的切边等价关系。

证明. 上述定理包含三个命题，我们逐一证明。

1. 若 $e = (u,v)$ 所在的等价类大小为 1，说明 e 不是切边。

   于是由切边的定义知 $\lambda(u,v) \geq 3$，即 u,v 在同一个 3–边连通分量中。

2. 设 $\lambda(x,y) \geq 3$，如果 $x \notin \{u,v\} \wedge y \notin \{u,v\}$，由 *定理 2.1.1* 知，存在 3 条从 x 到 y 边不相交的路径。同时，由收缩过程知对应的路径仍然存在 *(注意收缩是保留重边的)*，因此在 H 中仍然存在 3 条从 x 到 y 边不相交的路径。即 $\lambda(x,y) \geq 3 \Rightarrow \lambda(x',y') \geq 3$。当 $x,y \in \{u,v\}$ 时同理可证。

   若 $\lambda(x',y') \geq 3$ 且 $\lambda(x,y) < 3$，说明 x,y 存在大小为 2 的边割。由于 (u,v) 不可能作为切边，因此这两条边在 H 中都是对应存在的。

3. 注意到 G 中的每个圈可以通过收缩操作对应到 H 中的每个圈，反之亦然，于是 G,H 具有相同的切边等价关系。

□

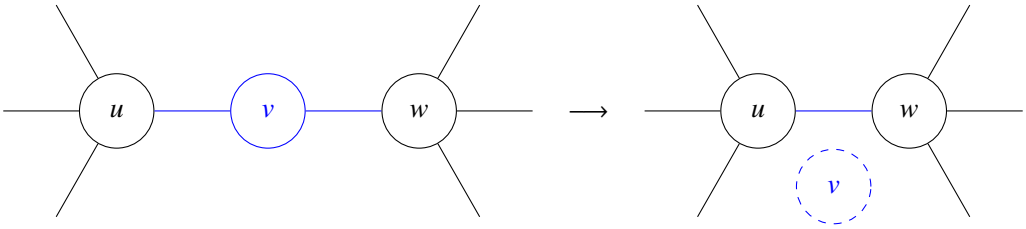

图 12

定理 4.1.4. 设 G 是 2–边连通无向图，其中 $d(v) = 2$。设 $N(v) = \{u,w\}$ *(其中 u 可以等于 w)*，$\{v\}$ 为一个单独的 3–边连通分量。

同时，令 H 为在 $G \setminus \{v\}$ 中，加入边 (u,w) 的图 *(如果 $u = w$ 则无需添加自环)*。则 H 的 3–边连通分量划分就是将 G 的 3–边连通划分将 $\{v\}$ 删去后的结果，且对应边具有相同的切边等价关系。

证明. 同样依次证明上述三个命题。

1. 对于 $\forall x \in V \setminus \{v\}$，由于 $(u,v),(w,v)$ 均为 v 和 x 的大小为 2 的点割集，因此 $\lambda(v,x) \leq 2$。于是 $\{v\}$ 自成一个 3–边连通分量。

2. 设 $\lambda(x,y) \geq 3$ $(x \neq v \wedge y \neq v)$，由定理 2.1.1 知存在 3 条从 x 到 y 边不相交的路径。我们将 (u,v) 和 (v,w) 替换为 (u,w) 即可找到 3 条 H 中对应的路径，反之亦然。故 $\lambda_G(x,y) \geq 3 \Leftrightarrow \lambda_H(x',y') \geq 3$。

3. G 中的每个圈仍然可以通过将 (u,v) 和 (v,w) 替换为 (u,w) 对应到 H 中的每个圈，反之亦然，故切边等价关系不变。

□

4.1.4 一个想法

反复利用定理 4.1.3 和定理 4.1.4，我们可以将一张图的规模不断缩小，从而得到原图的 3–边连通分量划分。

事实上，这个思路是可行的，我们需要依赖如下定理：

定理 4.1.5. 设 G 为 2–边连通无向图，满足 $\delta(G) \geq 3$，则必定存在一条边 (u,v)，满足 (u,v) 不和其它任何边切边等价 (即等价类大小为 1)。

在证明这个定理前，我们需要介绍切边等价和 dfs 树的关系。

设 T 为 2–边连通无向图 G 的 dfs 树，由之前的部分知每条返祖边**覆盖**了若干条树边。我们对于每条边 e，定义一个集合 $\mathsf{C}(e)$：

- 若 e 是返祖边，则 $\mathsf{C}(e) = e$。
- 若 e 是树边，在令 $\mathsf{C}(e)$ 为所有**覆盖**它的返祖边的集合。形式化地，

$$\mathsf{C}(e) = \left\{(u,v) \mid (u,v) \in G \setminus T; u,v \text{ 在 } T \setminus \{e\} \text{ 中不连通}\right\}$$

