УДК 004.421.2

РЕКУРСИВНЫЙ АЛГОРИТМ СРАВНЕНИЯ ДЕРЕВЬЕВ

А.В. ЛЫЧКОВСКИЙ

Белорусский государственный университет информатики и радиоэлектроники
ул. Л.Беды, 2(б), г. Минск, 220040, Республика Беларусь
mkdnmail@gmail.com

Задача сравнения моделей данных широко исследуется в информатике. Большой практический интерес имеют задачи связанные с данными представленными деревьями. Одной из задач является сравнение деревьев. Данная работа дает представление о подходе к сравнению и описывает рекурсивный алгоритм решающий такую задачу.

Ключевые слова: дерево, сравнение деревьев, рекурсивный алгоритм.

Деревья являются одной из хорошо изученных структур данных в информатике. На практике задача сравнения деревьев возникает в таких областях, как вычислительная биология, структурированные базы данных текста, анализ изображений, оптимизация компиляторов.

Дерево – узел (называемый корневым) соединенный с упорядоченной последовательностью несвязных деревьев[1]. Такая последовательность называется **лесом**. Под обозначением $l(A_1 \circ \ldots \circ A_n)$ будем понимать дерево, составленное из узла l, соединенного с последовательностью деревьев $A_1 \circ \ldots \circ A_n$. Это определение предполагает, что деревья являются упорядоченными, а вершины помеченными. Дерево называется помеченным, если определен порядок среди узлов дерева. Для вычисления редакционного расстояния на дереве определяются следующие операции:

- **удаление** узла и соединение его детей к предку узла с сохранением порядка;
- **вставка** узла между существующими узлом и подпоследовательностью упорядоченных детей узла;
- **переименование** метки узла.

Пускай F и G два леса. Редакционное расстояние между F и G, обозначаемое $d(F,G)$, минимальная стоимость операций редактирования необходимых для преобразования F в G. Используя рекуррентный подход, редакционное расстояние может быть представлено следующими формулами:

$$d(l(F)\circ T, l'(F')\circ T') = min\begin{cases} c_d(l) + d(F\circ T, l'(F')\circ T') \\ c_i(l') + d(l(F)\circ T, F'\circ T') \\ d\big(l(F), l'(F')\big) + d(T,T') \end{cases} \quad (1)$$

или

$$d\big(T\circ l(F), T'\circ l'(F')\big) = min\begin{cases} c_d(l) + d(T\circ F, T'\circ l'(F')) \\ c_i(l') + d(T\circ l(F), T'\circ F') \\ d\big(l(F), l'(F')\big) + d(T,T') \end{cases} \quad (2)$$

Разбиение согласно (1) называется левым, согласно (2) правым. Разные разбиения ведут к разному количеству рекурсивных вызовов.

Прямые попытки реализации рекурсивного алгоритма имеют экспоненциальную сложность. Современные алгоритмы динамического программирования имеют полиномиальное время выполнения. Решение подзадач сохраняется и используется повторно. Они используют стратегии разбиения для выбора левого или правого на каждом шаге.

Выбор между левым и правым рекурсивным решением может быть определен через **декомпозиции путей**[2]. Они используют корень-лист пути в дереве, для декомпозиции его на поддеревья и подлеса. Левый путь, правый путь и тяжелый путь рекурсивно соединяют предка с его крайним левым ребенком, крайним правым ребенком, с ребенком который определяют наибольшее поддерево соответственно. **Релевантные поддеревья** дерева для некоторого корень-лист пути – все поддеревья, которые получаются удалением пути из дерева. **Релевантные подлеса** дерева – это само дерево и все его поддеревья полученные удалением узлов в следующем порядке: удалить корень и остановиться, если узлов не осталось; удалить крайний левый корневой узел в результирующем дереве пока крайний левый корневой узел находится на корень-лист пути; удалить крайний корневой узел пока крайний правый корневой узел находится на корень-лист пути; рекурсивно повторять процедуру для результирующего поддерева.

Для каждой задачи состоящей из двух релевантных поддеревьев мы должны выбрать путь для декомпозиции. Путь может быть выбран как в левом так и в правом дереве. Набор путей который мы выбираем определяет **стратегию путей**.

Концепция декомпозиции путей использована для вычисления редакционного расстояния в оптимальном размере памяти $O(mn)$ [3]. **Функция одного пути** вычисляет редакционное расстояние для дерева из двух релевантных поддеревьев в соответствии с выбранным путем. Вычисленные расстояния сохраняются и повторно используются.

На основании описанного выше общий алгоритм может быть сформулирован. Для двух деревьев F и G расстояние между ними может быть вычислено в два шага. Во первых надо выбрать путь в одном из них, например F, и вычислить расстояние между релевантными поддеревьями F и деревом G. Эти расстояния буду повторно использованы на следующем шаге. Во вторых, нужно вычислить расстояние между F и G от низа к верху, вычисляя расстояния между релевантными подлесами F и всеми соответствующими подлесами G. На этом шаге мы не вычисляем заново уже полученные расстояния с первого этапа.

Общий алгоритм редакционного расстояния деревьев использует стратегию путей для его вычисления. Алгоритм состоит из следующих шагов:

- для данной пары деревьев F, G находит путь используя стратегию путей;
- если путь в F выполняет следующие шаги, иначе выполняет алгоритм для G, F: выполняет алгоритм для каждого релевантного поддерева F' в F и дерева G, вычисляет функцию одного пути для F, G в соответствии с типом пути.

Список литературы

1. Computer Science Laboratory of Lille: Analysis of tree edit distance algorithms. [Электронный ресурс]. – Режим доступа: http://www.lifl.fr/~touzet/Publications/cpm03.pdf. – Дата доступа: 10.12.2013.

2. Tree Edit Distance: General Tree Edit Distance Algorithm. [Электронный ресурс]. – Режим доступа: http://www.inf.unibz.it/dis/projects/tree-edit-distance/tree-edit-distance.php. – Дата доступа: 10.12.2013.

3. *K. Zhang, D. Shasha.* Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. 1989.