arXiv:2301.03390v1 [cs.DS] 9 Jan 2023

空间-查询范围子图计数与列举中的权衡

邓世远、卢尚琪、陶宇飞

香港中文大学计算机科学与工程系 香港，中国{
sydeng,sqlu,taoyf}@cse.cuhk.edu.hk

2023年1月10日

摘要

本文开创了范围子图计数和范围子图枚举的研究，这两类问题源于实践中的重要需求：人们需要对仅与选定顶点相关、而非与所有顶点相关的子图进行图分析。第一个问题中，给定一个无向图 G，其中每个顶点都带有一个实值属性。给定一个区间 q 和一个模式 Q，查询需要统计由属性值落在 q 内的顶点所诱导的 G 子图中， Q 出现的次数。第二个问题的设置相同，但查询需要以较小的延迟枚举（而非计数）这些出现。在这两个问题中，我们的目标是理解空间使用量与查询代价之间的权衡，具体而言：（i）给定查询效率目标，我们必须存储多少关于 G 的预计算信息？（ii）反过来，给定空间使用量预算，我们能期望达到的最佳查询时间是多少？针对各种查询模式，我们给出了关于此类权衡的一系列上界和下界结果。

本研究部分经费由香港研究资助局的GRF项目14207820、14203421和14222822资助。

1 引言

设 $G = (V, E)$ 为数据图， Q 为模式图。如果 G 的一个子图与 Q 同构，则称其为 Q 的一次出现。模式搜索的目标是枚举 Q 的所有出现，或统计其出现次数。这两者都是计算机科学中的基础问题，并在过去几十年中引起了广泛关注。

本文研究顶点诱导子图中的模式搜索。在这里，查询会选取顶点的一个子集$U \subseteq V$，并需要对 G'中的 Q 进行计数/枚举，其中 G' 是由 U诱导的 G的子图。请注意，如果某个出现包含 U之外的顶点，则不应对该出现进行计数/枚举。显然，可以先生成 G' ，再"从头开始"对 G' 中的Q 进行计数/枚举来回答查询，但这并未发挥预处理的优势。相反，我们的目标是将 G 存储在一种能够以非平凡保证回答所有查询的数据结构中。一个值得探究的问题是：在给定空间预算的条件下，我们能将查询时间降到多低；反过来，要达到目标查询时间，我们必须消耗多少空间。

在数据库系统中，顶点选择通过谓词 q完成，该谓词将 U 确定为满足 q}的 $\{v \in V \mid v$ 。我们重点关注范围谓词，所考虑的问题如下：

问题 1（范围子图计数）。$G = (V, E)$ 是一个无向图，其中每个顶点 $v \in V$ 都带有一个实值属性 A_v。对于一个区间 $q = [x_1, x_2]$，定义$V_q = \{v \in V \mid x_1 \leq A_v \leq x_2\}$和 G_q 为由 V_q在 G 中诱导的子图。令 Q 为一个连通（仅有一个连通分量）的模式图，含有 $O(1)$个顶点。给定一个区间 q，查询返回 G_q中 Q 的出现次数。所有查询的模式 Q 均固定不变。

问题 2（范围子图枚举）。设置与前文相同，但查询返回 G_q中 Q 的出现情况。

通用记号。本文中将始终使用若干记号。令 $n = |V|$ 和$m = |E|$。符号 $\omega < 2.37286$ [1] 表示矩阵乘法指数。记号~ ~$O(.)$ 和 $\Omega(.)$ 隐去一个相对于底层问题参数呈多对数级的因子。

1.1 动机

实际应用。子图模式对于理解数据图 G 的特征十分重要，这一点已在大量论文中得到记载，例如 [2, 3, 8, 10, 11, 17, 18,24–28, 30, 33, 36–38, 50]。在实践中，分析人员不仅关注整个 G 的模式，也关注仅与选定顶点相关的模式。设想一个社交网络 G ，其中每个顶点代表一个人。图的聚类系数 [49]，是网络科学中一种常见的度量，等于三角形（ 3- 团 [1] ）的数量与楔形（ 2- 路径 [2] ）的数量之比。然而， G 的系数只是一个单一数值，无法充分揭示特定人口群体的特征。例如，比较以下系数会更有参考价值：(i) 由年龄为 $\in [20, 30]$ 的人构成的 G 子图，以及 (ii) 由年龄为 $\in [60, 70]$ 的人构成的子图。更进一步，将针对每个$i \in [1, 10]$的"年龄 $\in [i \cdot 10, (i + 1) \cdot 10]$"所诱导的系数汇总起来，就能比较不同年龄组。随后，可以针对目标群体的模式出现次数进行更精细的分析。上述分析的强大之处源于

[1]一个 ℓ-*clique* 是含有 ℓ个顶点的团。[2]一个 ℓ-*path* 是含有 ℓ 条边的路径。

问题	模式 Q	空间	查询	备注
1（计数）	任意固定的 Q	$O(n^2)$	$\tilde{O}(1)$	近似最优†
1	楔形	$\tilde{O}(m^2/\lambda^2)$	$\tilde{O}(\lambda)$	对于任意 $\lambda \in [1, \sqrt{m}]$，近似最优†
1（下界）	楔形	$\tilde{O}(m^{2-\delta}/\lambda^2)$ 不可能	$\tilde{O}(\lambda)$	对于 $\lambda \in [1, \sqrt{m}]$ 和任意 $\delta > 0$，受强集合不相交性猜想限制。
1	ℓ-团	$O(m)$	$\tilde{O}(1)$	
2（重复）	任意固定的 Q	$O(m + m^{\rho^*}/\Delta)$	延迟 $\tilde{O}(\Delta)$	对于任意 $\Delta \geq 1$，$\rho^* =$ 分数。 Q的边覆盖数
2	三角形	$O(m)$	延迟 $\tilde{O}(1 + (m^*)^{\frac{\omega-1}{\omega+1}})$	$m^* =$ 中的边数 G_q中至少一个三角形
2	ℓ-星	$O(m)$	延迟 $\tilde{O}(1)$	近似最优
2	2ℓ-环	$\tilde{O}(\#P_\ell)$	延迟 $\tilde{O}(1)$	$\#P_\ell =$ G 中 ℓ-路径的数量

备注：“近似最优”意味着无法实现多项式级改进（即 n^δ 对于任意小的常数 $\delta > 0$）。标记为 † 的近似最优性以强集合不相交猜想为前提。

表 1：研究结果汇总

适用于问题 1 和 2 的任意选择范围查询。设计有效的数据结构对于避免冗长的响应时间至关重要。

空间-查询权衡的重要性。不要将空间-查询权衡与预处理时间和查询代价之间的权衡混淆，后者已在连接算法中得到广泛研究 [5, 12, 20–23, 35, 41–43, 45]。这两种权衡都很重要，但作用方式不同。与预处理时间不同，后者是“一次性成本”（因为数据结构一旦构建，便可永久使用），空间消耗则是永久性的。换句话说，空间-查询权衡对底层数据库系统的影响（要）持久得多。然而，尽管空间-查询权衡十分重要，连接方面的相关研究却少得惊人：我们只知道一篇论文 [20]，，正如第 1.3 节将讨论的那样，该论文未考虑查询谓词（或者等价地说，只存在一个始终输出整个连接的查询），且只关注报告（而不关注计数）。我们的工作可视为沿着 [20] 的同一方向迈出的一步，因为如第 5 节所述，子图搜索可以表述为连接问题（事实上，我们的一些结果明确针对连接），而且这实际上是谓词驱动查询和计数方面的第一步。

最后值得一提的是，无论一个有用的数据结构占用多么少的空间，都必须能够在多项式时间内构建。我们论文中构建的所有数据结构都满足这一点。事实上，忽略多对数因子，构建我们的每个数据结构所需的时间至多等于找出查询模式 Q 的所有出现次数所需的时间。

1.2 我们的贡献

表 1 总结 了本文的主要结果。接下来，我们将详细解释这些结果 1.

1.2.1 问题 1

楔形。我们将证明：

定理 1.1. 考虑带有 $Q =$ 楔形的问题 1。对于任意实数值 $\lambda \in [1, \sqrt{m}]$，存在一个占用 $\tilde{O}(m^2/\lambda^2)$ 空间的数据结构，可以在 $\tilde{O}(\lambda)$ 时间内回答查询。

这种空间-查询权衡可能令人失望。毕竟，楔形计数的一次性计算很简单：我们可以在 G 使用 $O(n+m)$ 时间统计楔形的数量。自然会让人想知道，定理 1.1 中的空间需求是否不可避免。我们通过证明，对定理 1.1 的任何实质性改进都将为集合不相交性带来重大突破，来回答这个问题：

集合不相交性。数据是由 $s \geq 2$ 个集合 S_1、S_2、...、S_s 组成的集合。给定不同的集合 ID a、$b \in [1, s]$，查询会返回 $S_a \cap S_b$ 是否为空。

令 $N = \sum_{i=1}^{s} |S_i|$ 为集合不相交性的输入规模。对于任意 $\lambda \in [1, \sqrt{N}]$，都有一个简单的结构，占用 $O(N^2/\lambda^2)$ 空间，查询时间为 $O(\lambda)$（见附录 B）。即使只针对一个任意的 λ，要将这种权衡改进一个多项式因子，也一直是一个悬而未决的问题。强集合不相交猜想 [31, 32] 指出，对于任意 $\lambda \geq 1$，查询时间为 λ 的结构必须使用 $\tilde{\Omega}(N^2/\lambda^2)$ 空间。我们将证明：

定理 1.2。考虑带有 $Q =$ 楔形的问题 1。固定任意 $\lambda \in [1, \sqrt{m}]$ 和任意常数 $\delta > 0$。假设我们能够获得一个占用 $\tilde{O}(m^{2-\delta}/\lambda^2)$ 空间、并能在 $\tilde{O}(\lambda)$ 时间内回答查询的结构。那么，对于任意输入规模为 N 的集合不相交性实例，我们都能获得一个占用 $\tilde{O}(N^{2-\delta}/\lambda^2)$ 空间、并能在 $O(\lambda)$ 时间内回答查询的结构（从而推翻强集合不相交猜想）。

团。我们将证明：

定理 1.3。对于带有 $Q = \ell$ 团的问题 1，存在一种占用 $O(m)$ 空间的数据结构，可在 $O(1)$ 时间内回答一次查询。

三角形计数（$\ell = 3$）似乎比楔形计数更难：在一次性计算中，目前已知最快的三角形计数算法需要 $O(m^{\frac{2\omega}{\omega+1}})$ 时间。因此，在问题 1 中看到 $Q =$ 三角形比 $Q =$ 楔形更容易，令人惊讶。根据定理 1.1 和 1.3 可知，计算任意 q 的聚类系数（见第 1.1 节）的问题，归结为对 G_q 中的楔形进行计数。实际上，这意味着该问题已得到最优解决（以强集合不相交猜想成立为前提）；由于聚类系数广受关注，这一结果具有实际意义。

任意子图。我们将证明：

定理 1.4。对于任意 Q，存在一种用于问题 1 的数据结构，占用 $O(n^2)$ 空间，并可在 $O(1)$ 时间内回答一次查询。

上述结果很难改进：将空间减少任意常数 n^δ 因子$\delta > 0$ 都会推翻强集合不相交猜想。具体而言，假设 $n = O(m)$。[3] 如果存在一种占用 $O(n^{2-\delta}) = O(m^{2-\delta})$ 空间、查询时间为 $\tilde{O}(1)$ 的数据结构，那么将该数据结构应用于 $Q =$ 楔形，就能通过定理 1.2 在集合不相交问题上取得突破。读者应注意，难点在于对所有 Q都给出保证；对于特殊模式，则有可能做得更好（定理 1.3）。因此，这一难点使 $Q =$ 楔形在问题 1 中具有独特的重要性。定理 1.4 进一步表明，在 $Q =$ 楔形下，当 G 最稀疏时，问题 1 最难：任意常数 $\epsilon > 0$ 下均有 $m = o(n^{1+\epsilon})$。为了解释原因，令 $m = n^{1+\epsilon}$，可得 $n^2 = m^{\frac{2}{1+\epsilon}}$。由于 $\frac{2}{1+\epsilon} = 2 - \frac{2\epsilon}{1+\epsilon}$，定理 1.4 给出了一种占用 $O(m^{2-\delta})$ 空间、查询时间为 $\tilde{O}(1)$ 的数据结构，且$\delta = \frac{2\epsilon}{1+\epsilon}$，从而在 $\lambda = \tilde{O}(1)$ 处以多项式因子改进了定理 1.1。

[3] 丢弃没有关联边的“孤立”顶点 es.