可以发现，$\mathsf{C}(e)$ 中的元素全是返祖边。由于 G 是 2–边连通图，因此由定理 3.3.2 知 $\forall e \in E, \mathsf{C}(e) \neq \varnothing$。

事实上，集合 $\mathsf{C}(e)$ 和切边等价有着密切的关系：

定理 4.1.6. 对于 2–边连通无向图 $G = (V,E)$ 和 $e_1, e_2 \in E$，$e_1 \overset{c}{\sim} e_2$ 当且仅当 $\mathsf{C}(e_1) = \mathsf{C}(e_2)$。

证明. 考虑任意一条返祖边 (u,v)。由定义知，所有满足 $(u,v)\in \mathsf{C}(e)$ 的边 e 构成 G 中的一个圈 C。

如果 $e_1 \overset{c}{\sim} e_2$，那么对于该圈 C，有 $(e_1\in C)\Leftrightarrow(e_2\in C)$，从而 $(u,v)\in\mathsf{C}(e_1)\Leftrightarrow(u,v)\in\mathsf{C}(e_2)$，由 (u,v) 的任意性知 $\mathsf{C}(e_1)=\mathsf{C}(e_2)$。

反之，若 $\mathsf{C}(e_1)=\mathsf{C}(e_2)$，则对于每条**只包含一条返祖边**的圈 C，均有 $(e_1\in C)\Leftrightarrow(e_2\in C)$。

注意到若对于边集 A,B，成立 $(e_1\in A)\Leftrightarrow(e_2\in A)$，则对于 $A\oplus B$[19]，也成立 $(e_1\in A\oplus B)\Leftrightarrow(e_2\in A\oplus B)$。

由连通图的**圈空间** [11] 的性质知，G 中的每一个圈均可以表示成若干个**只包含一条返祖边**的圈的对称差，于是对于 G 中的每个圈 C，都有 $(e_1\in C)\Leftrightarrow(e_2\in C)$，即 $e_1\overset{c}{\sim}e_2$。 □

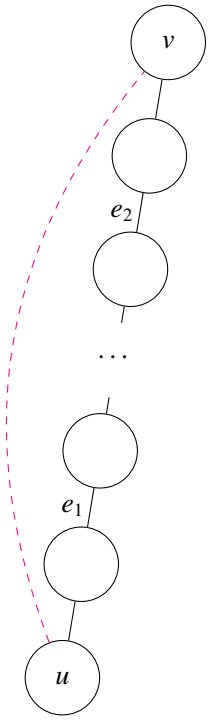

图 13

引理 4.1.1. 设 G 为 2–边连通无向图。若**树边** e_1,e_2 切边等价 $(e_1\overset{c}{\sim}e_2)$，则这两条边一定是祖先–后代关系。

证明. 设 $e_1=(u,p_u),e_2=(v,p_v)$。由于 $e_1\overset{c}{\sim}c_2$，由*定理 4.1.6* 知 $\mathsf{C}(e_1)=\mathsf{C}(e_2)\neq\varnothing$。

任取 $e\in\mathsf{C}(e_1)$，知非树边 e 覆盖 e_1,e_2，而由*定理 3.2.1* 知 e 是返祖边，从而 e_1,e_2 一定是祖先–后代关系。 □

引理 4.1.2. 设 $\mathcal{P}$ 是 *dfs* 树上一条从根到某个顶点的路径。则以下情形不会出现：$\mathcal{P}$ 中**按顺序**存在四条边 e_1,e_2,e_3,e_4，满足 $e_1\overset{c}{\sim}e_3,e_2\overset{c}{\sim}e_4$，但 $e_1\overset{c}{\not\sim}e_2$。

证明. 反设存在这种情况，如图 14，考虑 $e\in\mathsf{C}(e_1)$，知 $e\in\mathsf{C}(e_3)$，从而 e 覆盖 $e_1,e_3\Rightarrow e$ 覆盖 e_2，即 $e\in\mathsf{C}(e_2)\Rightarrow e\in\mathsf{C}(e_4)$。

同理，$e\in\mathsf{C}(e_4)\Rightarrow e\in\mathsf{C}(e_1)$，于是 $\mathsf{C}(e_1)=\mathsf{C}(e_4)$，$e_1\overset{c}{\sim}e_2$，矛盾。 □

