УДК 004.62

АЛГОРИТМ ОПРЕДЕЛЕНИЯ ИЗОМОРФИЗМА XML-СХЕМ

А.П. Сергеев

ООО «Диалектика»,
03187, Киев, проспект Академика Глушкова, 2, корп. 6, комн. 214,
тел.: (050)442-04-15,
e-mail: sergeev@dialektika.com

Рассматривается алгоритм определения изоморфизма XML-схем. Демонстрируется его применение в целях оптимизации памяти, выделяемой для хранения XML-документов, путем исключения XML-документов, XML-схемы которых изоморфны канонической (нормализованной) XML-схеме. Получена оценка вычислительной сложности алгоритма.

It is considered algorithm of definition of isomorphism of XML-schemas. Its application for optimization of the memory allocated for storage of XML-documents, by an elimination of the XML-documents which XML-schemas are isomorphic to the canonical (normalized) XML-schema is shown. The estimation of calculating pf complication of such algorithm is gained.

Вступление

Учитывая большую распространенность XML для обработки и хранения данных, возникает необходимость в совершенствовании применяемых при этом процедур. Предлагаемый алгоритм определения изоморфизма XML-схем позволит минимизировать объем памяти, выделяемой в хранилище данных для хранения XML-данных. Эта задача решается путем исключения XML-документов, XML-схемы которых изоморфны канонической (нормализованной) XML-схеме.

В работе [1] отмечается, что структура произвольного XML-документа (его разметка) определяется с помощью различных языков схем, но на практике чаще всего используются Document Type Definitions (DTD), Relax-NG, Schematron и W3C XSD (XSD-схемы).

Традиционно XML-схемы выполняют следующие функции:

- поддержку словарей, включающих списки элементов и атрибутов;
- связывают типы данных (например, целочисленных, строковых и т. д.) со значениями, определенными в XML-документе;
- определяют области видимости элементов и атрибутов, например, в любой главе должно быть название, за которым следует один или несколько текстовых абзацев;
- применяются для формального описания XML-документов.

XML-схемы обычно включают следующую информацию:

- представление связей между элементами данных, которое аналогично связям внешних ключей между таблицами, заданными в реляционной базе данных;
- представление уникальных идентификаторов, которые аналогичны первичным ключам;
- спецификацию типов данных каждого отдельного элемента и атрибута, включенных в XML-документ.

Далее приводится образец канонической XML-схемы, в данном случае в качестве XML-схемы выступает XML-схема, определенная в [1] (листинг 1).

Листинг 1. Образец XML-схемы

```
<?xml version = "1.0"
encoding = "utf-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="процессор" type="процессор"/>
  <xs:complexType name="процессор”>
  <xs:sequence>
    <xs:element name="название" type="xs:string"/>
    <xs:element name="цена" type="xs:decimal"/>
    <xs:element name="наличие на складе" type="xs:boolean"/>
  </xs:sequence>
  </xs:complexType>
</xs:schema>
```

На основе структуры, определенной XML-схемой, создается XML-документ. Пример XML-документа, сгенерированного на основе XML-схемы из листинга 1, показан в листинге 2.

Листинг 2. XML-документ, сгенерированный на основе канонической XML-схемы

```
<?xml version = "1.0"
encoding = "utf-8"?>
<процессор
  <xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="процессор.xsd">
    <название>Pentium C2D E7500 2,93GHz</название>
   <цена>126,70</цена>
   <наличие на складе>да</наличие на складе>
</процессор>
```

XML-схема, приведенная в листинге 1, может порождать XML-документы, включающих теги <название>, <цена> и <наличие на складе>.

По XML-схеме конструируется дерево, представляющее собой граф без циклов и петель. Например, по XML-схеме, которая показана в листинге 1, строится дерево, состоящее из одной вершины и трех, соединенных с ней узлов (рис. 1).

