Polynomial Algorithms for Graph Isomorphism in Directed and Undirected Graphs

Author: [Liang] [Chen]

1 Abstract

In this paper, only two graphs with identical initial degree sequences are compared, as graphs with different degree sequences can be easily distinguished through simple traversal, rendering further discussion unnecessary.

The core idea of this paper is reverse thinking: if two graphs are identical, they can be constructed step-by-step in the same manner, and they can also be deconstructed step-by-step in the same manner, with each deconstruction step producing identical intermediate results.

2 Keywords

2.1 Degree Sequence:

After deleting a vertex or during initialization, vertices are sorted in descending order based on their degrees. For undirected graphs, the sorting process ends here.

For directed graphs, additional steps are required. Vertices with the same total degree (sum of in-degree and out-degree) are further grouped and sorted. In this paper, if multiple vertices share the same total degree, they are additionally sorted in descending order based on their in-degrees. Vertices with larger total degrees always precede those with smaller total degrees, and only when the total degrees are identical are the vertices further sorted by their in-degrees.

2.2 Adjacent Vertices:

Two vertices sharing an edge are referred to as adjacent vertices.

2.3 Temporary Degree Sequence:

Deleting a vertex affects its adjacent vertices, necessitating a small-scale re-sorting of these affected vertices, following the same rules as in Section 2.1. However, the lifespan of this temporary sequence lasts only until the global vertex sorting forms a new degree sequence.

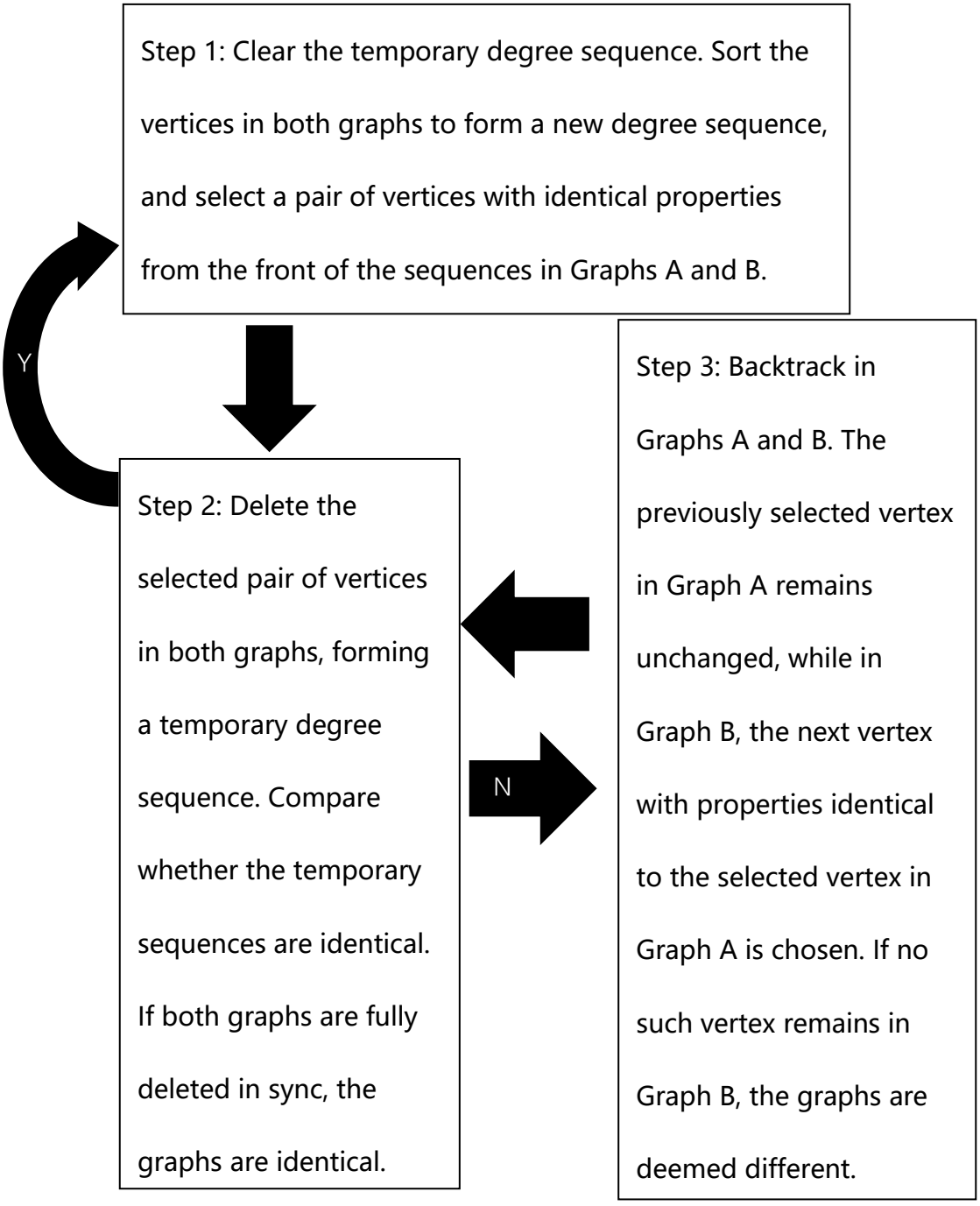
2.4 Graph A and Graph B:

Before determining graph isomorphism, the two graphs are labeled as A and B. During vertex deletion, Graph A selects a fixed vertex, while Graph B searches for a vertex with identical properties to the one selected in Graph A. This process continues until a matching vertex is found for deletion. Simply put, Graph A serves as the reference graph, and Graph B is the backtracking graph.

2.5 Vertex Properties:

In undirected graphs, this refers to the degree of a vertex. In directed graphs, it refers to the count of in-degrees and out-degrees of a vertex.

3 Algorithm Description



4 Time Complexity Analysis

Let n be the number of vertices in the graph.

Step 1:T(n)=$2nlg(n)$=$O(n\lg(n))$

Step 2:

Undirected Graphs:T(n)=$2nlg(n) + n + 2n^2 = O(n^2)$

Directed Graphs: T(n)=$2nlg(n) + n + 4n^2 = O(n^2)$

Step 3:T(n)=$1 + 1 = O(1)$

Overall:T(n)=$nlg(n) \times n^2 \times 1 = O(n^3 \lg(n))$

5 Proof

The algorithm ensures that after each deletion, the resulting degree sequence and temporary degree sequence remain identical, and the relationships between the vertices and other vertices are preserved. If Graphs A and B are different but a pair of deleted vertices share identical properties and produce identical temporary degree sequences, this does not affect the final result—it merely indicates that the graphs' differences lie elsewhere and that their local structures are identical. If Graphs A and B are identical but the deleted vertices do not correspond in the overall structure, this also does not affect the result. For example, suppose vertices $a_1a_2b_1b_2$share identical properties, with a_1 corresponding to b_1 and a_2 to b_2, but a_1 and b_2 or a_2 and b_1 do not correspond. If deleting a_1 and $b_2$produces the same temporary degree sequence, the consequences can be offset by subsequently deleting a_2 and b_1, ensuring no impact on the final result.

6 Appendix (Original Chinese Text)

一、摘要

在本文中只对初始度数队列完全一样的两张图进行比较，因为如果度数队列不一样可以通过简单遍历得出答案，不具备讨论价值。

本文的核心思想是逆向思维，如果两个图一样，即可以通过一样的步骤逐步搭建，也可以通过一样的步骤逐步拆解，并且每拆解一步产生的结果也完全一样。

二、关键词

（一）度数队列：

每次删除完一个顶点后和初始化都会将顶点按照度数从大到小排列，无向图的排序到此就结束了。

如果是有向图还有一些额外操作，还要对总度数一样的顶点进行分组排序，在本文中如果多个顶点总度数相同即入度和出度的和相同，还要分别按照入度从大到小再排一次顺序。总度数大的一定在总度数小的前面，只有总度数一样才会在按照入的度大小细分排序。

（二）顶点的相邻顶点：

两个顶点公用一条边，则称这两个顶点为相邻顶点。

（三）临时度数队列：

每次删除完一个顶点后会对其相邻顶点产生影响，需要对产生影响的这些顶点进行单独小规模的排序，排序规则与 2.1 相同。但此临时队列的生命周期只持续到全局顶点排序形成新的度数队列之前。