下面就可以证明定理 4.1.5 了。

证明 (4.1.5)**.**

考虑所有 $\overset{c}{\sim}$ 等价类中，最浅的树边的上端顶点最深的那个等价类。

设该等价类中最浅的树边为 $e=(v,p_v)$，则**依次证明**：

- 树中没有其它的边 f 满足 $e\overset{c}{\sim}f$。

  反之，f 在以 v 为根的子树中。设 $f=(u,p_u)$，如果 $p_u\neq v$，由*引理 4.1.2* 知中间其它边所在的等价类，最浅的树边更加深。

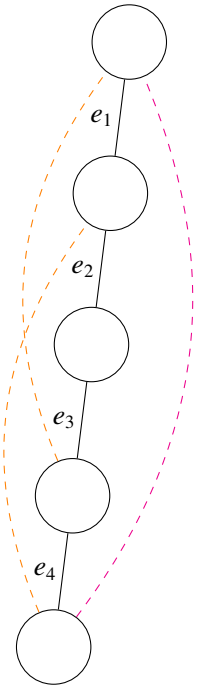

图 14

[19] $\oplus$ 在这里表示集合的对称差。

如果 $p_u = v$，由于 $d(v) \geq 2$，因此 v 不能引其它返祖边，则 v 必定还有其它子节点 y。于是 (v,y) 所在等价类中，(v,y) 作为最浅树边，与深度最大性矛盾。

- 没有返祖边 f 满足 $e \overset{c}{\sim} f$。

  否则，说明只有返祖边 $f = (a,b)$ *(b 为 a 的祖先)* 覆盖了 e。若 $b \neq u$，则 (b, p_b) 所在等价类的深度比 e 大，若 $b = u$，则有 $d(v) \geq 2$ 知 v 还有其它子节点 y，于是 (v,y) 所在等价类的深度比 e 大，矛盾。

于是 e 所在的等价类大小为 1，结论成立。 □

定理 4.1.5 告诉我们，在任一时刻，要么有一个点的度为 2，要么存在一个大小为 1 的边等价类。对于这两种情况，我们可以利用定理 4.1.4 和定理 4.1.3，不断将问题转化为规模更小的子问题。

4.1.5 实现

下面介绍如何实现这个算法。

首先需要能够快速判断两条边是否切边等价，由定理 4.1.6 知我们只需判断它们对应的 $\mathsf{C}(e)$ 是否相同。

而获取每条边的 $\mathsf{C}(e)$ 需要 $O(|E|)$[20] 的时间，又只有树边定义的 $\mathsf{C}(e)$ 元素个数才会 ≥ 1，从而获取所有边的 $\mathsf{C}(e)$ 需要 $O(|V||E|)$ 的时间。

这个显然是我们无法接受的，因此考虑利用经典的随机化技巧：对每一条非树边 e，随机一个 64 位权值 $w(e)$，然后定义[21]

$$\mathsf{c}(e) = \bigoplus_{f \in \mathsf{C}(e)} w(f)$$

于是当 $\mathsf{C}(e_1) = \mathsf{C}(e_2)$ 时一定有 $\mathsf{c}(e_1) = \mathsf{c}(e_2)$，而当 $\mathsf{C}(e_1) \neq \mathsf{C}(e_2)$ 时 $\mathsf{c}(e_1) = \mathsf{c}(e_2)$ 的概率只有 $\dfrac{1}{2^{64}}$ 量级，可以忽略不计。

现在，我们将问题转化为了对树上的一条链异或上一个数，最终询问每条边的值，这是一个静态问题，可以使用树上差分的技巧在 $O(|V| + |E|)$ 时间内解决。

设 $G = (V, E)$ 是 2–边连通图，算法的流程如下：

[20] 由于 G 是连通图，因此 $|V| \leq 1 + |E|$。

[21] $\oplus$ 在这里表示两个数的按位异或。

1. 对于每条边 e，求出 $\mathsf{c}(e)$，以判断两条边是否切边等价。
2. 令 $A = \left\{e \mid e \in E, e \text{ 所在的切边等价类的大小为 } 1\right\}, D = \{v \mid v \in V, d(v) = 2\}$ **(以下默认边集是可重集，点集不是可重集)**。
3. 对于每个顶点 v，维护其当前点度 d_v 和它所在的“3–边连通分量”L_v，起初 $d_v = d(v), L_v = \{v\}$，再使用并查集维护辅助图 H。
4. 若 $A = \varnothing$ 且 $D = \varnothing$，则转到步骤 15。
5. 若 $A \neq \varnothing$，则转到步骤 6，否则转到步骤 10。
6. 任取 $e = (u_0, v_0) \in A$，令 $A = A \setminus \{e\}$。
7. 设 u, v 分别为 u_0, v_0 在 (并查集) H 中所在的连通分量。
8. 若 $u = v$，则说明这两边的连通分量已经处理，因此直接令 $d_u = d_u - 2$ 即可。