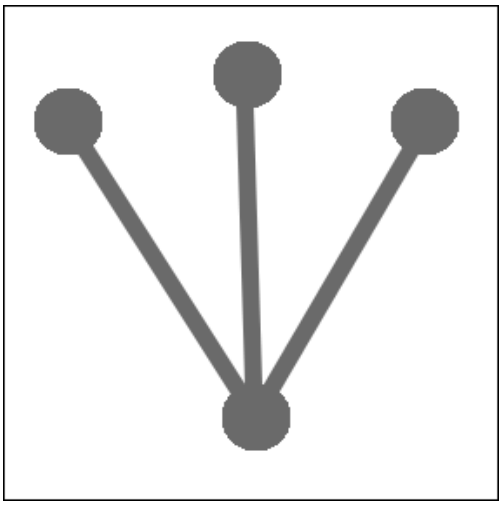

Рис. 1. Дерево, построенное по XML-схеме из листинга 1

Построение деревьев по XML-схемам выполняется естественным образом с помощью стандартных парсеров. Также возможно выполнение обратного действия — реконструкция XML-схемы по ее дереву. Деревья, построенные по XML-схемам, хранятся в виде *матриц смежности*. Матрица смежности представляет собой квадратную матрицу n×n (n – количество узлов дерева), элементы которой вычисляются по формулам (1):

Mij = 1, i,j=1…n, если узлы i и j соединены ребром; Mij = 0, i,j=1..n, если узлы i и j не соединены ребром (1). Для дерева, изображенного на рис. 1, строится следующая матрица смежности:

$$\begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

Далее утверждается (см. утверждение 3), что изоморфные XML-схемы порождают эквивалентные XML-документы. Следовательно путем исключения изоморфных XML-схем можно оставить эталонную (каноническую) XML-схему, которая будет порождать класс эквивалентных XML-документов.

Прежде, чем перейти к описанию алгоритма определения изоморфизма деревьев, напомню основные положения, имеющие отношение к алгоритму определения изоморфизма графов [2].

Алгоритм определения изоморфизма графов

Утверждение 1. Пусть V_1 – множество вершин графа G_1, а E_1 – множество его ребер. Пусть V_2 – множество вершин графа G_2, а E_2 – множество его ребер. Графы [2] G_1 и G_2 называются изоморфными, если существует взаимно однозначное соответствие $f(V_1)\rightarrow V_2$ такое, что (v,w) принадлежит V_1 тогда и только тогда, когда (f(v),f(w)) принадлежит V_2. Иными словами, существует соотношение между вершинами графа G_1 и вершинами графа G_2, которое сохраняет отношение смежности. Это отношение представляет собой количество ребер, входящих (или исходящих) из каждой вершины графа. То есть графы, между которыми установлено отношение изоморфизма, отличаются друг от друга лишь пометками вершин.

Утверждение 2. Под *инвариантом изоморфизма* понимаются характеристики графа, которые сохраняются без изменений при изоморфном отображении. Система инвариантов изоморфизма называется *полной*, если из равенства между собой всех инвариантов изоморфизма вытекает изоморфизм графов.

Простейшим инвариантом изоморфизма является количество вершин графа. Очевидно, что между графами с различным количеством вершин не может быть установлено взаимно однозначное отношение изоморфизма. В дальнейшем будут рассмотрены примеры других инвариантов изоморфизма.

В общем случае алгоритм определения изоморфизма графов выглядит следующим образом.

1. Определяется система инвариантов изоморфизма.
2. Выполняется сравнение инвариантов изоморфизма.
3. Если установлено неравенство хотя бы в одной из пар инвариантов изоморфизма, графы будут неизоморфны.
4. Если результаты проверки равенства инвариантов изоморфизма положительны и система инвариантов изоморфизма является полной, графы будут изоморфными.
5. Если система инвариантов изоморфизма является неполной, для установления изоморфизма графов используется алгоритм поиска с возвращением (back-tracking). Этот алгоритм рассматривается ниже.