1.2.2 问题 2

枚举查询保证延迟 Δ：若有新的 Q 出现，则在上一次出现[4]后的 Δ 时间内报告该出现；否则声明“没有更多出现”。

任意子图。我们将展示：

定理 1.5。对于任意 Q 和 $\Delta \geq 1$，存在一种用于问题 2 的数据结构，占用 $\tilde{O}(m + m^{\rho^*}/\Delta)$˜空间，并且查询延迟为 $O(\Delta)$，其中 ρ^* 是 Q的分数边覆盖数。

设想为 Q的每条边赋予一个非负权重，使得：(i) 对于 Q的每个顶点，其所有关联边的权重之和至少为 1；(ii) 所有边的总权重最小。 Q 的分数边覆盖数 ρ^* 就是最优赋值的总权重。 G 中 Q 的最大出现次数为 $O(m^{\rho^*})$ [4]，并且在最坏情况下该界是紧的。

我们的数据结构实际上解决了一个关于自然连接的问题：

> 范围连接。设 $\mathcal{R}$ 为一个由 $O(1)$个关系组成的集合，其中每个关系都有 $O(1)$个实值属性。用 join($\mathcal{R}$) 表示这些关系 $\mathcal{R}$上的自然连接结果。给定一个区间 $q = [x_1, x_2]$，查询将返回 $t \in$ join($\mathcal{R}$) 中满足 t 的每个属性都落在 q内的所有元组。

令 N 为 $\mathcal{R}$中关系的元组总数。对于任意 $\Delta \geq 1$，我们给出一个占用$\tilde{O}(N + N^{\rho^*}/\Delta)$ 空间的数据结构，其回答查询的延迟为 $\tilde{O}(\Delta)$。这里，分数边覆盖数 ρ^* 是相对于连接的超图而言的（详见第 5 节）。

定理 1.5 的难点在于设计一种适用于所有 Q的数据结构。针对特定的 Q，则有可能做得更好。接下来，我们介绍三个例子；它们不仅本身是重要的子问题，也展示了不同的技术。

三角形。我们将证明：

定理 1.6。对于包含 $Q =$ 三角形的问题 2，存在一种占用 $O(m)$ 空间的数据结构，其回答查询的延迟为 $\tilde{O}(1 + (m^*)^{\frac{\omega-1}{\omega+1}})$，其中 m^* 是至少出现在一个已报告三角形中的边数。

˜三角形的分数边覆盖数 ρ^* 为 1.5。为确保 $O(m)$ 空间，定理 1.5 需要设置 $\Delta = \sqrt{m}$。如 $\frac{\omega-1}{\omega+1} < 0.408$所示，定理 1.6 在延迟方面实现了多项式级改进。读者应注意，定理 1.6 中的值 m^* 从不超过 m，但可能小得多（当需要列出的三角形很少时就会如此）。问题 2 中，三角形的 $Q =$ 且 q固定为 $(-\infty, \infty)$，曾被用作 [20], 先前工作的动机问题；该工作描述了一种占用 $O(m)$ 空间、延迟为 $O(\sqrt{m})$ 的数据结构，因此定理 1.6 对其实现了严格改进。

ℓ-星。 ℓ-星是一棵有 ℓ 个叶节点和一个非叶节点的树（楔形是 2-星）。我们将证明：

定理 1.7。对于 $Q = \ell$-星问题 2，存在一种占用 $O(m)$ 空间的数据结构，能够以 $O(1)$ 的延迟回答一次˜查询。

作为推论，对于任意区间 q，只需 $O(m)$ 空间，即可使用 $O(1)$ 时间检测是否存在 ℓ-星，其中使用 $\lnot G_q$。对于 $Q =$ 楔形，这意味着定理 1.2 所体现的困难确实源于计数。

2ℓ-环[5]。我们将证明：

[4]读者可以假设，查询算法总是在开始时输出一个虚拟出现。[5]一个具有 2ℓ 个顶点的环。

˜定理 1.8. 对于满足 $\ell \geq 2$ 的 $Q = 2\ell$-环问题 2，存在一种占用 $O(\#P_\ell)$ 空间˜的数据结构，能够以 $O(1)$ 的延迟回答查询，其中 $\#P_\ell$ 是 G 中 ℓ-路径的数量。

分数边覆盖数 ρ^* 对于 2ℓ-环为 ℓ 。定理 1.5 需要 $\tilde{O}(m^\ell)$ 空间˜，才能达到 $O(1)$ 的延迟。定理 1.8 中的空间显著更优。对于 $\ell = 2(Q = 4$-˜环)，空间为 $O(nm)$，即 G 中楔形的最大数量。对于 $\ell > 2$，空间为$\tilde{O}(m^{\lceil(\ell+1)/2\rceil})$，即 G 中 ℓ-路径的最大数量。

1.3 相关工作

前几节介绍了最相关的现有结果。现在我们将讨论其他相关工作。

模式搜索在一次性计算中已得到广泛研究。关于计数和枚举算法，我们分别请读者参阅 [3,8,10,17,18,27,28,30,37,50]和 [2,11,17,24–26,33,36,38,39],及其中的参考文献。生成 G_q 后，这些算法可应用于问题 1 和 2。本文关注的是避免完整生成 G_q，因为这可能需要 $\Omega(m)$ 的时间。

在另一个极端情况下，我们可以预先计算 G 中 Q 的出现集合 S 。假设 Q 的大小为常数， S 的大小为 $O(m^{\rho^*})$ （ AGM 界）。借助标准的计算几何技术 [19], ，我们可以将 S 存储在空间为 $\tilde{O}($ $m^{\rho^*})$ 的数据结构中，从而在 $O(1)$ 时间内回答问题 1 的查询 ˜˜，并以 $O(1)$ 的延迟回答问题 2 的查询。对于问题 1，定理 1.4 对每个满足 $\rho^* \geq 2$ 的 Q 都能达到更优的空间界。当 $\rho^* < 2$ 时， Q 至多有三个顶点：1-路径（单边）、楔形或三角形。我们已经解决了楔形和三角形的情形（定理 1.1 和 1.3），而对于 $Q = 1$-路径，问题 1 是平凡的。对于问题 2，定理 1.5 将上述极端思路作为 $\Delta = O(1)$ 的特殊情形加以体现，并提供可调节的空间—查询权衡。

Bannister 等人提出的关系事件图 [6], 是一种图 $G = (V, E)$，其中每条边 $e \in E$ 都带有一个实数值时间戳。对于区间 $q = [x_1, x_2]$，令 G_q^{edge}表示由 G中时间戳落在 q覆盖范围内的所有边所诱导的子图。模式搜索查询用于统计/枚举模式 Q 在 G_q^{edge}中的出现次数。参见 [6, 14, 15] ，其中介绍了若干专为此类查询设计的数据结构。虽然听起来相似，但在关系事件图上进行模式搜索与问题 1 和问题 2 截然不同，因此我们的解决方案与 [6, 14, 15]中的方案，无论结果还是技术，都几乎没有重叠。

延迟最小化是连接和合取查询领域文献中的一个重要主题；参见 [5,9,12,13,20–23,34,35,41,43–45] 及其参考文献。就我们的问题而言，我们不知道此前有任何工作给出了优于前文所述结果的结论。我们对范围连接枚举的表述（第 1.2.2 节）表明，从索引的角度看，查询谓词的存在会给连接（以及合取查询）带来新的挑战。Deep 和 Koutris [20]˜在问题 2 上证明了一个与定理 1.5 等价的结果（相差一个 $O(1)$ 因子），但仅限于查询涉及整个 G这一特殊情形，即将查询范围 q 固定为 $(-\infty, \infty)$。

2 预备知识

本节将介绍我们将在解决方案中使用的若干技术工具。

多维点的数据结构。下面将介绍一些广为人知的几何数据结构。读者无需费心了解这些数据结构的

细节，因为我们会将它们作为“黑盒”使用。设 P 是 d 维空间 $\mathbb{R}^d$ 中的一个 n 点集合，其中 d 为常数。给定一个形式为 $[x_1, y_1] \times [x_2, y_2] \times ... \times [x_d, y_d]$ 的矩形 q，范围报告查询会枚举 $P \cap q$ 中的点。我们可以在 P 上构建一棵范围树 [7, 19]，它使用 $O(n)$ 空间，并允许我们以 $O(1)$ 的延迟回答此类查询。当 $d = 2$ 时，我们可以用 Chazelle 结构 [16] 替换范围树；该结构保留了上述查询性能，同时将空间消耗降至 $O(n)$。

我们还需要考虑这样一种场景下对 P 进行范围求和查询：P 中的每个点都是二维的（即 $d = 2$），并且带有实数权重。给定一个矩形 $q = [x_1, y_1] \times [x_2, y_2]$，这类查询会返回 $P \cap q$ 中各点权重的总和。我们同样可以在 P 上构建一个 Chazelle 结构 [16]~，其占用 $O(n)$ 空间，并能在 $O(1)$ 时间内回答查询。

从“含重复项的延迟”到“互异条件下的延迟”。我们来考虑一种设计低延迟算法时经常遇到的重复项移除场景。假设我们有一个算法 $\mathcal{A}$，用于枚举由 S 个元素组成的集合。延迟为 Δ 时，$\mathcal{A}$ 可以报告一个元素 $e \in S$，但无法保证 e 之前从未被报告过。另一方面，好消息是，对于某个 $\alpha \geq 1$，$\mathcal{A}$ 输出同一个元素的次数至多为 α。

通过修改 [47], 中的缓冲技术，我们可以将 $\mathcal{A}$ 转换为一种算法，该算法仅枚举 S 中的互异元素，延迟为 $O(\alpha \cdot \Delta \log |S|)$。从概念上讲，将 $\mathcal{A}$ 的执行划分为多个轮次，每个轮次运行 $\alpha \cdot \Delta$ 时间[6]。随着 $\mathcal{A}$ 的运行，我们使用缓冲区 B 暂存已由 $\mathcal{A}$ 找到但尚未报告的互异元素集合。每当 $\mathcal{A}$ 找到一个元素 $e \in S$时，我们检查 e 是否曾经存在于 B 中（利用维护在迄今找到的所有元素上的二叉搜索树，这需要 $O(\log |S|)$ 时间）。如果存在，则忽略 e；否则，将其加入 B。在每个轮次结束时，我们从 B 中输出任意一个元素，并将其从 B中移除。最后，在 $\mathcal{A}$ 终止后，我们只需输出 B中剩余的元素。

B 在每个轮次结束时始终至少包含一个元素。为说明原因，考虑某个 $t \geq 1$的第 t个轮次结束时的情况。此时，$\mathcal{A}$ 已经运行了 $t \cdot \alpha \cdot \Delta$ 时间，因此必定已经报告了 $t \cdot \alpha$ 个元素，这些元素可能并不互异。然而，由于每个元素最多只能被报告 α次，因此在这 $t \cdot \alpha$ 个元素中至少有 t 个（互异元素）。由于我们在前面的轮次中只报告了 $t - 1$ 个元素，因此在第 t个轮次结束时，B 中仍至少有一个元素。现在很容易验证，修改后的算法在枚举 S中的互异元素时，其延迟为 $O(\alpha \cdot \Delta \log |S|)$。

3 问题 1：匹配上界与下界

本节将确立定理 1.2 中的条件下界及其匹配上界（定理 1.4）。我们对上界的讨论还将确立定理 1.3。全文中，我们假设 G 的顶点具有各不相同的属性值。该假设不失一般性，因为可以通过顶点 ID 打破平局。

3.1 下界

假设在 $Q =$ 楔形下，问题 1 存在一种数据结构，使用 $\tilde{O}(m^{2-\delta}/\lambda^2)$ 空间，并且~能在 $O(\lambda)$ 时间内回答查询，其中某个 $\lambda \geq 1$ 满足条件。我们将设计一种用于集合不相交性的 data structure，使用 $\tilde{O}(N^{2-\delta}/\lambda^2)$ 空间，并能在 $\tilde{O}(\lambda)$ 时间内回答查询。回顾一下，集合不相交性问题的输入数据由 $s \geq 2$ 个集合 $S_1, ..., S_s$ 组成，总大小为 N。定义 $\mathcal{U} = \bigcup_{i=1}^{s} S_i$。