   否则，令 $L_u = L_u \cup L_v, L_v = \varnothing, d_u = d_u + d_v - 2$，在 H 中将 u, v 所在的连通分量合并。
9. 检查此时是否有 $d_u = 2$，如果是则令 $D = D \cup \{u\}$，回到步骤 4。
10. 任取 $x \in D$，令 $D = D \setminus \{x\}$。

    如果 $d_x \neq 2$，则回到步骤 4。[22]
11. 枚举 L_x 中的顶点，来找到当前与它关联的两条边，记为 (x, u) 和 (x, v)。
12. 如果 u, v 已经在 H 中同一个连通分量中，则回到步骤 4。
13. 将边 (x, u) 和 (x, v) 合并为 (u, v)。
14. 检查此时 (u, v) 所在的切边等价类大小是否为 1，如果是则令 $A = A \cup \{(u, v)\}$，回到步骤 4。
15. 此时，所有非空的 L_i 构成 G 的 3–边连通分量一个划分。

这里是上述算法的一个实现，可以看出代码不是很长。

[22] 这是因为可能经过某些操作后原本度数为 2 的点的度数又不是 2 了。

4.1.6 演示

下面用粉色点表示 2 度点，不同颜色的边表示不同 $\overset{c}{\sim}$ 等价类，**黑色边表示大小为 1 的等价类**，紫色表示已完成划分的 3–边连通分量。