Предположим, что в нашем распоряжении имеются два графа $G_1 = (V_1,E_1)$ и $G_2 = (V_2,E_2)$, для которых нужно установить факт изоморфизма. Пусть $V_1 = V_2 = \{1,2,...,n\}$ – множество вершин графов G_1 и G_2. Один из графов, например G_1, выбирается в качестве эталона для сравнения в процессе установления факта изоморфизма. Пусть $G_1(k)$ – подграф графа G_1, индуцированный вершинами $\{1,2,\ldots,k\}$, $0 \le k \le n$. $G_1(0)$ – пустой (нулевой) подграф, а $G_1(1)$ – подграф, который состоит из одной вершины и не имеет ребер. Очевидно, что пустой подграф $G_1(0)$ изоморфен пустому подграфу $G_2(0)$. Предположим, что на определенном этапе найден подграф G_2, который состоит из набора вершин S, являющегося подмножеством множества вершин V_2. Установлено, что этот подграф изоморфен подграфу G_1. Продолжим отношение изоморфизма на подграф $G_1(k+1)$ путем выбора вершины v из подмножества V_2/S. Если подобная вершина v найдена, зафиксируем соотношение $f_{k+1} \rightarrow v$ и продолжим отношение изоморфизма на подграф $G_1(k+2)$. Если же требуемая вершина v не найдена, возвращаемся к подграфу $G_1(k)$ и выбираем другую вершину среди k вершин из множества V_1. Описанный процесс продолжается до тех пор, пока k не станет равным n и, соответственно, не будет установлен изоморфизм между подграфом $G_1(n)=G_1$ и G_2. Если же на определенном шаге процесса перебора обнаруживается, что подграфы неизоморфны, это означает, что графы G_1 и G_2 также будут неизоморфными.

В худшем случае (при осуществлении перебора всех подграфов G(k), где k=1...n), вычислительная сложность этого алгоритма равна O(n!) операций. На практике же вычислительная сложность будет существенно меньше. Она будет полиномиальной, если факт изоморфизма устанавливается на основе равенства инвариантов изоморфизма.

Алгоритм определения изоморфизма XML-схем

На основе общего алгоритма определения изоморфизма графов строится частный алгоритм определения изоморфизма деревьев. Сначала сформулируем ряд вспомогательных утверждений.

Для XML-схем, справедливо утверждение.

Утверждение 3. Если XML-схемы изоморфны (в смысле изоморфизма соответствующих им деревьев), порождаемые ими XML-документы будут эквивалентны.

Эквивалентность XML-документов понимается как единообразие структуры (одинаковое количество тегов разметки, узлов / дочерних узлов, а также связей между ними).

Из вышесказанного вытекает утверждение.

Утверждение 4. Проверка изоморфизма XML-схем сводится к проверке изоморфизма соответствующих этим схемам деревьев.

Из вышесказанного следует, что проверка изоморфизма XML-схем сводится к проверке изоморфизма деревьев. В этом случае вычислительная сложность алгоритма определения изоморфизма существенно уменьшается. Более точная оценка будет приведена в заключительном разделе работы.

Поскольку дерево является частным случаем графа (граф без циклов и петель), по отношению к ним справедливо утверждение 1. Перефразируя это утверждение можно сказать, что два дерева будут изоморфны, если они имеют одинаковую структуру. Внешний вид изоморфных деревьев может отличаться. На рис. 2 показаны два изоморфных дерева, а на рис. 3 – два неизоморфных дерева.

Естественно, что по отношению к деревьям может применяться описанный в предыдущем разделе алгоритм определения изоморфизма графов. Для проверяемых деревьев вычисляются инварианты изоморфизма, выполняется их сравнение и в случае равенства выполняется дальнейшая проверка на изоморфизм с помощью упрощенного варианта алгоритма поиска с возвращением (back-tracking).

Сначала рассматривается ограниченный класс деревьев, которые имеют общий корень. Подобные деревья порождаются XML-схемами, которые имеют общий корневой элемент, например, <процессор> (см. листинг 2).

Для установки изоморфизма деревьев, имеющих общий корень, можно воспользоваться *алгоритмом сравнения*. Этот алгоритм идентифицирует структуру самого дерева, не учитывая пометки вершин, т.е. изоморфные деревья считаются идентичными.