[6]请注意，RAM 模型中的“时间”定义为所执行的原子操作数量（例如加法、乘法、比较、访问内存字等）。每一轮本质上都是由 $\alpha \cdot \Delta$ 个此类操作组成的序列。

按如下方式创建图 $G=(V,E)$。 V 有 $2s+|\mathcal{U}|$ 个顶点，包括 $2s$ 个集合顶点和 $|\mathcal{U}|$ 个元素顶点。每个集合 S_i ($i\in[1,s]$) 定义两个集合顶点，其属性值分别设为 i 和 $s+i$。 $\mathcal{U}$ 中的每个元素都定义一个元素顶点，其属性值与之相同$s+1/2$。集合 E 包含 $2N$ 条边：对于每个元素 $e\in S_i$，在 E 中添加两条边，每条边连接 e 的元素顶点和 S_i 的一个集合顶点。现在，在 G 上创建一个 $Q=$ 楔形下的问题 1 结构。该数据结构占用 $\tilde{O}(N^{2-\delta}/\lambda^2)$ 空间。

考虑一个集合不相交查询，其集合 ID 为 a 和 b。不失一般性地假设 $a<b$，我们针对 G 上的区间 $q_1=[a,s+b]$、 $q_2=[a+1,s+b]$、$q_3=[a,s+b-1]$ 和 $q_4=[a+1,s+b-1]$，分别发起四个问题 1 楔形计数查询。令 c_1、c_2、...、c_4 为返回的计数。~当且仅当 $c_1-c_2-c_3+c_4>0$时，我们才判定 $S_a\cap S_b$非空。查询时间为 $O(\lambda)$。附录 A 证明了该算法的正确性。至此，定理 1.2 的证明完成。

3.2 上界

接下来，我们将通过允许 Q 为任意模式图来解决问题 1。考虑 G中 Q 的任意一次出现。令 u（分别为 v）表示该次出现中属性最小（分别为最大）的顶点。我们将该次出现登记在数对 (u,v) 处。记 $c_{u,\ v}$ 为登记在 (u,v)处的出现次数。

对于一次带有 $q=[x_1,x_2]$ 的查询，登记在 (u,v) 处的出现，当且仅当 $A_u\geq x_1$ 且 $A_v\leq x_2$ 时，才会出现在 G_q（即由 V_q 诱导的 G 的子图）中。因此，我们可以将问题转化为对二维点进行范围求和。对于每一对 $(u,v)\in V\times V$，创建一个权重为 $c_{u,\ v}$ 的点 (A_u,A_v)。令 P 为所创建点的集合；显然， $|P|=O(n^2)$。查询结果就是被矩形 $[x_1,\infty)\times(-\infty,x_2]$ 覆盖的 P 中所有点的总权重（即一次范围求和操作）。我们可以将 P 存储在 Chazelle 结构（见第 2 节）中，该结构占用 $O(|P|)=O(n^2)$ 空间，并能在 $O(1)$ 时间内执行~次范围求和操作。由此证明了定理 1.4。

团的改进。当 Q 是一个团时，我们的数据结构所需空间可降至 $O(m)$。关键观察是，在(u,v)处登记一次出现意味着 $\{u,v\}\in E$。我们只向 P 添加具有非零$c_{u\text{权重},\ v}$ 的点(A_u,A_v)（权重为零的点不会影响范围求和操作）。这会使 P 的大小至多降至 m ，从而使 Chazelle 结构所需空间降至 $O(m)$。由此，我们完成了定理 1.3 的证明。

4 问题 1：楔形

本节将说明，如何在 $Q=$楔形的情形下，为问题 1 实现定理 1.1 中的保证。我们将 $G=(V,E)$中的一次楔形出现表示为 wedge(u,v,w)，其中 u、 v和 w是 V中的顶点，而 $\{u,v\}$ 和 $\{v,w\}$ 是 E中的边。下面介绍一个略有不同的问题：

> 彩色范围楔形计数。像问题 1 中那样，为每个 $v\in V$定义 $G=(V,E)$ 和 A_v。 V中的每个顶点都被着成黑色或白色。给定一个区间 q，查询返回满足 $A_u\in q$、 $A_w\in q$且 v为黑色的楔形(u,v,w)的出现次数。

请注意，对于 A_v 以及 u 和 w 的颜色，没有任何要求。

令 $\mathcal{C}$ 为 V 的子集构成的集合。若满足以下条件，我们称 $\mathcal{C}$ 为规范集合：

- (P4-1) V 中的每个顶点都出现在 $\mathcal{C}$ 中的 $O(1)$ 个子集中；

• (P4-2) 对于任意区间 q，我们可以将 V_q （即 V 中属性˜值位于 q 的顶点集合）划分为 O(1) 个互不相交的子集，每个子集都是 $\mathcal{C}$ 的成员。这些子集的 ID˜可以在 O(1) 时间内获得。

找到满足 $\sum_{U\in\mathcal{C}}|U| = \tilde{O}(n)$ 的规范集合 $\mathcal{C}$ 是显而易见的。[7] 此后，我们将使用这样的 $\mathcal{C}$ 。在附录 B 中，我们证明：

引理 4.1。考虑带颜色的范围楔形计数问题。对于任意实数值 $\lambda \in [1, \sqrt{m}]$，存在一个占用 $\tilde{O}(m^2/\lambda^2)$ 空间的数据结构，能够在 $\tilde{O}(\lambda)$ 时间内回答查询。

具备上述结果后，我们现在回到问题 1，考虑 $Q =$ 楔形。

数据结构。对于每个 $U \in \mathcal{C}$ （其中 U 是 V 的子集），我们通过以下三个步骤添加边，构造一个图 G_U：

1. 将 G_U 初始化为空图，其中不含任何顶点和边。

2. 对于每个顶点 $u \in U$，我们将其在 G（即原始数据图）中的所有边添加到 G_U。添加一条边 $\{u, v\}$ 时，若顶点 v 尚不存在于 G_U 中，则在 G_U 中创建该顶点。

3. 最后，如果某个顶点来自 U，则将其在 G_U 中染为黑色；否则染为白色。

现在，我们在 G_U 上构建引理 4.1 中的数据结构，其使用 $\tilde{O}(|E_U|^2/\lambda^2)$ 空间，其中 E_U 是 G_U 中边的集合˜。根据性质 P4-1， G 中的每条边 $\{u, v\}$ 都可以加入 O 的 E_U 个子集$U \in \mathcal{C}$。因此可得 $\sum_{U\in\mathcal{C}}|E_U| = \tilde{O}(m)$。所有 $U \in \mathcal{C}$ 的数据结构总共占用 $\tilde{O}(m^2/\lambda^2)$ 空间。

˜查询。现在考虑一个区间为 q 的（问题 1）查询。根据性质 P4-2，我们可以在 O(1) 时间内从 $\mathcal{C}$ 中选出 $h = O$(1) 个成员 U_1、...、U_h，以划分 V_q。对于每个 $i \in [1, h]$，在 G_{U_i} 上针对区间 q 执行一次带颜色的范围楔形计数查询。我们返回 h 个查询的输出之和。˜ ˜总查询时间为 $h \cdot O(\lambda) = O(\lambda)$。

为验证正确性，首先注意，由彩色查询在 G_{U_i} 上计数的每个楔形(u, v, w)都满足： $A_u \in q$, $A_w \in q$ （彩色范围楔形计数的定义），并且 $A_v \in q$ （因为 v 为黑色意味着 $v \in U_i \subseteq V_q$）。反过来，每个满足 $\{A_u, A_v, A_w\} \subseteq q$ 的楔形(u, v, w)出现，都只会被计数一次：由在 G_{U_i} 上的彩色查询计数，其中 U_i 是（所有 $i \in [1, h]$ 中）唯一包含 v 的子集。事实上，对于任意满足 $j \neq i$ 的 U_j， v 要么在 G_{U_j} 中不存在，要么是白色；这两种情况下，该楔形都不会被计数。由此即可得出正确性。

5 问题 2：任意子图

现在我们继续处理任意查询模式 Q 对应的问题 2。实际上，我们将解决第 1.2.2 节中定义的范围连接问题。如附录 D 所示，只需稍作转换，即可将我们的数据结构用于证明定理 1.5。

对于一个关系 $R \in \mathcal{R}$ （回想一下， $\mathcal{R}$ 是输入关系的集合；参见第 1.2.2 节），其模式 scheme(R) 是 R中的属性集合。令 $\mathcal{X} = \bigcup_{R\in\mathcal{R}}$scheme($R$)。现在，输入规模 N 可以表示为

[7]只需根据顶点的属性值构建一棵二叉搜索树 T。 T 中的每个节点定义了$\mathcal{C}$中的一个子集，该子集由属性 A_v 存储在该节点子树中的所有 $v \in V$ 组成。众所周知（例如参见 [46]），对于任意区间 q， T中存在 $O(\log\ n)$个 *canonical nodes*，它们的子树彼此不相交，且合起来恰好包含 q中的所有属性值。这些节点可以在 $O(\log n)$时间内找到，并且相对于 V_q满足性质 P4-2。

$\sum_{R\in\mathcal{R}}|R|$。不失一般性，我们假设：(i) $\mathcal{R}$ 中的关系具有不同的模式；(ii) N 是 2 的幂；(iii) 每个属性 $X\in\mathcal{X}$ 的定义域 dom(X) 都包含 $[1,N]$ 中的整数。给定一个区间 $q=[x_1,x_2]$，查询会列出 join($\mathcal{R}$) 中满足以下条件的每个元组 t——即在 $\mathcal{R}$ 上的自然连接结果——对于所有 $X\in\mathcal{X}$，均有 $t[X]\in q$，其中 $t[X]$ 是该元组在属性 X下的值。我们的目标是设计一种空间占用较小的数据结构，以较低的延迟回答此类查询。

使用超图 $\mathcal{G}=(\mathcal{X},\mathcal{E})$ 会比较方便，其中 $\mathcal{E}=\{\text{scheme}(R)\mid R\in\mathcal{R}\}$。给定一条边 $e\in\mathcal{E}$，我们用 R_e 表示模式为 e 的唯一关系 $R\in\mathcal{R}$。对于一个函数 W，它为每个 $e\in\mathcal{E}$赋予一个非负权重 $W(e)$，其权重总和为 $\sum_{e\in\mathcal{E}}W(e)$。如果函数 W满足以下条件，则称其为分数边覆盖：对每个属性 $X\in\mathcal{X}$，均有 $\sum_{e\in\mathcal{E}:X\in e}W(e)\geq 1$。$\mathcal{G}$ 的分数边覆盖数 ρ^* 是所有分数边覆盖中权重总和的最小值。此后，我们将用 W 表示权重总和为 ρ^*的最优赋值函数。

本节的主要结果如下：

定理 5.1。对于范围连接问题（见第 1.2.2 节），给定任意 $\Delta\geq 1$，存在一个占用 $\tilde{O}(N+N^{\rho^*}/\Delta)$ 空间的数据结构，它能够以 $\tilde{O}(\Delta)$ 的延迟回答查询。

5.1 AGM 界的推广

经典的 AGM 界 [4] 指出，$|\text{join}(\mathcal{R})|\leq\prod_{e\in\mathcal{E}}|R_e|^{W(e)}$。接下来，我们将给出这个不等式的一个更一般的版本。

令 $d=|\mathcal{X}|$，并对 d属性施加任意排序：$X_1,X_2,...,X_d$。给定区间$I_1,I_2,...,I_d$，其中对每个 $i\in[1,d]$，都有 $I_i\subseteq\text{dom}(X_i)$，定义 $B(I_1,...,I_d)$ 为 d维盒$I_1\times...\times I_d$。对于关系 $R\in\mathcal{R}$，我们用 $R\ltimes B(I_1,...,I_d)$ 表示元组的集合 $t\in R$，其中 $t[X_i]\in I_i$ 对每个满足 $X_i\in\text{scheme}(R)$ 的 i 均成立。