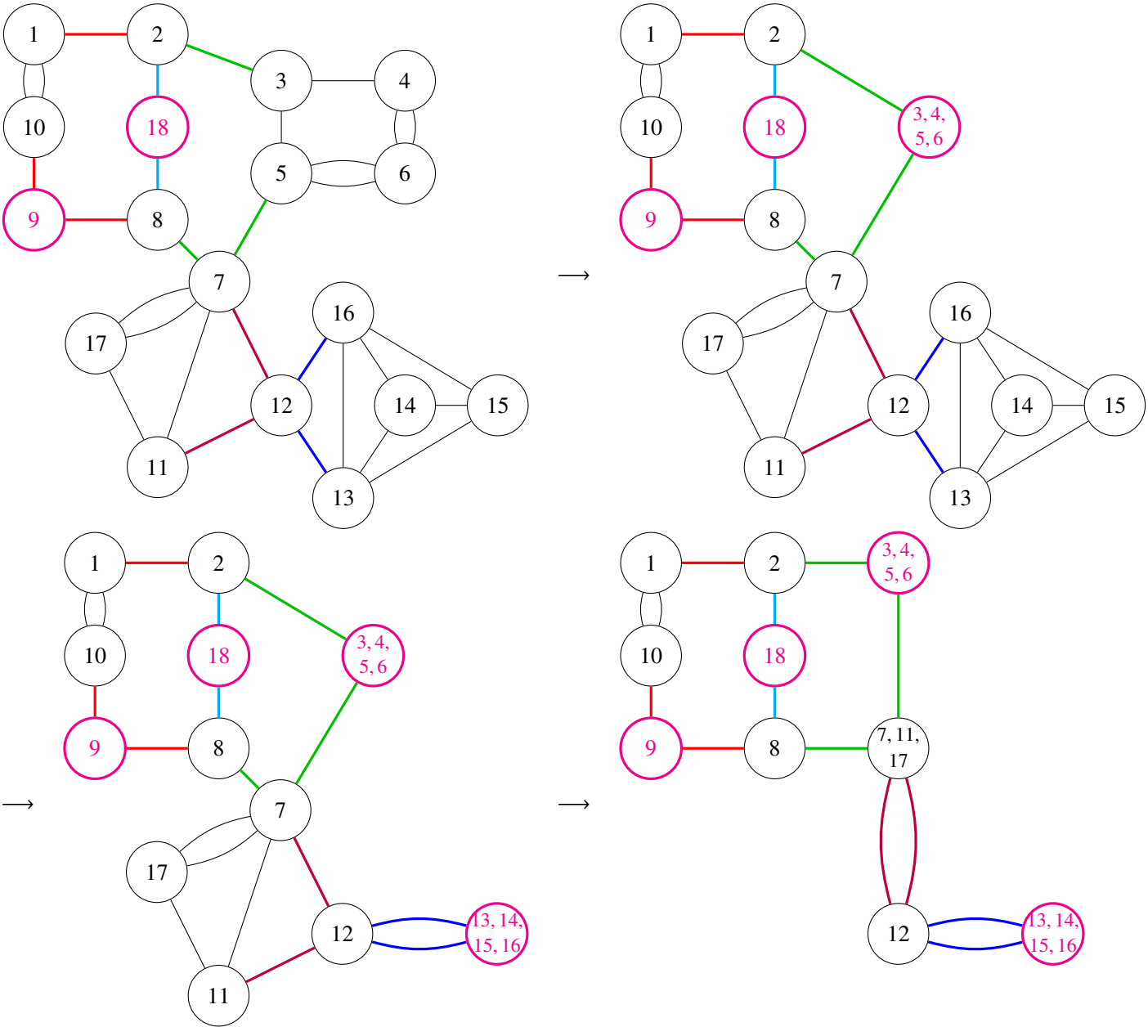

图 15

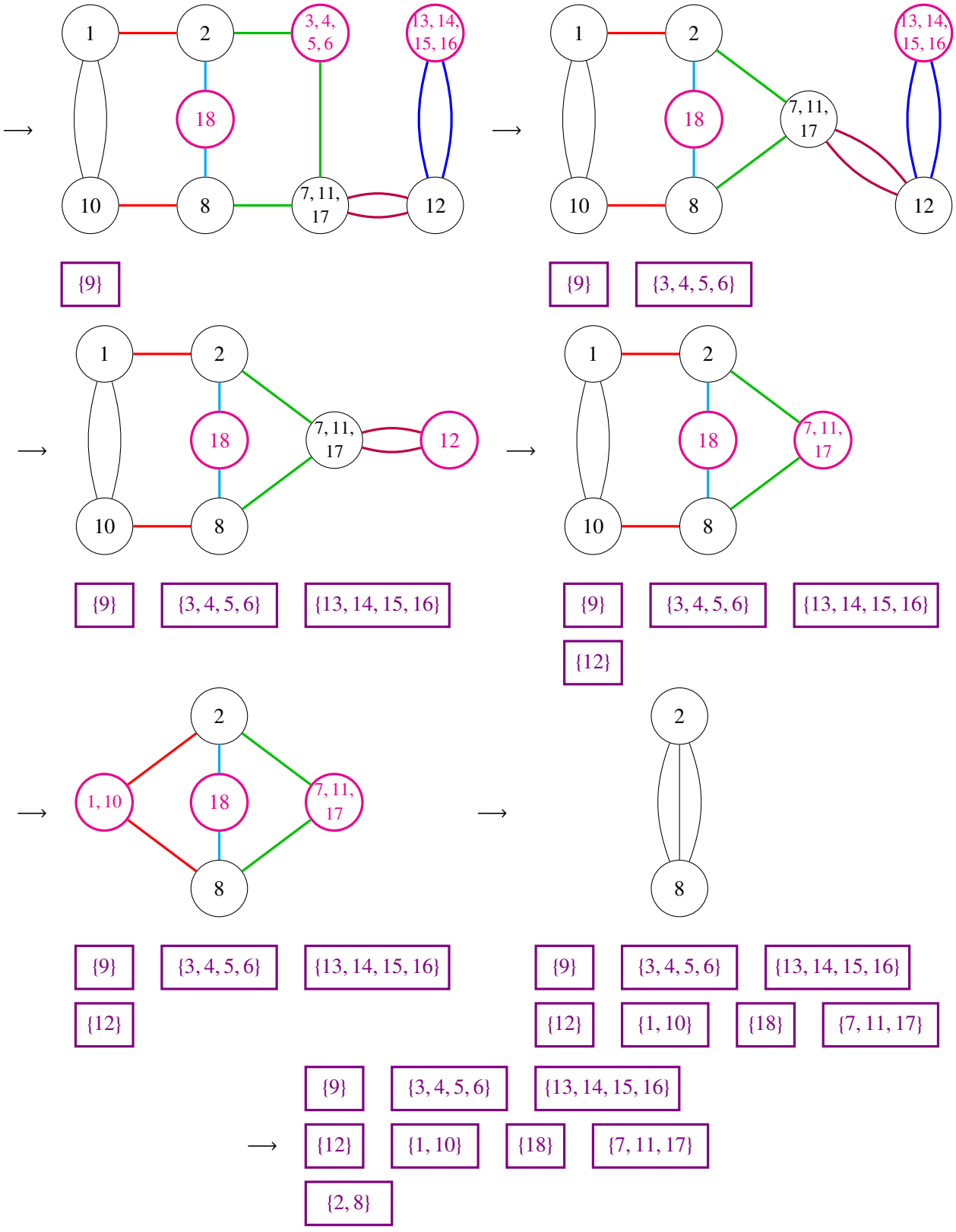
1
2
3, 4, 5, 6
13, 14, 15, 16
18
10
8
7, 11, 17
12
{9}
{3, 4, 5, 6}
{13, 14, 15, 16}
{12}
1, 10
{1, 10}
{18}
{7, 11, 17}
{2, 8}