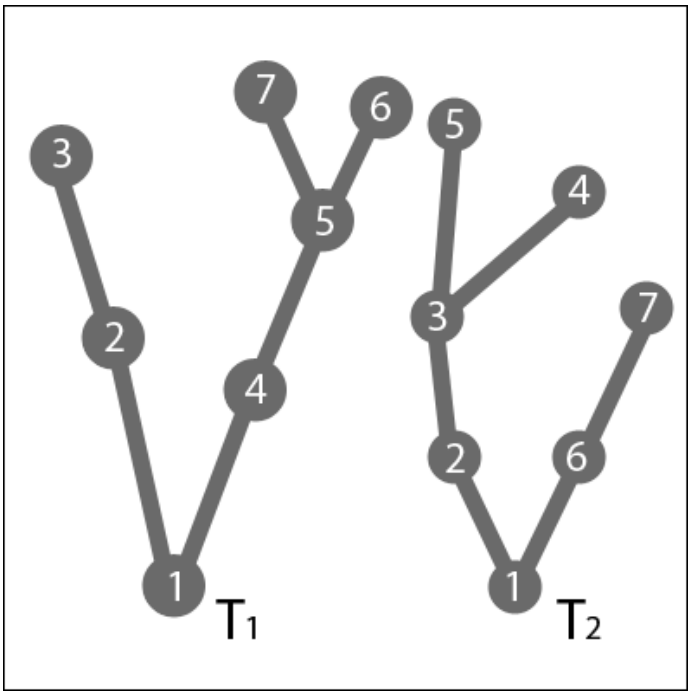

Рис. 2. Изоморфные деревья

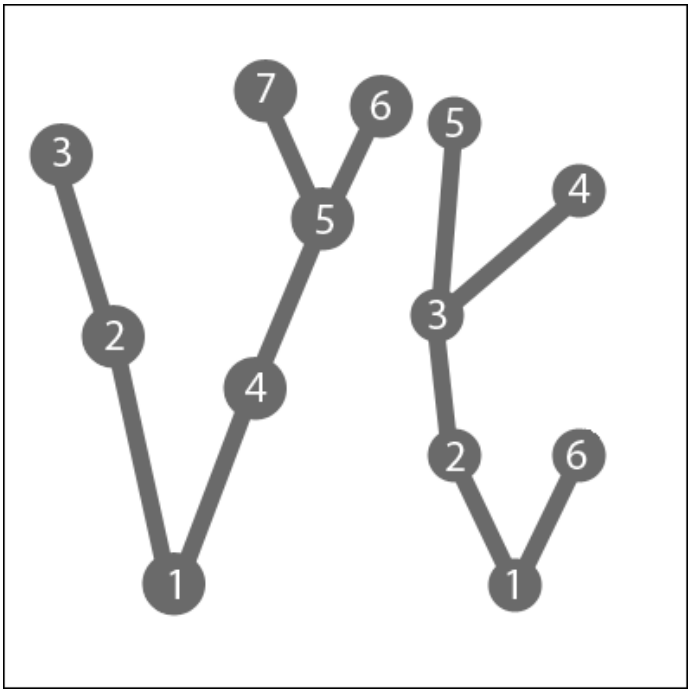

Рис. 3. Неизоморфные деревья

На рис. 2 показаны два изоморфных дерева T_1 и T_2. Следует обратить внимание, что эти деревья имеют одинаковую структуру (дерево в правой части рисунка является зеркальным отражением дерева, изображенного в левой части рисунка). Между вершинами деревьев T_1 и T_2 существует следующее взаимно однозначное соответствие (2):

$$
\begin{gathered}
1 \leftrightarrow 1 \\
2 \leftrightarrow 6 \\
3 \leftrightarrow 7 \\
4 \leftrightarrow 2 \\
5 \leftrightarrow 3 \\
6 \leftrightarrow 5 \\
7 \leftrightarrow 4
\end{gathered}
$$

Каждой вершине дерева ставится в соответствие числовая последовательность вида $a,b,(x_1,x_2,...x_n)$, где a – уровень вершины дерева, отсчитывая от его корня; b – количество потомков для данной вершины (длина максимальной линии, образованной потомками); $(x_1, x_2,\ldots,x_n)$ – ряд, содержащий количество потомков для дочерних вершин данной вершины. Назовем эту последовательность *характеристическим вектором*. Далее будет показано что именно характеристический вектор является инвариантом изоморфизма для деревьев.