我们在附录C中证明：

L引理 5.2。令 $\mathbf{I}_i$, $i\in[1,d]$ 为 dom(X_i) 中一组不相交区间。 n:

$$\sum_{I_1\in\boldsymbol{\mathcal{I}}_1}\sum_{I_2\in\boldsymbol{\mathcal{I}}_2}...\sum_{I_d\in\boldsymbol{\mathcal{I}}_d}\prod_{e\in\mathcal{E}}|R_e\ltimes B(I_1,...,I_d)|^{W(e)}\leq\prod_{e\in\mathcal{E}}|R_e|^{W(e)}. \tag{1}$$

为了解 (1) 如何体现 AGM 界，考虑特殊的 $\mathbf{I}_i$，其大小为 $|\text{dom}(X_i)|$；也就是说，$\mathbf{I}_i$ 中的每个区间都是 dom(X_i) 中的一个值，反之亦然。因此，$|R_e\ltimes B(I_1,...,I_d)|$ 要么为 0，要么为 1，使得 (1) 左侧恰好是 $|\text{join}(\mathcal{R})|$。然而，(1) 的真正威力在于允许 $\mathbf{I}_i$ 是任意的不相交区间集合；这一特性对于我们证明定理 5.1 至关重要。

这里有必要说明引理 5.2 为何并非显然。如果将 (1) 中的项 $\prod_{e\in\mathcal{E}}|R_e\ltimes B(I_1,...,I_d)|^{W(e)}$ 替换为 $\{R_e\ltimes B(I_1,...,I_d)\mid e\in\mathcal{E}\}$中关系的连接结果大小，那么结论便是显然的。根据 AGM 界，项 $\prod_{e\in\mathcal{E}}|R_e\ltimes B(I_1,...,I_d)|^{W(e)}$ 是连接 $\{R_e\ltimes B(I_1,...,I_d)\mid e\in\mathcal{E}\}$大小的上界。真正不平凡的目标是证明，所有这些上界的总和（即 (1) 的左侧）仍不超过 $\prod_{e\in\mathcal{E}}|R_e|^{W(e)}$。

5.2 范围连接

本小节旨在证明定理 5.1。给定一个 $\ell\geq 0$，若某个区间具有形式 $[i\cdot 2^\ell+1,(i+1)\cdot 2^\ell]$，其中 $i\geq 0$ 为某个整数，则称该区间为第 ℓ 级二进制区间。由于 N 是 2 的幂，对于每个 $\ell\in[0,\log_2 N]$，我们都可以将 $[1,N]$ 划分为 $N/2^\ell$ 个互不相交的第 ℓ 级二进制区间。

二进制组合是一个由 d 个二进制区间组成的序列（$I_1, ..., I_d$）；回忆一下，$d = |\mathcal{X}|$。该组合在 $\{R_e \ltimes B(I_1, ..., I_d) \mid e \in \mathcal{E}\}$ 中的关系上定义了一个（自然的）连接实例。我们将该实例记为 $\mathcal{R}_{I_1,...,I_d}$。定义

$$\mathrm{AGM}(I_1, ..., I_d) = \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, ..., I_d)|^{W(e)}. \tag{2}$$

AGM 界保证了 $|\text{连接}(\mathcal{R}_{I_1, ..., I_d})| \leq \mathrm{AGM}(I_1, ..., I_d)$。

数据结构。若二进制组合 $(I_1, ..., I_d)$ 的连接$(\mathcal{R}_{I_1, ..., I_d})$非空，则称其为重组合，当 $\mathrm{AGM}(I_1, ..., I_d) > \Delta$；否则称其为轻组合。对于每个重组合，我们都会构建一个 [20]˜数据结构，用于枚举连接$(\mathcal{R}_{I_1, ..., I_d})$中的元组，且延迟为 $O(\Delta)$。该数据结构占用的空间不超过 $O(\mathrm{AGM}(I_1, ..., I_d)/\Delta)$。[8]

我们说明，所有重（二进制）组合对应的数据结构总共占用 $\tilde{O}(N^{\rho^*}/\Delta)$ 空间。固定 d 个任意层级编号 $\ell_1, ..., \ell_d$，每个编号都介于 0 和 $\log_2 N$ 之间。对于 $i \in [1, d]$，令 $\mathbf{I}_i$ 为所有层级为 ℓ_i 的二进制区间组成的集合。所有重组合 $(I_1, ..., I_d) \in \mathbf{I}_1 \times ... \times \mathbf{I}_d$ 的数据结构所占用的总空间为

$$\frac{1}{\Delta} \sum_{(I_1,...,I_d) \in \boldsymbol{\mathcal{I}}_1 \times ... \times \boldsymbol{\mathcal{I}}_d} \mathrm{AGM}(I_1, ..., I_d). \tag{3}$$

˜至多相差一个 $O(1)$因子。上面包含了每个轻组合对应的一项，但这些项只会高估空间。每个 $\mathbf{I}_i$ 都是 $\mathrm{dom}(X_i)$ 中由不相交区间组成的集合。根据定义 (2) 和引理 5.2，可以看出 (3) 有界于 N^{ρ^*}/Δ，注意到 (1) 右侧至多为 N^{ρ^*}。

在上述分析中，我们固定了一个 ℓ_1、...、ℓ_d 的集合。由于每个 ℓ_i 有 $O(\log N)$ 种选择，因此总共有 $O(\log^d N) = \tilde{O}(1)$ 个不同的 ℓ_1、...、ℓ_d 集合。由此可得，整体空间为 $\tilde{O}(N^{\rho^*}/\Delta)$。

最后，我们需要一个哈希表，以便在常数时间内判断一个二进制组合是否为重组合。哈希表占用 $\tilde{O}(N^{\rho^*}/\Delta)$ 空间，因为先前的分析表明，重二进制组合的数量有界于 $\tilde{O}(N^{\rho^*}/\Delta)$。因此，整个数据结构的总体空间为$\tilde{O}(N + N^{\rho^*}/\Delta)$，其中 $\tilde{O}(N)$ 这一项表示存储 $\mathcal{R}$ 的关系所需的空间。

查询。考虑一个区间为 $q = [x_1, x_2]$ 的范围连接查询。不失一般性，我们假设 x_1 andx_2 是 $[1, N]$ 中的整数。在 $\tilde{O}(1)$ 时间内，我们可以将盒 $B(q, ..., q)$ 划分为 $O(\log^d N) = \tilde{O}(1)$⏞$t$

个互不相交的盒子，每个盒子的形式为 $B(I_1, ..., I_d)$，其中 $(I_1, ..., I_d)$ 是一个二进制组合；我们称 $(I_1, ..., I_d)$ 对于 q 是规范的。查询结果为

$$\bigcup_{\text{canonical } (I_1, ..., I_d)} join(\mathcal{R}_{I_1,...,I_d}).$$

所有规范（I_1,..., I_d）的连接结果 join$(\mathcal{R}_{I_1,...,I_d})$ 两两不相交。如果某个规范（I_1,..., I_d）是˜重的，我们利用 [20] on（I_1, ..., I_d）的数据结构，以 $O(\Delta)$ 的延迟枚举 join$(\mathcal{R}_{I_1,...,I_d})$。否则，我们应用最坏情况最优连接算法 [39, 40, 48]，计算 join$(\mathcal{R}_{I_1,...,I_d})$。

[8]严格来说，空间开销还应计入 $\mathcal{R}_{I_1,...,I_d}$ 中的关系。在我们的上下文中，只需存储一次 $\mathcal{R}$ 的关系，并在回答查询时生成 $\mathcal{R}_{I_1,...,I_d}$ 中的关系即可。附录 D 对此有更多详细说明: [20]。

~ ~该算法在 O(AGM(I_1, ..., I_d)) 时间内完成，根据轻 dyadic~组合的定义，这为 $O(\Delta)$。我们的算法保证延迟为 $O(\Delta)$。定理 5.1 的证明至此完成。

备注。在 [36], Khamis 等人使用二进制区间设计了算法，用于一次性计算 join($\mathcal{R}$)。他们主要的技术难题是选择"良好"的二进盒（即形如$B(I_1, ..., I_d)$的盒），以便恰好覆盖 join($\mathcal{R}$) 中的元组一次。这个问题在我们的情境中并不存在；我们的主要障碍是论证式（3）中给出的总空间是可承受的。我们借助引理 5.2 克服了这一障碍。尽管鉴于 AGM 界已有的各种变体，这或许不再令人惊讶，但我们认为它仍值得仔细阐述，而且此前似乎尚未有人这样做过。

6 问题 2：三角形

本节将介绍一种用于 $Q =$ triangle 问题 2 的数据结构。我们将首先在第 6.1 节和第 6.2 节讨论两个基础问题；解决这两个问题对于证明定理 1.6 至关重要，该定理的证明将在第 6.3 节给出。

6.1 范围三角形边问题

本小节将讨论以下独立问题。

> 范围三角形边（RTE）。设 G 是一个有 m 条边的无向图。给定一个区间 $q = [x_1, x_2]$，查询将返回：(i) 在 G_q 中至少一个三角形里出现的所有边；以及 (ii) $\Theta(m^*)$ 个三角形，其中 m^* 是 (i) 中报告的边数。

~ 我们将构建一个占用 $O(m)$ 空间的数据结构，能够在 $O(m^*)$ 时间内回答查询。此外，该查询可以枚举 m^* 条边和 $\Theta(m^*)$ 个三角形，二者的延迟均为 Δ。

我们将 G 中的一个三角形出现表示为 triangle(u, v, w)，其中 u、v 和 w 是该三角形的顶点。顺序很重要：我们始终遵循约定 $A_u < A_v < A_w$。给定一个区间 q，我们用 E_q^* 表示在 G_q 中至少一个三角形里出现的边的集合。因此，$m^* = |E_q^*|$。如果 triangle(u, v, w) 出现在 G_q 中，我们称 $\{u, v\}$ 为类型 1 边，$\{v, w\}$ 为类型 2 边，$\{u, w\}$ 为类型 3 边。三种类型边的总数介于 m^* 和 $3m^*$ 之间。[9]。接下来，我们将说明如何在 G_q 中提取每种类型的边。

分为类型 1 和类型 2。我们只讨论类型 1，因为类型 2 是对称的。对于 $\{u, v\}$ in$G$ 中的每条边（不妨设为 $A_u < A_v$），按如下方式为 $\{u, v\}$ 确定一个哨兵顶点 w^*：

- $w^* =$如果 G没有形如 triangle(u, v, w) 的出现，则为 null；
- 否则，w^* 是在 G 中构成三角形出现 triangle(u, v, w) 的所有顶点 w 中属性最小的顶点。

考虑任意区间 $q = [x_1, x_2]$。注意，当且仅当$x_1 \le A_u$ 且 $A_{w^*} \le x_2$时，$\{u, v\}$是 q的一条类型 1 边。这启发我们将类型 1 边的检索转化为对二维点进行范围报告（第 2 节中介绍）。为此，创建一个点集合 P，对于每个哨兵 w^*非空的 $\{u, v\}$，在集合中加入一个点（A_u, A_{w^*}）。将边 $\{u, v\}$附加到点（A_u, A_{w^*}），这样就能连同后者一起取回前者。P 的大小至多为 m。给定 $q = [x_1, x_2]$，我们可以通过枚举 P 中位于矩形内的点，找到所有类型 1 边

[9]一条边在不同的三角形出现中可能属于不同的类型。

（ $[x_1, \infty)$ × $(-\infty, x_2]$）。因此，我们可以将 P存储在 Chazelle 结构（见第 2 节）中，该数据结构占用 $O(|P|) = O(m)$ 空间，并确保报告任意 q的类型 1 边时延迟为 $O(1)$。

类型 3。类似的方法也适用于类型 3。设 $\{u, w\}$ 是至少出现在 G 中一个出现三角形 (u, v, w) 中的边。当且仅当 $x_1 \leq A_u$ 和 $A_w \leq x_2$ 时，它才是 $q = [x_1, x_2]$ 的类型 3 边。通过直接调整前面的讨论，我们可以得出结论：存在一个占用 $O(m)$ 空间的数据结构，使我们能够以 $O(1)$ 的延迟检索所有类型 3 边。

枚举 $\Theta(m^*)$ 个三角形。以上内容已经说明了如何检索 E_q^*，但 RTE 查询仍需报告 $\Theta(m^*)$ 个三角形。接下来，我们稍作修改此前的解法来解决这个问题。