图 15 (续)

4.1.7 时间复杂度

分析上述算法的时间复杂度。

根据算法的流程可知，一旦应用定理 4.1.3 或定理 4.1.4，图的点数就会减小 1，因此步骤 4 的运行总次数为 $O(|V|)$。

考虑步骤 8 中集合的合并，我们可以使用**链表**来维护每个 v 的 L_v，因为我们只需要枚举 L_v 的所有顶点和合并两个集合。于是步骤 8 的总时间复杂度为 $O(|V|)$。

考虑步骤 11 中枚举顶点，注意到当我们枚举 L_x 的顶点后，点 x 就会被孤立出来，从而 L_x 中的点自成一个 3–边连通分量。这说明，G 中一个顶点至多被枚举到一次，故这部分的总时间复杂度为 $O\left(\sum_{v\in V} d(v)\right) = O(|E|)$。

所有步骤中并查集调用的总次数为 $O(|E|)$，故此部分时间复杂度为 $O(|E|\,\alpha(|V|))$。

剩下步骤的时间复杂度均为单次操作 $O(1)$，故总时间复杂度 $O(|V|)$。

综上，整个算法的时间复杂度为 $O(|E|\,\alpha(|V|))$。

事实上，3–边连通算法是有线性做法的 [12]。不过限于篇幅等原因，这里就不再介绍了。上面的做法所需的引理较少，解法也比较自然，在实际测试中运行时间完全不逊于 [12] 中的方法。

4.2 3–点连通[23]

4.2.1 引入

在 2–点连通时，我们引入了描述点双连通分量的树结构——**块割树**和**圆方树**。在研究 3–点连通时，我们也有与之对应的树结构——**SPQR 树**。

在引入 SPQR 树的定义之前，仍然可以利用定理 2.5.3，可得任意两个 3–点连通分量至多交于两个点。这两个点可以有边相连，也可以无边相连。

4.2.2 边的等价性

定义 4.2.1 (割点对)**.** *对于 2–点连通无向简单图 (下略) G，若某个二元点集 $\{u,v\}$ 是 G 的点割集，则称 (u,v) 是一对*__割点对__ *(split pair)。*

在处理 2–点连通分量时，我们利用了边的等价性，而在 3–点连通分量时，边仍然具有类似的等价性。

我们取两个顶点 u,v，考虑顶点 u,v 定义的局部关系 $\rho_{u,v}$。

[23] 本小节内容默认假设图是 2–点连通无向简单图。

对于 $e,f\in E$，称 $(e,f)\in\rho_{u,v}$，当且仅当存在一条经过 e,f 的路径，除了路径端点外不能是 u 或 v。

定理 4.2.1. *$\rho_{u,v}$ 是等价关系。*

证明. *若 $(e_1,e_2)\in\rho_{u,v},(e_2,e_3)\in\rho_{u,v}$，则我们将对应的两段路径接起来，就得到了一个经过 e_1,e_3 的，端点非 u,v 的路径了。* □

考虑 $\rho_{u,v}$ 将 E 划分的等价类 $E_1,E_2,\cdots,E_n$，我们称如下两种情况为**平凡情形**：

1. $n=1$。
2. $n=2$，且其中一个等价类只包含边 (u,v)。

定理 4.2.2. *设 $|G|\geq 4$。对于 $u,v\in V$，$\rho_{u,v}$ 是平凡等价关系当且仅当 (u,v) 不是割点对。*

证明. *若 u,v 是平凡等价关系，则对于 $\forall a,b\in V\setminus\{u,v\}$，显然 a,b 有不连接 u,v 的邻边 (否则与平凡矛盾)，于是这两条边必然是 $\rho_{u,v}$ 等价的，即 a,b 在 $G\setminus\{u,v\}$ 中连通。*

若 u,v 不是平凡等价关系，于是存在两条不等价的边，由定义知这两条边在 $G\setminus\{u,v\}$ 中不连通，因此 u,v 是割点对。 □

对于所有非平凡的等价关系，我们通过取交得到一个全局关系 ρ_t：对 $e_1,e_2\in E$，称 $(e_1,e_2)\in\rho_t$，当且仅当对于所有**非平凡的** $\rho_{u,v}$，均有 $(e_1,e_2)\in\rho_{u,v}$。