Построим характеристический вектор для рассматриваемого примера. Для вершин дерева T_1 этот вектор будет иметь следующий вид (3):

- вершина 1 – 0, 3, (1,2);
- вершина 2 – 1, 1, (0);
- вершина 3 – 2, 0, (0);
- вершина 4 – 1, 2, (1, 1);
- вершина 5 – 2, 1, (0, 0);
- вершина 6 – 3, 0, (0);
- вершина 7 – 3, 0, (0).

А теперь построим характеристический вектор для вершин дерева T_2 (4):

- вершина 1 – 0, 3, (2,1);
- вершина 2 – 1, 2, (1,1);
- вершина 3 – 2, 1, (0,0);
- вершина 4 – 3, 0, (0);
- вершина 5 – 3, 0, (0);
- вершина 6 – 1, 1, (0);
- вершина 7 – 2, 0, (0).

Результатом построения характеристических векторов для деревьев T_1 и T_2 явились два массива записей. Теперь в соответствии с утверждением 1 можно заключить, что если между элементами этих двух массивов возможно установление взаимно однозначного соответствия, то заданные этими массивами деревья (частный случай графа) будут изоморфными. И действительно, на основе анализа характеристических векторов (3) и (4) нетрудно заметить, что имеет место однозначное соответствие (изоморфизм) между вершинами деревьев T_1 и T_2 (1). С точностью до перестановки элементов характеристические векторы (3) и (4) равны между собой, и поскольку они соответствуют деревьям, имеющим общий корень, из равенства векторов вытекает факт изоморфизма деревьев (и соответствующих им XML-схем). В данном случае характеристические векторы представляют собой полную систему инвариантов изоморфизма, т.е. на основе их равенства между собой можно сделать вывод об изоморфизме деревьев.

Алгоритм сравнения безупречно работает в том случае, когда сравниваемые деревья имеют один и тот же корень. Учитывая это возникает вопрос, как быть в более общем случае, когда деревья имеют разные корневые вершины, как показано на рис. 4.

Если требуется установить изоморфизм для деревьев с различными корневыми вершинами, то в дополнение к вышеописанному алгоритму сравнения применяется *алгоритм перенумерации вершин*. Далее приводится краткий обзор этого алгоритма.

1. Для первого дерева (рис. 4, слева) следует найти *концевую вершину* (вершина, у которой отсутствуют потомки), затем сделать ее корневой. Другими словами, происходит «захват» дерева за одну из вершин и его «вытягивание» вниз. Полученное в результате выполнение этой операции дерево принимается за эталонное (рис. 5, справа).

2. Выбираются все концевые вершины для второго дерева, которые поочередно «превращаются» в корневые. Затем производится сравнение преобразованного дерева с эталонным. При этом применяется описанный ранее алгоритм сравнения.

3. Если в результате применения алгоритма сравнения было хоть раз достигнуто равенство, это означает наличие изоморфизма деревьев.

Согласно простым оценкам вычислительная сложность описанного алгоритма проверки изоморфизма деревьев является полиномиальной и в большинстве случаев оценивается величиной $O(n^2)$, где n – количество вершин в любом из сравниваемых деревьев. Если проверяется изоморфизм деревьев, имеющих общий корень, вычислительная сложность оценивается как $O(n)$.

На рис. 6 показано схематическое изображение алгоритма определения изоморфизма деревьев.