回想一下，在处理类型 1 时，我们将边 $\{u, v\}$ 附加到由该边生成的点 (A_u, A_{w^*}) 上。现在，我们也将三角形 (u, v, w^*) 附加到 (A_u, A_{w^*}) 上。这样，找到 (A_u, A_{w^*}) 时，我们就能免费获得 $\{u, v\}$ 和三角形 (u, v, w^*)。对类型 2 和类型 3 应用相同的思路后，我们可以断定，每当查询算法找到一条类型 1、类型 2 或类型 3 的边时，它也一定找到了 G_q 中的一个三角形。因此，该算法可以以 $\tilde{O}(1)$ 的延迟报告 G_q 中的三角形，不过同一个三角形最多可能被报告三次[10]。通过应用第 2 节中的去重技术，我们现在得到了一个算法，可以以 $O(1)$ 的延迟枚举 $\Theta(m^*)$ 个不同的三角形。报告的不同三角形数量至少为 $m^*/3$，至多为 $3m^*$。

6.2 小延迟三角形枚举问题

在本小节中，我们将集中讨论如下定义的一个独立问题。

> 小延迟三角形枚举（SDTL）。G 是一个具有 m 条边的无向图，其中每条边至少出现在一个三角形中。给定 $\Omega(m)$ 个免费三角形和 $O(m)$ 个禁用三角形。设计一种算法，以较小的延迟枚举 G 的所有三角形——禁用三角形除外——（必须枚举免费三角形）。不允许进行预处理。

我们将通过一种延迟为 $\tilde{O}(m^{\frac{\omega-1}{\omega+1}})$ 的算法解决该问题。

假设 G 总共有 OUT 个三角形。我们的出发点是 Bjorklund 等人的一种算法 [11]，该算法能够在 $\alpha \cdot m^{\frac{3(\omega-1)}{\omega+1}} k^{\frac{3-\omega}{\omega+1}}$时间内枚举 k 个三角形，其中 $\alpha = \tilde{O}(1)$，对于参数 $k \in [\Omega(m), \text{OUT}]$。就 Bjorklund 等人的算法 [11] 而言，我们可以认为 OUT 是已知的，因为它可以在 $O(m^{2\omega/(\omega+1)})$ 时间内求得 [3]，即 $O(m^{\frac{3(\omega-1)}{\omega+1}} k^{\frac{3-\omega}{\omega+1}})$。Bjorklund 等人的算法没有小延迟，但我们将把它改造成一种具有小延迟的算法。

我们运行 Bjorklund 等人的算法 [11]，采用几何级数递增的 k，并且每次运行只报告部分三角形，而非全部三角形。每次运行报告多少个三角形，将通过策略性安排来决定，以保持较小的延迟。设 S_{no}^0 为初始时的禁用三角形集合，S_{yes}^0 为初始时的免费三角形集合。令 $k_0 = |S_{no}^0| + |S_{yes}^0|$。第 i 次运行 Bjorklund 等人的算法 [11] 时，我们将其参数 k 设为 $k_i = \min\{3^i k_0, \text{OUT}\}$。我们保持这样一个不变量：第 i 次运行开始时，始终存在禁用三角形集合 S_{no}^{i-1} 和免费三角形集合 S_{yes}^{i-1}。集合 S_{yes}^{i-1} 将在第 i 次运行期间以较小的延迟报告（具体细节稍后说明）。

[10]只有当 $\{u, v\}$、$\{v, w\}$ 或 $\{u, w\}$ 分别作为类型 1、2 或 3 的边输出时，才能报告出现 $triangle(u, v, w)$。

具体而言，假设第 i 次运行找到一个包含 k_i 个三角形的集合 S^i_{raw}（其中一些已在此前的运行中输出）。我们按如下方式生成下一次运行所需的禁用集合和自由集合：

$$S^i_{no} = S^{i-1}_{no} \cup S^{i-1}_{yes} \text{ and then } S^i_{yes} \;=\; S^i_{raw} \setminus S^i_{no}.$$

运行 i 在 $\alpha \cdot m^{\frac{3(\omega-1)}{\omega+1}} k_i^{\frac{3-\omega}{\omega+1}}$ 时间内结束。我们指示该运行每执行 S^{i-1}_{yes} 一次

$$\alpha \cdot \frac{m^{\frac{3(\omega-1)}{\omega+1}} k_i^{\frac{3-\omega}{\omega+1}}}{|S^{i-1}_{yes}|} \tag{4}$$

次原子操作 。我们将证明 $|S^{i-1}_{yes}| = \Omega(k_i)$，据此可将式 (4) 中的延迟界定为 s:

$$\tilde{O}\left(\frac{m^{\frac{3(\omega-1)}{\omega+1}}}{k_i^{\frac{2\omega-2}{\omega+1}}}\right) = \tilde{O}\left(m^{\frac{\omega-1}{\omega+1}}\right) \tag{5}$$

其中所用的等式为 $k_i \geq k_0 = \Omega(m)$。

对于 $i = 1$，根据 SDTL 问题的定义（即一开始就有 $\Omega(m)$ 个免费三角形），可直接得出 $|S^{i-1}_{yes}| = \Omega(k_0)$。为了证明 $|S^{i-1}_{yes}| = \Omega(k_i)$ 对于 $i \geq 2$ 成立，我们推导如下：

$$|S^{i-1}_{no}| \leq |S^0_{no}| + |S^0_{yes}| + \sum_{j=1}^{i-2} |S^j_{raw}| = k_0 + \sum_{j=1}^{i-2} 3^j \cdot k_0 = \sum_{j=0}^{i-2} 3^j \cdot k_0 < \frac{3^{i-1}k_0}{2}.$$

因此：

$$|S^{i-1}_{yes}| \;\; \geq \;\; |S^{i-1}_{raw}| - |S^{i-1}_{no}| > k_{i-1} - 3^{i-1}k_0/2 = k_{i-1}/2 = \Omega(k_i).$$

由此我们得出结论：我们的算法的延迟如式 (5) 所示。

6.3 定理 1.6 的证明

现在我们准备说明如何用 $Q =$ 三角形解决问题 2。预处理时，我们在 G 上构建一个 RTE 结构（第 6.1 节）。现在考虑一个区间为 q 的（问题 2）查询。我们首先执行一次 RTE 查询，以检索 E^*_q，即至少出现在 G_q 的一个三角形中的边的集合。实际上，这会生成 G^*_q，它是由 E^*_q 中的边诱导的 G_q 的子图。此外，RTE 查询还枚举了 G_q 中一组 $\Theta(m^*)$ 三角形 S，其中 $m^* = |E^*_q|$。S 的大小属于 $[\frac{m^*}{3}, 3m^*]$。

我们剩下的任务是枚举 G^*_q 中位于 S 之外的三角形。注意，G^*_q 是一个有 m^* 条边且至少包含 $\Theta(m^*)$ 个三角形的图。这促使我们将任务转化为 SDTL 问题，该问题已在第 6.2 节中得到解决。然而，SDTL 问题要求将 $\Theta(m^*)$ 个免费三角形和 $O(m^*)$ 个禁用三角形作为输入的一部分。遗憾的是，目前我们似乎还没有这些三角形。

有趣的是，我们通过将 S 划分为 S_{yes} 和 S_{no} 来克服这一障碍，使得 S_{yes}（分别）作为自由（分别）禁止三角形的集合。回想一下，RTE 查询算法~（记作 $\mathcal{A}$）旨在枚举 E^*_q 中的一条边，延迟为 $\Delta = O(1)$，并枚举 S 中的一个三角形，延迟也为 Δ。因此，它必须在 $t_{max} = \max\{\Delta\cdot(|E^*_q|+1),\ \Delta\cdot(|S|+1)\} \leq \Delta\cdot(3m^*+1)$ 时间内完成。现在，我们可以结合 $\alpha = 18$ 应用第 2 节中的缓冲技术，将 $\mathcal{A}$ 转化为一种算法

该算法在每个轮次结束时输出一个三角形，每个轮次的长度为 18Δ。轮次数最多为 $\frac{t_{max}}{18\Delta} \leq \frac{3m^*+1}{18}$。因此，当 $\mathcal{A}$ 完成时，我们至多已输出 $(3m^*+1)/18$个三角形，而缓冲区 B（定义见第 2 节）中仍至少有 $|S| - \frac{3m^*+1}{18} = \Theta(m^*)$ 个三角形。因此，我们可以在 $\mathcal{A}$ 完成时，将 B 的内容设为 S_{yes}，并将已输出的三角形设为 S_{no}。

现在，我们可以对 G_q^* 应用 SDTL 算法，从而完成定理 1.6 的证明。

7 问题 2：近常数延迟

˜本节将重点讨论问题 2 的两个实例，在这两个实例中，可以在空间远小于定理 1.5 所需空间的情况下实现 $O(1)$ 延迟。我们将先在第 7.1 节讨论 $Q = \ell$-星形，再在第 7.2 节讨论 $Q = 2\ell$-环。我们将重点说明如何以 $O(1)$ 延迟枚举可能并非-˜互异的出现，同时确保每个出现仅输出常数次。借助第 2 节中的去重方法，我们可以修改˜算法，以 $O(1)$ 延迟仅枚举互异的出现。

7.1 ℓ-星形

回顾一下，ℓ-星形是一种树，其中只有一个非叶节点，我们称之为星形中心。考虑一个区间为 q 的查询。若节点 u 满足 G_q 至少有一个以 u 为中心的 ℓ-星形出现，则称节点 u 为 q-中心。一旦找到 u，以 u 为中心枚举所有ℓ-星形就变得轻而易举，且延迟为 $O(1)$。具体来说，我们可以先（使用二叉搜索树）检索 G 中满足 $A_v \in q$ 的 u 的所有邻居 v。从这些邻居中任取 ℓ 个互异的˜顶点，再与作为中心的 u 一起，就构成一个 ℓ-星形。只需简单处理，即可确保枚举所有这些星形时的延迟为 $O(1)$。

˜接下来，我们着重设计一种数据结构，以延迟 $O(1)$ 枚举 q-中心。考虑 G 中任意一个中心为 u的 ℓ-星。将该星的 $\ell+1$ 个顶点按属性升序排列，并查找 u的位置。如果 u 是第 r 小的，我们就称该星为秩为 r ℓ 的星，并称 u 为秩为 r q 的中心。

现在，固定一个 n $r \in [1, \ell+1]$。我们将介绍一种数据结构，用于支持以下操作：

给定一个区间 q，找出所有秩为 r q 的中心，即所有满足以下条件的顶点 $u \in V$：G_q 具有一个以 u 为中心的秩为 $r\ell$ 的星。

考虑 G 中以 u 为中心的任意一个秩为 r ℓ 的星。将该星的顶点按属性升序写为 $v_1, ..., v_{r-1}, u, v_{r+1}, ..., v_\ell$。对于一个 $q = [x_1, x_2]$，当且仅当 $x_1 \leq A_{v_1}$ 且 $A_{v_\ell} \leq x_2$ 时，ℓ 星才会出现在 G_q 中。将 v_1 称为 u 的左 r 哨兵，将 v_ℓ 称为 u 的右 r 哨兵。从 u 的所有左 r 哨兵中（每个以 u 为中心、秩为 r ℓ 的星各取一个），找出属性最大的那个 v_1^*。类似地，从 u 的所有右 r 哨兵中，找出属性最小的那个 v_ℓ^*。注意，当且仅当 $x_1 \leq A_{v_1^*}$ 且 $A_{v_\ell^*} \leq x_2$ 时，u 才是秩为 r q 的中心。因此，我们可以将检索秩为 r q 的中心转化为对二维点的范围报告（回顾第 2 节），方法与第 6.1 节中的示例相同。按照第 2 节的方法，我们可以为 n 个点构建一个 Chazelle 结构——每个点都按所述方式为 ˜ 中的一个顶点 $u \in V$ 创建——该结构占用 $O(n)$ 空间，并且对于任意 q，都能以 $O(1)$ 的延迟列出秩为 r q 的中心。定理 1.7 的证明至此完成。

7.2 2ℓ-环

我们先作一个假设：所有查询都指定固定的 $q = (-\infty, \infty)$，也就是说，实际上只有一个查询，它枚举 G 中所有的 2ℓ 环。这个假设使我们能够以最少的技术细节解释核心思想，之后我们最终会去掉这个假设。