图 16 是一个 ρ_t 对应的等价类划分的例子。

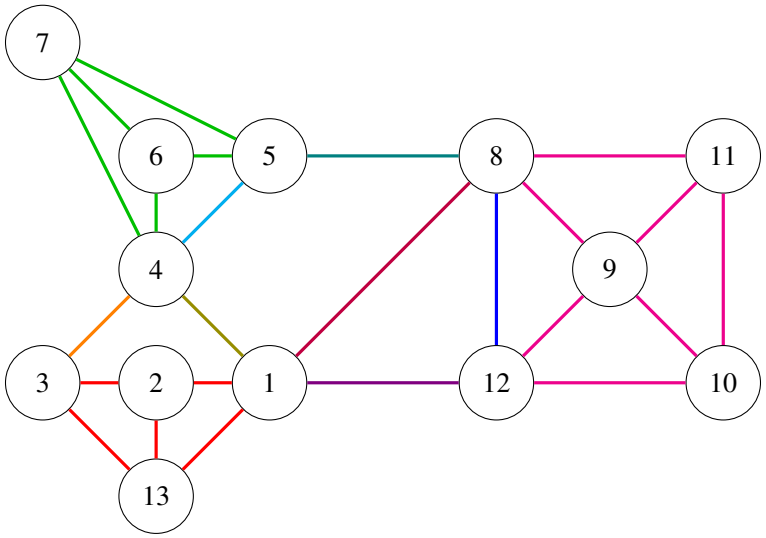

图 16

4.2.3 用割点对划分图

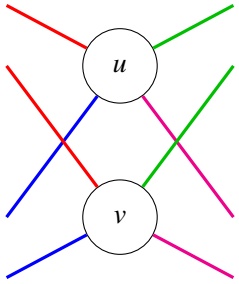

图 17

设 (u,v) 是一个割点对。设关系 $\rho_{u,v}$ 将边集划分为等价类 $E_1,E_2,\cdots,E_n$，设 A,B 为若干个等价类的并，满足 $A\cap B=\varnothing, A\cup B=E, \min\{|A|,|B|\}\geq 2$。

定义两张新图 G_A,G_B，其中 G_A 是包含 A 中所有边，再额外加上边 (u,v) 得到的图，G_B 同理。我们加上的新边被称为**虚边** (virtual edge)，是用来表示划分过程的，虚边是可以有重边的，无论原图中是否存在对应的边。

不断执行这些上述划分操作，直到无法操作为止，这样我们就得到了一张图的**原始点三连通分量**。

注意到圈图 C_n 有很多划分的方法，不过最终本质是一个 n 边形的三角剖分，这是我们不希望看到的，因此我们需要合并这些圈图。

具体地，对于划分完毕后的两张小图 A,B，如果它们**具有共同的虚边**，那么我们定义 A,B **合并**的结果是将它们的点集合并，边集合并并去掉共同虚边。可以看出，**合并是划分的逆过程**。

我们将所有可合并的圈图合并成若干个大圈，将所有可合并的偶极图[24]合成一个大偶极图，最终得到的结果就称为 G 的**点三连通分量**。

4.2.4 SPQR 树

所有的点三连通分量可以分为四类：

- Q (Trivial)：2 个点一对重边的图，作为某些边界情况讨论。由于我们假设 G 为简单图，因此这里我们忽略 Q 情形。
- S (Serial)：至少 3 个点的圈图。圈图中的每一条边可以是原图中的边，也可以是**虚边**。
- P (Parallel)：至少 3 条边的偶极图。由于简单图的假设，因此其中至多一条是原图的边，其余的边均为**虚边**。

[24]dipole graph, 见 [13]。

- R (Rigid)：3–点连通子图，其中每一条边可以是原图中的边，也可以是**虚边**。

定义 4.2.2 (SPQR 树)**.** *SPQR 树 $T = (\mathcal{V}, \mathcal{E})$，其中 $\mathcal{V}$ 表示所有点三连通分量的集合，由之前的结论知可以分为四类。*

对于 $u, v \in \mathcal{T}$，$(u, v) \in \mathcal{E}$ 当且仅当 u, v 具有共同的虚边，即它们是可合并的。

定理 4.2.3. *对于 2–点连通无向图 G，它的 SPQR 树是一棵树。* □

定理 4.2.4. *对于 2–点连通无向图 G，它的 SPQR 树是唯一确定的。* □

4.2.5 构造

根据定义，我们可以得到一个 $O\left(|V||E|^2\right)$ 的做法——不停寻找原图的割点对，将原图划分成更小的三连通分量。

事实上，SPQR 树可以在线性时间内构造，不过限于篇幅这里就不展开了，有兴趣的读者可以见参考文献 [14] [15]。

4.2.6 应用

SPQR 树有较为广泛的应用，下面举两个例子：

例. *给定 2–点连通无向简单图 G，找出 G 的所有割点对。*

考虑 G 的 SPQR 树，注意到 (u, v) 是割点对当且仅当 $\rho_{u,v}$ 不平凡。而不平凡的 $\rho_{u,v}$ 在最终 SPQR 树的表现只有两种：