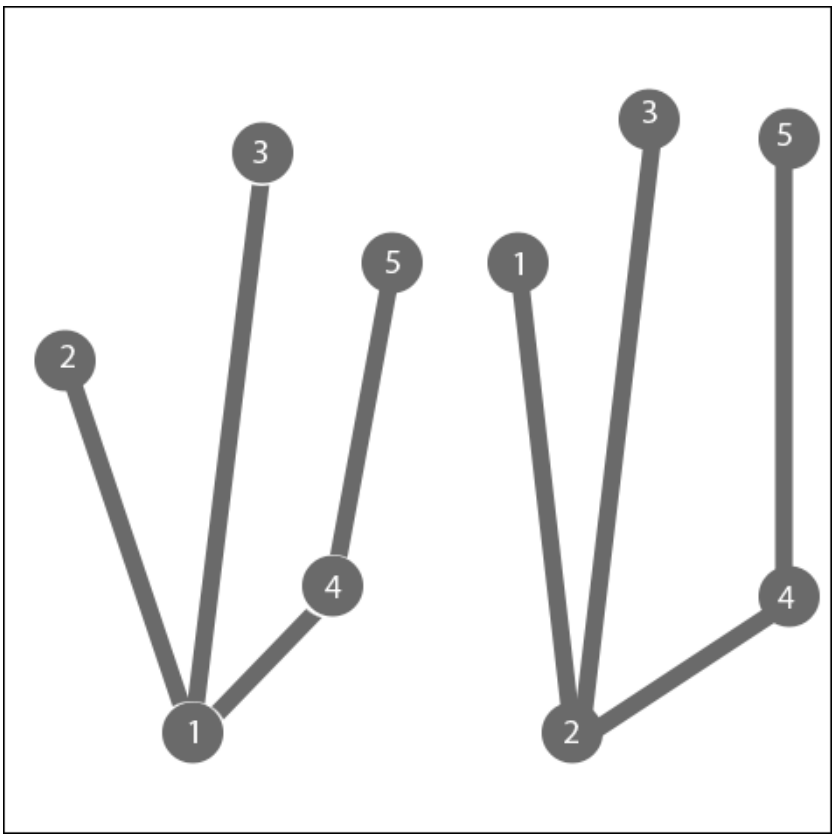

Рис. 4. Изоморфные деревья, имеющие разные корневые вершины

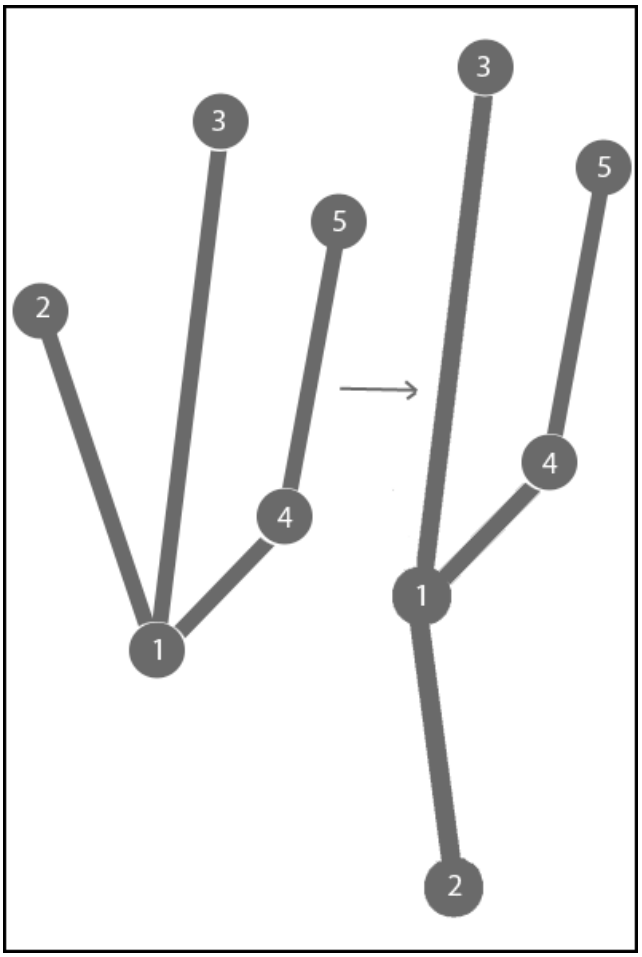

Рис. 5. «Вытягивание» дерева

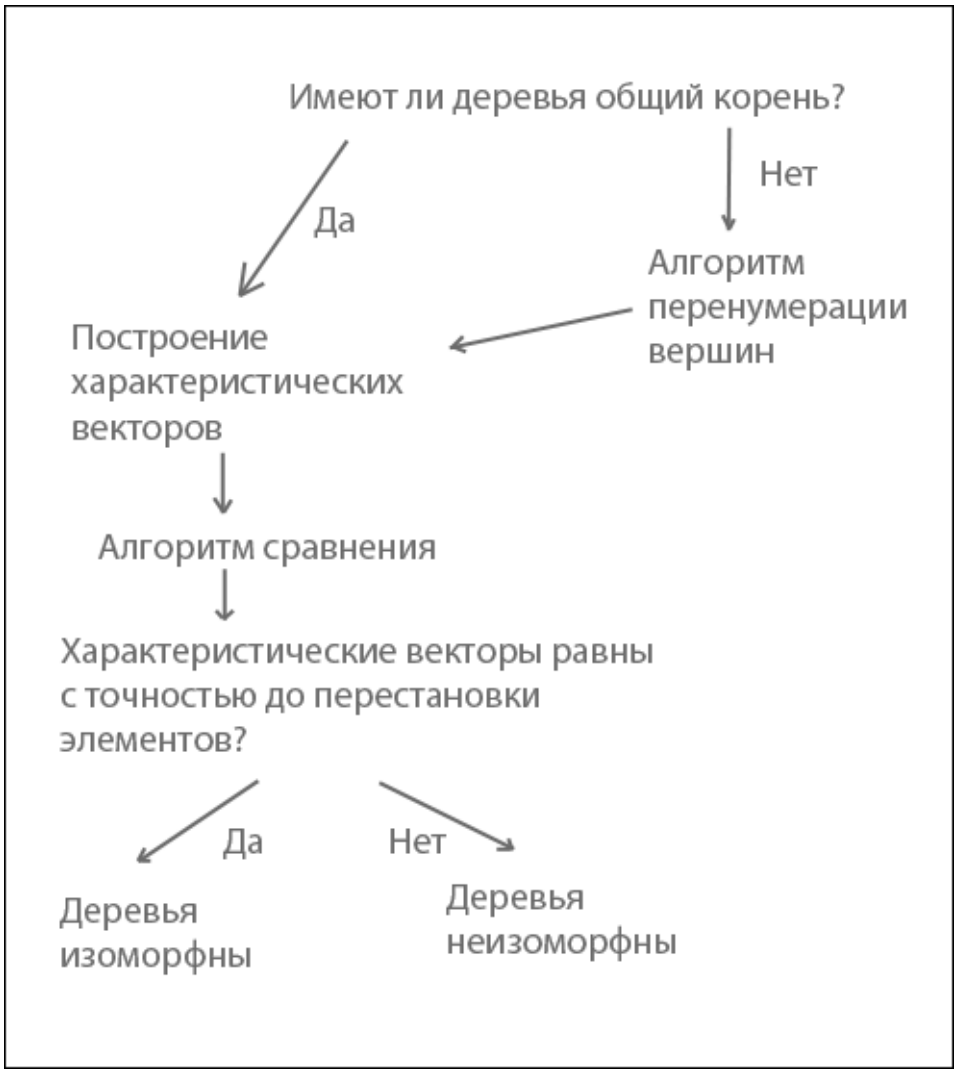

Рис. 6. Схематическое изображение алгоритма проверки изоморфизма деревьев

Выводы

В работе рассмотрен алгоритм определения изоморфизма XML-схем, сводящийся к определению изоморфизма соответствующих им деревьев. Благодаря применению этого алгоритма из хранилища данных исключаются XML-документы, XML-схемы которых изоморфны каноническим XML-схемам. В результате существенно уменьшается объем памяти хранилища данных, выделяемой для хранения XML-документов.

1. *W3C/XML* Technology/Schema [http://www.w3.org/standards/xml/schema].
2. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – С. 202–204.
3. *Сергеев А.П.* Быстрый алгоритм определения изоморфизма графов // УСиМ. – 1996. – № 4–5 – С. 35–38.