使用 $\boldsymbol{q = (-\infty, \infty)}$ 的查询。给定一个 2ℓ-环出现，我们将环中属性最小的顶点 u 称为该次出现的锚点。令 v 为环中满足以下条件的顶点：在 u 和 v 处切开环，会得到两条连接 u 和 v的 ℓ-路径。我们将 v 称为该次出现的逆锚点，将点对 (u, v) 称为锚点对，并将上述两条路径称为环 ℓ-路径。环 ℓ-路径的数量至多为 $\#P_\ell$ （回想一下，$\#P_\ell$ 是 ℓ-路径的总数）。

这个问题看起来可能简单得具有迷惑性：我们能不能只需针对每个锚点对 (u, v)，将从 u 到 v 的任意两条环 ℓ-路径拼接起来，就回答一个查询？这种做法行不通，因为两条环 ℓ-路径可能还共享 u 和 v以外的公共顶点，这种情况下拼接结果并不是一个 2ℓ-环！这引出了一个关键概念：若两条环 ℓ-路径 (i) 具有相同的锚点对 (u, v)，且 (ii) 除了 u 和v之外不共享任何公共顶点，则称它们内部不相交。当且仅当从 u 到 v 的两条环 ℓ-路径内部不相交时，将它们拼接才会形成一个 2ℓ-环。此刻我们面临的挑战是以下问题。

> 设计一种数据结构，支持以下操作：给定一条从锚点 u~到逆锚点 v 的环 ℓ-路径 π，以 $O(1)$ 的延迟列出所有与 π 内部不相交的环 ℓ-路径。

~我们将借助一个占用 $O(\#P_\ell)$ 空间的数据结构来克服这一挑战。

我们的主要观察是，该操作可以转化为对 $(\ell - 1)$ 维点进行范围报告（参见第 2 节）。具体来说，考虑任意一条从锚点u 到逆锚点 v 的环 ℓ-路径 π。去除 u 和 v后，该路径还剩下 $\ell - 1$ 个顶点。按属性[11]升序将它们依次记为w_1、w_2、...、$w_{\ell-1}$。将 π 转换为一个 $(\ell - 1)$ 维点 $(A_{w_1}, ..., A_{w_{\ell-1}})$。令 $P_{u,v}$为由所有以 (u, v) 为锚点对的环 ℓ-路径所得到的点构成的集合。

现在，考虑另一条从 u 到 v 的环 ℓ-路径 π'。同样，按属性升序将 π' 中除 u 和 v以外的顶点依次记为 w'_1、w'_2、...、$w'_{\ell-1}$。如果 π' 与 π内部不相交，那么每个 $A_{w'_i}(i \in [1, \ell - 1])$ 都必须落在以下某个 ℓ 开区间内：

$$(-\infty, A_{w_1}), (A_{w_1}, A_{w_2}), ..., (A_{w_{\ell-2}}, A_{w_{\ell-1}}), (A_{w_{\ell-1}}, \infty). \tag{6}$$

因此，$(A_{w'_1}, ..., A_{w'_{\ell-1}})$——由 π' 转换而来的点——必定落在以下某个$\ell^{\ell-1} = O(1)$ 矩形中：$q_1 \times q_2 \times ... \times q_{\ell-1}$，其中每个 $q_i(i \in [1, \ell - 1])$ 均独立地取自~(6) 中的某个区间。根据第 2 节，通过在 $P_{u,v}$ 的 $O(|P_{u,v}|)$ 维空间上创建范围树，我们~可以以 $O(1)$ 的延迟枚举此类矩形中的所有点。

由此可得，对于每个锚点对 (u, v)，我们可以创建一个占用 $O(|P_{u,v}|)$ 维空间的范围树~。给定任意一个从 u 到 v 的环 ℓ-路径环 π，该范围树即可枚举出~每个与 π 内部不相交的环 ℓ-路径 π'，延迟为 $O(1)$。所有~~锚点对的数据结构总共占用 $\sum_{\text{anc. pair } (u,v)} O(|P_{u,v}|) = O(\#P_\ell)$ 空间。

攻克这一难题后，枚举所有 2ℓ环就变得轻而易举。我们只需查看每个环 ℓ路径 π，找出与 π内部不相交的每条 ℓ路径 π'，然后通过~拼接 π 和 π'来构成一个环。环报告的延迟为 $O(1)$（每个 2ℓ环最多报告两次）。

[11]这个顺序不应与顶点在 π 中出现的顺序混淆。

任意查询。接下来，我们移除约束 $q = (-\infty, \infty)$，处理具有任意 q的查询。现在出现了一个新问题：查询再也无法遍历所有环 ℓ路径。我们称从锚点 u 到逆锚点 v 的环 ℓ路径对 G_q 有贡献，当且仅当它与另一条内部不相交的环 ℓ路径在 G_q 中构成一个 2ℓ环。我们需要一种方法，只枚举有贡献的环ℓ路径。

固定任意一个带锚点对 (u, v) 的环 ℓ-路径 π。令 S_π 为 G 中包含 π 且以 (u, v) 为锚点对的 2ℓ-环集合。从 S_π 中任取一个环。根据锚点的定义，u 的属性在该环的所有顶点中最小。令 w 为该环中属性最大的顶点。对于 $q = [x_1, x_2]$，当且仅当 $x_1 \le A_u$ 且 $A_w \le x_2$ 时，该环才会出现在 G_q 中。令 w^* 为所有此类 w 中属性最小的顶点。显然，当且仅当 $x_1 \le A_u$ 且 $A_{w^*} \le x_2$ 时，π 才会对 $q = [x_1, x_2]$ 的 G_q 作出贡献。因此，我们可以使用第 6.1 节中的方法，将检索有贡献的环 ℓ-路径转化为对二维点进行范围报告。生成的数据结构（即 Chazelle 结构）为每条环 ℓ-路径存储一个转换得到的点，并使用 O P_ℓ q ℓ G_q (#) 空间。给定任意 ，我们可以以 O(1) 的延迟列出对 有贡献的环 - 路径。

假设我们已找到一条有贡献的环 ℓ-路径 π，其锚点对为 (u, v)。和之前一样，我们继续寻找与 π内部不相交的环 ℓ-路径 π'。不过，这里新增的要求是 π' 也必须有贡献。回想一下，在 $q = (-\infty, \infty)$ 的情形中，我们将问题转化为对 $(\ell - 1)$ 维点进行范围报告。为了处理任意 $q = [x_1, x_2]$，我们将维度增加一维。

类似之前的做法，为了说明这一点，我们将 π 的顶点列出——排除 u 和 v 后——按属性升序排列为 w_1,..., $w_{\ell-1}$。用 $w_{\max}$ 表示 π 中属性最大的顶点（$w_{\max}$ 可以是 v）。将 π 转换为一个 ℓ 维点 $(A_{w_1}, ..., A_{w_{\ell-1}}, A_{w_{\max}})$。令 $(A_{w'_1}, ..., A_{w'_{\ell-1}}, A_{w'_{\max}})$ 为以相同方式从 π' 转换得到的点。我们已知 $A_u \in [x_1, x_2]$（回想一下，π 是一条有贡献的路径），当且仅当满足以下条件时，π' 是我们想要的路径：

- $A_{w'_i}$ $(1 \le i \le \ell - 1)$ 落在 (6) 中的某个 ℓ 区间内；
- $A_{w'_{\text{最大值}}} \le x_2$.

因此，点 $(A_{w'_1}, ..., A_{w'_{\ell-1}}, A_{w'_{\max}})$ 必须落在以下 $\ell^{\ell-1} = O(1)$ 个矩形之一中：$q_1 \times q_2 \times ... \times q_{\ell-1} \times (-\infty, x_2]$，其中每个 q_i $(i \in [1, \ell - 1])$ 都是从 (6) 中独立选取的区间。根据上述推理，对于每个锚点对 (u, v)，我们创建一个由 $P_{u,v}$ 个 ℓ维点组成的集合，每个点都由一个环 ℓ路径转换而来，其锚点对为 (u, v)，然后在 $P_{u,v}$上构建一棵范围树。所有锚点对的范围树总共占用 $\sum_{\text{锚点对 }(u,v)} O(|P_{u,v}|) = O(\#P_\ell)$ 空间。

现在，我们详细说明如何使用参数q回答（问题-2）查询的整体算法。首先，以 O(1)的延迟枚举所有对 G_q 有贡献的环 ℓ-路径；我们称之为外层枚举。每当得到这样一条路径 π ——例如，其锚点对为（u，v）——我们就暂停外层枚举，并利用$P_{u,\ v}$ 上的范围树，以 O(1)的延迟找出先前讨论的所有路径 π' 。每交付一条 π'，就将其与 π 拼接，并输出所得的 2ℓ-环。穷尽所有此类 π'后，我们恢复外层枚举。至此，定理 1.8 的证明完成。

附录

第 3.1 节归约的正确性

在我们的构造中， S_i ($i \in [1, s]$)对应两个集合顶点，其属性值分别为 i 和$i + s$。为便于推导，我们复制每个集合：对每个$i \in [s + 1, 2s]$，定义 $S_i = S_{i-s}$。在证明的其余部分，我们采用这样的观点：每个 S_i($i \in [1, 2s]$)仅对应一个集合顶点，即属性值为 i的集合顶点。

考虑一个顶点为 u、v和 w 的楔形出现，其中边为 $\{u, v\}$ 和 $\{v, w\}$。我们将其分为以下两种类型之一：

- （类型 e-s-e） u 和 w 是元素顶点， v 是集合顶点；
- （类型 s-e-s） u 和 w 是集合顶点， v 是元素顶点。

引理 A.1. 对于满足 $1 \le x < s + 1/2 < y \le 2s$ 的任意区间 $q = [x, y]$，我们有

- G_q 中元素-集合-元素楔形的数量为 $\sum_{i\in[x,y]} \binom{|S_i|}{2}$；
- G_q 中集合-元素-集合楔形的数量为 $\sum_{i\in[x,y],\ j\in[i+1,y]} |S_i \cap S_j|$。

证明。为证明第一条，定义元素-集合-元素元组为 (e_1, S_i, e_2)，其中 $i \in q$、 e_1 和 e_2 是 S_i 中互异的元素。此类元组的数量为 $\sum_{i=x}^{y} \binom{|S_i|}{2}$。我们的构造确保元素-集合-元素元组与 G_q 中的元素-集合-元素楔形一一对应。

为证明第二条，定义集合-元素-集合元组为 (S_i, e, S_j)，其中 $x \le i < j \le y$ 和$e \in S_i \cap S_j$。此类元组的数量为 $\sum_{i\in[x,y],\ j\in[i+1,y]} |S_i \cap S_j|$。我们的构造确保集合-元素-集合元组与 G_q 中的集合-元素-集合楔形一一对应。 □

为了确定 $S_a \cap S_b$ 是否为空，我们的归约分别使用区间为 $q_1 = [a, s+b]$、 $q_2 = [a+1, s+b]$、$q_3 = [a, s+b-1]$和 $q_4 = [a+1, s+b-1]$ 的四个问题1查询。上述引理适用于所有这些区间。对于 $i \in [1, 4]$，令 c'_i（分别为 c''_i）表示 G_{q_i} 中 e-s-e（分别为 s-e-s）楔形的数量；这意味着 c_i（即 G_{q_i} 中楔形的总数）等于 $c'_i + c''_i$。根据引理 A.1，我们有：

$$
\begin{aligned}
& c'_1 - c'_2 - c'_3 + c'_4 \\
= & \sum_{i\in[a,s+b]} \binom{|S_i|}{2} - \sum_{i\in[a+1,s+b]} \binom{|S_i|}{2} - \sum_{i\in[a,s+b-1]} \binom{|S_i|}{2} + \sum_{i\in[a+1,s+b-1]} \binom{|S_i|}{2} \\
= & \ 0
\end{aligned}
$$

and

$$
\begin{aligned}
& c''_1 - c''_2 - c''_3 + c''_4 \\
= & \left(\sum_{\substack{i\in[a,s+b] \\ j\in[i+1,s+b]}} |S_i \cap S_j| - \sum_{\substack{i\in[a+1,s+b] \\ j\in[i+1,s+b]}} |S_i \cap S_j| \right) - \\
& \left(\sum_{\substack{i\in[a,s+b-1] \\ j\in[i+1,s+b-1]}} |S_i \cap S_j| - \sum_{\substack{i\in[a+1,s+b-1] \\ j\in[i+1,s+b-1]}} |S_i \cap S_j| \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{j\in[a+1,s+b]} |S_a \cap S_j| - \sum_{j\in[a+1,s+b-1]} |S_a \cap S_j| \\
&= |S_a \cap S_{s+b}| \\
&= |S_a \cap S_b|.
\end{aligned}
$$