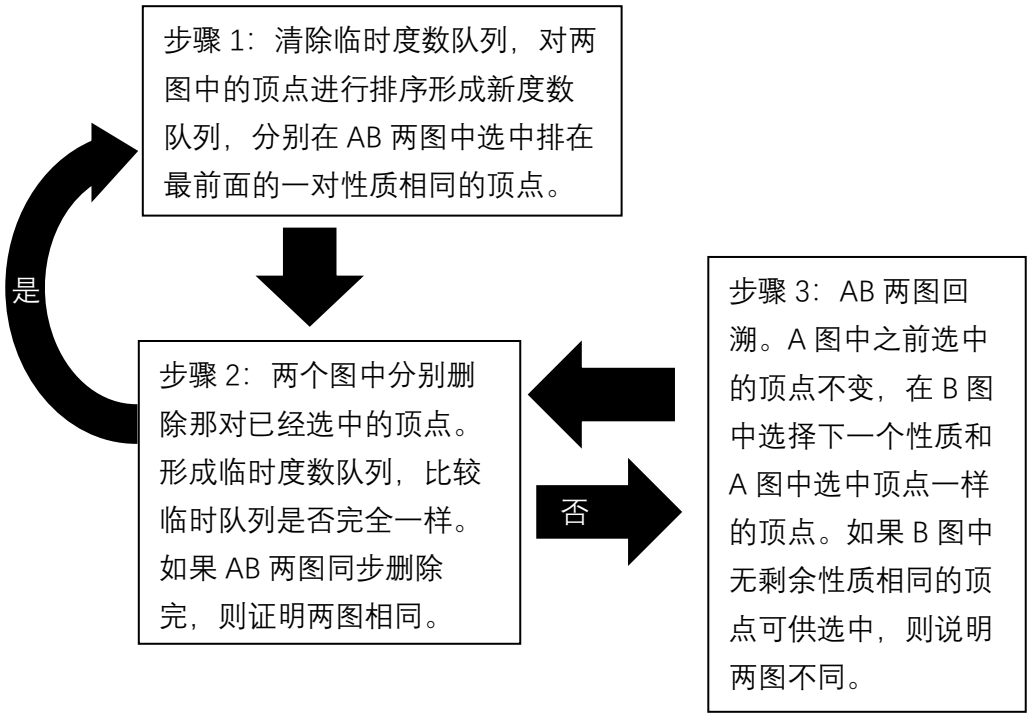
（四）A 图和 B 图：

在判断同构图之前为两个图设置代号，分别是 A 和 B。在删除顶点过程中 A 图选定一个顶点不变，会遍历 B 图找到和 A 图选中顶点性质一样的顶点，直至删除到满足条件的顶点。可以简单理解为，A 图是基准图，B 图是回溯图。

（五）顶点的性质：

无向图中某顶点的度数，有向图中某顶点的入度的数量和出度的数量。

三、算法描述



四、时间复杂度分析

n 为图中顶点数量。

步骤 1：T(n)=$2nlg(n)$=$O(n\lg(n))$

步骤 2：

无向图：T(n)=$2nlg(n)+n+2n^2=O(n^2)$

有向图：T(n)=$2nlg(n)+n+4n^2=O(n^2)$

步骤 3：T(n)=$1+1=O(1)$

T(n)=$nlg(n)\times n^2\times 1=O(n^3\lg(n))$

五、证明

能保证每次删除后产生的度数队列和临时度数队列一样，两个顶点和其他顶点之间的关联也一样。如果 AB 两图不同，但删除的某对顶点性质及其产生的临时度数队列相同，也不影响最终结果，这只能证明两图的差异不在此处和两图局部结构相同。如果 AB 两图相同，但删除的两点在整体结构中并不是对应的，也不会影响结果，假设$a_1a_2b_1b_2$四个顶点性质相同，a_1b_1与a_2b_2一一对应并且a_1b_2与a_2b_1无法对应，四个顶点删除后产生的临时度数队列也相同，那么删除a_1b_2产生的后果可通过在后续删除a_2b_1来抵消，不会对最终结果产生影响。