- 作为 SPQR 树中的一条**虚边**。
- 作为圈图的**虚边**被合并。

因此，我们可以得到：(u, v) 是割点对，当且仅当 u, v 是 SPQR 树上的虚边，或 u, v 是 SPQR 树中 S 型顶点 (圈图) 中的不相邻顶点 (因为它们被合并了)。

例. *3–点连通平面图，又称**多面体图**[25]，是表示凸多面体结构的图，可以理解为凸多面体的球极投影。*

定理 4.2.5 (Steinitz)**.** *G 为表示凸多面体结构的图，当且仅当 G 是 3–点连通平面图。*

定理 4.2.6. *若 G 是多面体图，则当选定无穷平面后，在拓扑意义下 G 具有唯一的平面嵌入。*

这些定理，和本文的关系不大，可以见参考文献 [17] [18]。

[25] polyhedral graph, 见 [16]

5 总结

本文介绍了图的连通性理论，从连通性的定义出发，介绍了 k–连通问题的一般解法，以及当 k 较小时的特殊做法和应用。本文挑选的做法都是在 OI 中较有实际应用的，希望能在信息学竞赛中有所普及。

同时，本文对大家熟悉 2–点/边连通问题作了更本质的讲解，并做了一个自然的推广——3–点/连通问题，且使用适量的插图来辅助理解。希望大家在阅读本文后，能对 2–连通问题有更深入的了解，对 3–连通问题有一定的认知。

不过，限于笔者水平有限，对于 k–点连通分量的问题，目前并没有较为高效实用的做法。希望本文能起到一个抛砖引玉的作用，吸引更多读者来研究图论以及组合数学类的问题。

感谢

感谢中国计算机学会提供学习和交流的平台。
感谢国家集训队高闻远教练的指导。
感谢父母对我的照顾与支持。
感谢符水波老师、应平安老师对我的关心与教导。
感谢潘佳奇、钱易等同学与我交流，给我启发。
感谢孙睿泽、宣毅鸣、翁伟捷同学为本文审稿。

参考文献

[1] Wikipedia contributors. Max-flow min-cut theorem. *Wikipedia*, https://en.wikipedia.org/wiki/Max-flow_min-cut_theorem.

[2] Wikipedia contributors. Maximum flow problem. *Wikipedia*, https://en.wikipedia.org/wiki/Maximum_flow_problem#Algorithms.

[3] Wikipedia contributors. Karger's algorithm. *Wikipedia*, https://en.wikipedia.org/wiki/Karger's_algorithm.

[4] Wikipedia contributors. Stoer–Wagner algorithm. *Wikipedia*, https://en.wikipedia.org/wiki/Stoer%E2%80%93Wagner_algorithm.

[5] Shimon Even (1979), *Graph Algorithms*, Computer Science Press.

[6] Dan Gusfield (1990), *Very Simple Methods for All Pairs Network Flow Analysis*, SIAM J. Computing.

[7] Wikipedia contributors. Biconnected component. *Wikipedia*, https://en.wikipedia.org/wiki/Biconnected_component#Block-cut_tree.

[8] Robert Tarjan (1972), *Depth-first search and linear graph algorithms*, SIAM J. Computing.

[9] Wikipedia contributors. Logical equivalence. *Wikipedia*, https://en.wikipedia.org/wiki/Logical_equivalence.

[10] Wikipedia contributors. Edge contraction. *Wikipedia*, https://en.wikipedia.org/wiki/Edge_contraction.

[11] Wikipedia contributors. Cycle basis. *Wikipedia*, https://en.wikipedia.org/wiki/Cycle_basis.

[12] Yung H. Tsin (2007), *A Simple 3-Edge-Connected Component Algorithm*. Theory of Computing Systems.

[13] Wikipedia contributors. Dipole graph. *Wikipedia*, https://en.wikipedia.org/wiki/Dipole_graph.

[14] Carsten Gutwenger, Petra Mutzel (2001), *A linear time implementation of SPQR-trees*, Lecture Notes in Computer Science.

[15] John Hopcroft, Robert Tarjan (1973), *Dividing a graph into triconnected components*, SIAM J. Computing.

[16] Wikipedia contributors. Polyhedral graph. *Wikipedia*, https://en.wikipedia.org/wiki/Polyhedral_graph.

[17] Wikipedia contributors. Steinitz's theorem. *Wikipedia*, https://en.wikipedia.org/wiki/Steinitz's_theorem.

[18] Saunders Mac Lane (1937), *A structural characterization of planar combinatorial graphs*, Duke Mathematical Journal.