因此，我们得出结论： $c_1 - c_2 - c_3 + c_4 = |S_a \cap S_b|$。

B 引理 4.1 的证明

我们首先考虑集合不相交问题的一种变体。

加权集合交集大小。设有 $s \geq 2$ 个集合 S_1、S_2、.....、S_s。每个 S_i ($i \in [1, s]$) 都关联一个权重函数 $_{S_i}$，该函数为每个元素 $e \in S_i$ 赋予一个值 weight$_{S_i}$(e)。给定不同的集合 ID a、$b \in [1, s]$，一次查询返回

$$size(S_a, S_b) = \sum_{e\in S_a\cap S_b} weight_{S_a}(e) \cdot weight_{S_b}(e). \tag{7}$$

设 $N = \sum_{i=1}^{s} |S_i|$。对于任意 $\lambda \in [1, \sqrt{N}]$，可以直接构建一个占用 $O(N^2/\lambda^2)$ 空间的数据结构，并在 $O(\lambda)$ 时间内回答查询。若 $|S_i| > \lambda$，则称 S_i ($i \in [1, s]$) 为大集合，否则称为小集合。大集合的数量至多为 N/λ。对于每一对 $(i, j) \in [1, s] \times [1, s]$, $i \neq j$，只要 S_i 和 S_j 都是大集合，我们就存储 size(S_i, S_j)；所需空间为 $O(N^2/\lambda^2)$。给定参数为 a 和 b 的查询，如果 S_a 和 S_b 都是大集合，则直接返回 size(S_a, S_b)。否则，不妨设 S_a 是小集合。我们使用哈希表在 $O(\lambda)$ 时间内计算 $S_a \cap S_b$ （对每个 $e \in S_a$，检查 $e \in S_b$ 是否成立）。然后即可轻松得到结果 size(S_a, S_b)。

有了上述内容，接下来我们描述一个用于带颜色的范围楔形计数问题的数据结构，以证明引理 4.1。

数据结构。首先，获得一个规范集合 $\mathcal{C}$ of V（定义见第 4 节），满足 $\sum_{U\in\mathcal{C}} |U| = O(n)$。对于每个 $U \in \mathcal{C}$ ——回忆一下，U 是 V 的子集——按如下方式构造一个加权集合：

- $S_U =$ 与 U中至少一个顶点相邻的黑色顶点集合；
- 对于每个 $b \in S_U$，权重$_{S_U}$(b) =是 U中与 b相邻的顶点数量。

这些加权集合构成一个加权集合交集大小问题的实例。使用给定参数 λ，在该实例上构建前文描述的数据结构。下面的引理表明，该数据结构占用 $\tilde{O}(m^2/\lambda^2)$ 空间。

˜引理 B.1。 $\sum_{U\in\mathcal{C}} |S_U| = O(m)$。

证明。每个 $b \in S_U$都与一个顶点 $u \in U$相邻。对于每一对这样的顶点 (b, u)，向边 $\{b, u\}$ 支付一美元。由于一条边只有在其包含一个属于 U 的顶点时才能收到一美元，因此最多可以收到两美元[12]。 $|S_U|$ 不超过支付的美元数。对所有 $U \in \mathcal{C}$ 执行上述操作。每条属于 G 的边˜ ˜总共可以收到 $O(1)$ 美元，因为每个顶点都出现在 $\mathcal{C}$ 中的 $O(1)$ 个子集中（ $\mathcal{C}$ 的性质 P4-1；见第 4 节）。

□

[12]有两种可能：当 b 和 u 均为黑色且都出现在 U 中时，就会发生这种情况。

对于任意不同的 $U, U' \in \mathcal{C}$，按式 (7) 定义 size($S_U, S_{U'}$)。另一方面，对于每个 $U \in \mathcal{C}$，定义

$$size(S_U, S_U) \quad = \quad \sum_{b \in S_U} \binom{weight_{S_U}(b)}{2}.$$

我们为所有 U 存储 size(S_U, S_U) 的值。总空间为 $\tilde{O}(m^2/\lambda^2)$。

在继续之前，读者应注意函数 size(., .) 的以下微妙事实：

事实 B-1：size($S_U, S_{U'}$) 是 G 中满足 $u \in S_U$、$w \in S_{U'}$ 且 v 为黑色的楔形 wedge(u, v, w) 的出现次数。

即使 $U = U'$，该事实仍然成立。

~ ~查询。给定一个区间为 q 的查询，我们可以在 $O(1)$ 时间内从 $\mathcal{C}$ 中选出 $h = O(1)$ 个成员 U_1、...、U_h，它们构成 V_q 的一个划分（$\mathcal{C}$ 的性质 P4-2）。查询返回

$$\sum_{i,j \in [1,h]: i \le j} size(S_{U_i}, S_{U_j}). \tag{8}$$

每个 size(S_{U_i}, S_{U_j}) 要么被显式存储，要么可以在 $O(\lambda)$ 时间内从加权集合交集~size 结构中获取。因此，总查询时间为 $O(\lambda)$。

事实 B-1 以及 $U_1, ..., U_h$ 构成 V_q 的一个划分，保证了 (8) 只统计 G 中满足 $A_u \in q$，$A_w \in q$ 且 v 为黑色的楔形(u, v, w) 出现。为了完成正确性论证，我们还需要证明 (8) 将每个这样的出现恰好统计一次。事实上，存在唯一的 $a, b \in [1, h]$ 使得 $a \le b$, $u \in U_a$ 且 $w \in U_b$。该楔形只会由 (8) 中满足 $i = a$ 和 $j = b$ 的项统计。

C 引理 5.2 的证明

我们先回顾 H¨older 不等式。固定正整数 α 和 β。令

- $x_{i,j}$，对每个 $i \in [1, \alpha]$ 和 $j \in [1, \beta]$，均为非负实数；
- y_j，对每个 $j \in [1, \beta]$，均为满足 $\sum_{j=1}^{\beta} y_j \ge 1$ 的非负实数。

按照约定 $0^0 = 0$，H¨older 不等式指出：

$$\sum_{i=1}^{\alpha} \prod_{j=1}^{\beta} x_{i,j}^{y_j} \le \prod_{j=1}^{\beta} \left(\sum_{i=1}^{\alpha} x_{i,j} \right)^{y_j}. \tag{9}$$

证明可见 [29]。

现在我们回到引理 5.2 的背景。给定任意 $j \in [1, d-1]$ 以及 ($I_1, I_2, ..., I_j$) $\in \mathbf{I}_1 \times ... \mathbf{I}_j$，我们将用 $B(I_1, I_2, ..., I_j)$ 作为 d 维盒的简写

$$B(I_1, ..., I_j, dom(X_{j+1}), ..., dom(X_d)).$$

作为一种特殊情况，定义 $B(\emptyset) = B(\text{dom}(X_1), ..., \text{dom}(X_d))$。

引理 C.1. 对于任意 $j \in [1, d]$，我们有

$$\sum_{I_j \in \boldsymbol{\mathcal{I}}_j} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \leq \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, ..., I_{j-1})|^{W(e)}.$$

证明。定义

$$\mathcal{E}_j = \{e \in \mathcal{E} \mid X_j \in e\}.$$

由于 $\sum_{e \in \mathcal{E}_j} W(e) \geq 1$ (W 是一个分数超边覆盖)，根据 H¨older 不等式 (9)，我们有

$$\begin{aligned}
&\sum_{I_j \in \boldsymbol{\mathcal{I}}_j} \prod_{e \in \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \\
\leq\ & \prod_{e \in \mathcal{E}_j} \Big(\sum_{I_j \in \boldsymbol{\mathcal{I}}_j} |R_e \ltimes B(I_1, ..., I_j)| \Big)^{W(e)} \\
\leq\ & \prod_{e \in \mathcal{E}_j} \Big| R_e \ltimes B\big(I_1, ..., I_{j-1}, dom(X_j)\big) \Big|^{W(e)} \\
=\ & \prod_{e \in \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_{j-1})|^{W(e)} \qquad (10)
\end{aligned}$$

其中，第二个不等式用到了 I_j 是 dom(X_j) 中一组不相交区间这一事实。

对于每个 $e \in \mathcal{E} \setminus \mathcal{E}_j$，$R_e \ltimes B(I_1, ..., I_j)$ 不依赖于 I_j，因此可以改写为 $R_e \ltimes B(I_1, ..., I_{j-1})$。由此可得:

$$\begin{aligned}
&\sum_{I_j \in \boldsymbol{\mathcal{I}}_j} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \\
=& \sum_{I_j \in \boldsymbol{\mathcal{I}}_j} \Big(\prod_{e \in \mathcal{E} \setminus \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \cdot \prod_{e \in \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \Big) \\
=& \prod_{e \in \mathcal{E} \setminus \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \cdot \sum_{I_j \in \boldsymbol{\mathcal{I}}_j} \prod_{e \in \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_j)|^{W(e)} \\
\leq& \prod_{e \in \mathcal{E} \setminus \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_{j-1})|^{W(e)} \cdot \prod_{e \in \mathcal{E}_j} |R_e \ltimes B(I_1, ..., I_{j-1})|^{W(e)} \\
=& \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, ..., I_{j-1})|^{W(e)}.
\end{aligned}$$

其中使用了不等式 (10)。 □

我们可以通过应用引理 C.1 的 d 次来证明引理 5.2：

$$\begin{aligned}
& \sum_{I_1 \in \boldsymbol{\mathcal{I}}_1} \ldots \sum_{I_d \in \boldsymbol{\mathcal{I}}_d} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, \ldots, I_d)|^{W(e)} \\
\leq & \sum_{I_1 \in \boldsymbol{\mathcal{I}}_1} \ldots \sum_{I_{d-1} \in \boldsymbol{\mathcal{I}}_{d-1}} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, \ldots, I_{d-1})|^{W(e)} \\
\leq & \sum_{I_1 \in \boldsymbol{\mathcal{I}}_1} \ldots \sum_{I_{d-2} \in \boldsymbol{\mathcal{I}}_{d-2}} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1, \ldots, I_{d-2})|^{W(e)} \\
\leq & \ldots \\
\leq & \sum_{I_1 \in \boldsymbol{\mathcal{I}}_1} \prod_{e \in \mathcal{E}} |R_e \ltimes B(I_1)|^{W(e)} \\
\leq & \prod_{e \in \mathcal{E}} |R_e|^{W(e)}.
\end{aligned}$$

定理 1.5 的证明

读者应在读完第 5 节后再阅读此证明。基本思想是将问题 2 转换为范围连接。令 $\mathcal{X}$ （分别为 $\mathcal{E}$）表示模式图Q中的顶点（分别为边）集合。读者不应将 $\mathcal{X}$ 和 $\mathcal{E}$ 与 V 和 E混淆：后两者定义在数据图 G上。对于每条边 $e \in \mathcal{E}$，构造一个具有两个属性的关系 R_e：对于 G中的每条边 $\{u, v\}$，插入两个元组（u，v）和（v，u）。由此定义一个连接实例 $\mathcal{R} = \{R_e \mid e \in \mathcal{E}\}$，其输入规模为 $N = 2m \cdot |\mathcal{E}| = O(m)$。

Q 的每次出现都对应于连接($\mathcal{R}$)中的常数个元组。受此启发，给定一个区间为 q的问题 2 查询，我们在 $\mathcal{R}$ 上针对 q发出范围连接查询，这保证检索出所有出现次数。然而，问题在于，连接($\mathcal{R}$)中的并非每个元组都会产生一次出现。为说明这一点，考虑 $Q = 4$-环，因此，$\mathcal{R}$ 具有四个模式分别为（X_1，X_2）、（X_2，X_3）、（X_3，X_4）和（X_4，X_1）的关系。令 $\{u, v\}$ 为 E中的任意一条边；四个关系中分别存在元组（u，v）、（v，u）、（u，v）和（v，u）。因此，连接($\mathcal{R}$)包含一个元组（u，v，u，v），但该元组不对应任何出现。

稍微修改一下 [20], 的数据结构即可消除这个问题，接下来我们将回顾这一修改。考虑第 5 节中定义的任意关系集合 $\mathcal{R}$（属性数量不限）。Deep 和 Koutris [20] 证明了存在一个盒集合 $\mathcal{B}$，满足：

- 每个盒的形式为 $B(I_1, \ldots, I_d)$，其中 I_i 是 dom(X_i) 中的一个区间，对应于 $i \in [1, d]$;
- 这些盒两两不相交，且它们的并集为 B(dom(X_1), dom(X_2), ...,dom (X_d));
- 对于每个盒 $B(I_1, \ldots, I_d)$，连接实例 $\mathcal{R}_{I_1, \ldots, I_d}$ 的结果非空;
- 每个盒 $B(I_1, \ldots, I_d)$ 都满足 AGM(I_1, ..., I_d) $\leq \Delta$;
- $|\mathcal{B}| = O(N^{\rho^*}/\Delta)$。

[20] 的数据结构仅存储 $\mathcal{B}$ 本身，并使用 $O(N^{\rho^*}/\Delta)$ 空间[13]。为了枚举 join($\mathcal{R}$)，[20] 的算法会遍历每个 $B(I_1, \ldots, I_d) \in \mathcal{B}$，并应用最坏情况最优连接算法~[39, 40, 48]，在 O(AGM(I_1, ..., I_d)) $= O(\Delta)$ 时间内计算 join($\mathcal{R}_{I_1, \ldots, I_d}$)。这保证了 ~ 延迟为 $O(\Delta)$。

[13]显然 ，$\mathcal{R}$ 的关系也需要单独存储 。

现在，我们调整该数据结构，以列出 G 中 Q 的所有出现次数（固定 $q = (-\infty, \infty)$）。按照之前的方法，从 G 和 Q 构造$\mathcal{R}$ 。应用 [20] ，找到一个满足前文所述所有性质的集合 $\mathcal{B}$ 。然后，依次检查每个盒 $B(I_1, ..., I_d) \in \mathcal{B}$ ；如果 join($\mathcal{R}_{I_1,...,I_d}$) 能产生的所有 Q 出现次数都已经可以由之前检查过的盒产生，就将该盒从 $\mathcal{B}$ 中移除。 $\mathcal{B}$ 的大小只会减小，因此仍然有 $O(N^{\rho^*}/\Delta)$ 的上界。要找到这些出现次数，可对 $\mathcal{B}$ 中的每个盒应用最坏情况最优连接算法。由于每个盒至少会产生一个新的˜出现次数，因此我们保证延迟为 $O(\Delta)$。

为支持任意 q 的（问题-2）查询，在第 5.2 节所给出的解法中，用调整后的数据结构替换 [20] 。所有分析依然成立。至此，我们完成了定理 1.5 的证明。

参考资料

[1] Josh Alman 和 Virginia Vassilevska Williams. 一种改进的激光法及更快的矩阵乘法. 载于 ACM-SIAM离散算法年会（SODA）论文集，第 522–539 页，2021年.[2] Noga Alon、Raphael Yuster 和 Uri Zwick. 彩色编码. ACM会刊（JACM），42(4)：844–856，1995.[3] Noga Alon、Raphael Yuster 和 Uri Zwick. 查找并计数给定长度的环. Algorithmica，17(3)：209–223，1997.[4] Albert Atserias、Martin Grohe 和 Daniel Marx. 关系连接的大小界与查询计划. SIAM 计算杂志，42(4)：1737–1767，2013年.[5] Guillaume Bagan、Arnaud Durand 和 Etienne Grandjean. 关于无环合取查询与常数延迟枚举. 载于计算机科学逻辑，第 208–222 页，2007年.[6] Michael J Bannister、Christopher DuBois、David Eppstein 和 Padhraic S myth. 关系事件之窗：连续边序列的数据结构. 载于 ACM-SIAM离散算法年会（SODA）论文集，第 856–864 页，2013年.[7] Jon Louis Bentley. 可分解搜索问题. 信息处理快报（IPL），8(5)：244–251，1979.[8] Suman K. Bera、Noujan Pashanasangi 和 C. Seshadhri. 有界退化图中的近线性时间同态计数：长诱导环的障碍. 载于 ACM-SIAM离散算法年会（SODA）论文集，第 2315–2332 页，2021年.[9] Christoph Berkholz、Jens Keppeler 和 Nicole Schweikardt. 更新条件下的合取查询应答. 载于 ACM数据库系统原理研讨会（PODS）论文集，第 303–318 页，2017年.[10] Andreas Bjorklund、Petteri Kaski 和 Lukasz Kowalik. 通过打包计数细子图，速度快于 meet-in-the-middle 时间. ACM算法汇刊，13(4)：48:1–48:26，2017年.[11] Andreas Bjorklund、Rasmus Pagh、Virginia Vassilevska Williams 和 Uri Zwick. 三角形枚举. 载于国际自动机、语言与程序设计研讨会（ICALP）论文集，第 223–234 页，2014年。

[12] Nofar Carmeli 和 Markus Kroll. 合取查询的并的枚举复杂度. ACM 数据库系统汇刊（TODS），46(2)：5:1–5:41，2021年.[13] Nofar Carmeli、Shai Zeevi、Christoph Berkholz、Benny Kimelfeld 和 Nicole Schweikardt. 使用随机访问和随机顺序枚举应答合取查询（的并）. 载于 ACM数据库系统原理研讨会（PODS）论文集，第 393–409 页，2020年.[14] Farah Chanchary 和 Anil Maheshwari. 图的时间窗口数据结构. J. Graph Algorithms Appl.，23(2)：191–226，2019年.[15] Farah Chanchary、Anil Maheshwari 和 Michiel Smid. 使用彩色范围查询数据结构查询关系事件图. 离散应用数学，286：51–61，2020年.[16] Bernard Chazelle. 数据结构的函数式方法及其在多维搜索中的应用. SIAM 计算杂志，17(3)：427–462，1988年.[17] N. Chiba 和 T. Nishizeki. 森林度与子图列举算法. SIAM 计算杂志，14(1)：210–223，1985年.[18] Radu Curticapean、Holger Dell 和 D´aniel Marx. 同态是计数小子图的良好基础. 载于 ACM 计算理论研讨会（STOC）论文集，第 210–223 页，2017年.[19] Mark de Berg、Otfried Cheong、Marc van Kreveld 和 Mark Overmars. 计算几何：算法与应用. 施普林格出版社，第 3 版，2008年.[20] Shaleen Deep 和 Paraschos Koutris. 合取查询结果的压缩表示. 载于 ACM数据库系统原理研讨会（PODS）论文集，第 307–322 页，2018年.[21] Arnaud Durand. 查询的细粒度复杂性分析：从判定到计数与枚举. 载于 ACM数据库系统原理研讨会（PODS）论文集，第 331–346 页，2020年.[22] Arnaud Durand 和 Etienne Grandjean. 有界度结构上的一阶查询可通过常数延迟计算. ACM计算逻辑汇刊，8(4)：21，2007年.[23] Arnaud Durand、Nicole Schweikardt 和 Luc Segoufin. 枚举低度数据库上一阶查询的答案. 载于 ACM数据库系统原理研讨会（PODS）论文集，第 121–131 页，2014年.[24] David Eppstein. 森林度与二部子图列举算法. 信息处理快报（IPL），51(4)：207–211，1994年.[25] David Eppstein. 平面图中的子图同构及相关问题. J. Graph Algorithms Appl.，3(3)：1–27，1999年.[26] David Eppstein、Maarten Loffler 和 Darren Strash. 在近最优时间内枚举稀疏图中的所有极大团. 载于国际算法与计算研讨会（ISAAC），第 6506 卷，第 403–414 页，2010年.[27] Peter Floderus、Miroslaw Kowaluk、Andrzej Lingas 和 Eva-Marta Lundell. 检测并计数小型模式图. SIAM离散数学杂志，29(3)：1322–1339，2015年。

[28] Fedor V. Fomin、Daniel Lokshtanov、Venkatesh Raman、Saket Saurabh 和 B. V. Raghavendra Rao。更快地查找和计数子图的算法。《计算机与系统科学杂志》（JCSS），78(3)：698–706，2012年。[29] Ehud Friedgut。超图、熵与不等式。《美国数学月刊》，111(9)：749–760，2004年。[30] Pierre-Louis Giscard、Nils M. Kriege 和 Richard C. Wilson。一种用于计数任意长度简单环和简单路径的通用算法。《Algorithmica》，81(7)：2716–2737，2019年。[31] Isaac Goldstein、Tsvi Kopelowitz、Moshe Lewenstein 和 Ely Porat。空间/时间权衡的条件下界。载于《算法与数据结构研讨会》（WADS），第421–436页。施普林格，2017年。[32] Isaac Goldstein、Moshe Lewenstein 和 Ely Porat。关于有界全集下集合不相交性和集合交的难度。载于《国际算法与计算研讨会》（ISAAC），第7:1–7:22页，2019年。[33] Chinh T. Hoang、Marcin Kaminski、Joe Sawada 和 R. Sritharan。查找和枚举诱导路径与环。《离散应用数学》，161(4-5)：633–641，2013年。[34] Ahmet Kara、Milos Nikolic、Dan Olteanu 和 Haozhe Zhang。层次查询静态与动态求值中的权衡。载于《ACM数据库系统原理研讨会》（PODS）论文集，第375–392页，2020年。[35] Wojciech Kazana 和 Luc Segoufin。在具有有界扩张的数据结构类上枚举一阶逻辑查询。载于《ACM数据库系统原理研讨会》（PODS）论文集，第297–308页，2013年。[36] Mahmoud Abo Khamis、Hung Q. Ngo、Christopher R´e 和 Atri Rudra。通过几何分解实现连接：最坏情况及更广范围。载于《ACM数据库系统原理研讨会》（PODS）论文集，第213–228页，2015年。[37] Ton Kloks、Dieter Kratsch 和 Haiko M¨uller。高效查找和计数小型诱导子图。《信息处理快报》（IPL），74(3-4)：115–121，2000年。[38] Jaroslav Nesetril 和 Svatopluk Poljak。关于子图问题的复杂度。《布拉格查理大学数学评论》，26(2)：415–419，1985年。[39] Hung Q. Ngo、Ely Porat、Christopher Re 和 Atri Rudra。最坏情况最优连接算法。《ACM会刊》（JACM），65(3)：16:1–16:40，2018年。[40] Hung Q. Ngo、Christopher Re 和 Atri Rudra。倾斜再度来袭：连接算法理论的新进展。《SIGMOD Record》，42(4)：5–16，2013年。[41] Dan Olteanu 和 J akub Zavodny。查询结果因子化表示的大小界。《ACM 数据库系统汇刊》（TODS），40(1)：2:1–2:44，2015年。[42] Saladi Rahul。正交点包含查询和正交细分中点定位的改进界，适用于 $\mathbb{R}^3$。载于《ACM-SIAM离散算法年会》（SODA）论文集，第200–211页，2015年。

[43] Nicole Schweikardt、Luc Segoufin 和 Alexandre Vigny。在处处稀疏图上枚举一阶逻辑查询。载于《ACM数据库系统原理研讨会》（PODS）论文集，第151–163页，2018年。[44] Luc Segoufin。合取查询的常数延迟枚举。《SIGMOD Record》，44(1)：10–17，2015年。[45] Luc Segoufin 和 Alexandre Vigny。在具有局部有界扩张的数据库上实现一阶逻辑查询的常数延迟枚举。载于《国际数据库理论会议》（ICDT）论文集，第68卷，第20:1–20:16页，2017年。[46] 陶宇飞。独立查询采样的算法技术。载于《ACM数据库系统原理研讨会》（PODS）论文集，2022年。[47] 陶宇飞和易科。更新条件下的交集连接。《计算机与系统科学杂志》（JCSS），124：41–64，2022年。[48] Todd L. Veldhuizen。Triejoin：一种简单的最坏情况最优连接算法。载于《国际数据库理论会议》（ICDT）论文集，第96–106页，2014年。[49] Duncan J. Watts 和 Steven H. Strogatz。“ 小世界 ” 网络的集体动力学。《自然》，393：440–442，1998 年。[50] Virginia Vassilevska Williams 和 Ryan Williams。查找、最小化和计数加权子图。《SIAM 计算杂志》，42(3)：831–854，2013年。