МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

III Всероссийской молодежной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 22–23 мая 2015 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательский Дом Томского государственного университета
2015

3. *Matrosova A., Ostanin S., Kirienko I.* Increasing Manufacturing Yield Using Partially Programmable Circuits with CLB implementation of Incompletely Specified Boolean Function of the Corresponding Sub-circuit // Proc. IEEE Intl. Symp. on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2015). – 2015.

4. *Lavagno L., McGeer P., Saldanha A., Sangiovanni-Vincentelli A.* Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool// Proc. 32nd Design Automation Conf. – 1995. – P. 254–260.

5. *Shiple T., Hojati R., Sangiovanni-Vincentelli A., Bryaton R.* Heuristic Minimization of BDDs Using Don't Cares// Proc. Design Automation Conf. – 1994. – P. 225–231.

6. *Chang S., Cheng D., Marek-Sadowska M.* Minimizing ROBDD Size of Incompletely Specified Multiple Output Functions // Proc. European Design and Test Conf. – 1994. – P. 620–624.

7. *Oliveira A., Carloni L., Villa T., Sangiovanni-Vincentelli A.* Exact minimization of Binary Decision Diagrams Using Implicit Techniques// IEEE Trans. Computers. – 1998. – V. 47. – No. 11. – P. 1282–1296.

ИССЛЕДОВАНИЕ ДЕКАРТОВОГО ДЕРЕВА И НАПИСАНИЕ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО ЕГО ПОСТРОЕНИЮ

Т. С. Овчинникова, В. А. Сибирякова
Томский государственный университет
E-mail: tanyalastochkina@mail.ru, val349@mail.ru

Введение

В настоящее время роль информационных технологий очень важна. Поэтому возникает необходимость создания обучающих систем. Но в нынешнем разнообразии систем нет эффективных программ для изучения деревьев поиска. А эти структуры данных очень важны для хранения и быстрого поиска информации в базах данных и других поисковых системах.

Поэтому целью данной работы было создание обучающей системы для изучения декартового дерева поиска. Данная система может быть использована не только студентами, но и преподавателями для проверки контрольных работ, а также любыми желающими с целью изучить декартовы деревья и, в дальнейшем, применять их. Большой плюс обучающей программы в том, что желающий может работать с ней в любой день, в любое время суток и продолжительность обучения не ограничена.

1. Постановка задачи

Требуется изучить свойства декартового дерева и написать обучающую программу. Обучающая программа должна:

1) демонстрировать алгоритм построения дерева;
2) проверять знания пользователя по построению дерева.

2. Определение декартового дерева

Декартово дерево – это двоичное дерево поиска, в узлах которого хранятся:

1) ссылки на правое и левое поддерево;
2) ключи x и y, которые являются двоичным деревом поиска по ключу x и двоичной кучей по ключу y; а именно, для любого узла дерева n:
   a) ключи x узлов правого (левого) поддерева больше (меньше либо равны) ключа x узла n;
   b) ключи y узлов правого и левого детей больше либо равны ключу y узла n.

В англоязычной литературе очень популярно название treap, которое наглядно показывает суть структуры: tree + heap. В русскоязычной же иногда можно встретить составленные по такому же принципу: дерамида (дерево + пирамида) или дуча (дерево + куча).

Впервые дерамиды были предложены в статье Seidel, Raimund; Aragon, Cecilia R. (1996), «Randomized Search Trees» [1].

3. Достоинства и недостатки декартового дерева

Декартово дерево не является самобалансирующимся в обычном смысле (сбалансированность означает, что для каждой вершины высота её двух поддеревьев различается не более чем на единицу), и применяют его по таким причинам:

1) просто программируется относительно самобалансирующихся деревьев поиска, например сбалансированных, красно-черных;[2]
2) хорошо ведёт себя «в среднем», если приоритеты у раздать случайно (такое дерево называется декартово дерево по неявному ключу);
3) типичная для сортирующего дерева операция «расчленить по ключу x_0 на „меньше x_0“ и „больше x_0“» работает за $O(h)$, где h – высота поддерева с корнем x_0.

Недостатки декартового дерева:

1) большие накладные расходы на хранение: вместе с каждым элементом хранятся два-три указателя и случайный ключ y;
2) скорость доступа $O(n)$ в худшем случае (где n – количество вершин в дереве), хотя и при критических объемах данных это очень маловероятно.

4. Почему дерево называется декартовым?

Попробуем нарисовать его на координатной сетке. Возьмем набор пар «ключ-приоритет» и расставим на координатной сетке соответствующие точки (x,y). А потом соединим соответствующие вершины линиями, образуя дерево. Таким образом, декартово дерево отлично укладывается на плоскости благодаря своим ограничениям, а два его основных параметра — ключ и приоритет — в некотором смысле, координаты. Результат построения показан на рис. 1: слева в стандартной нотации дерева, справа — на декартовой плоскости.

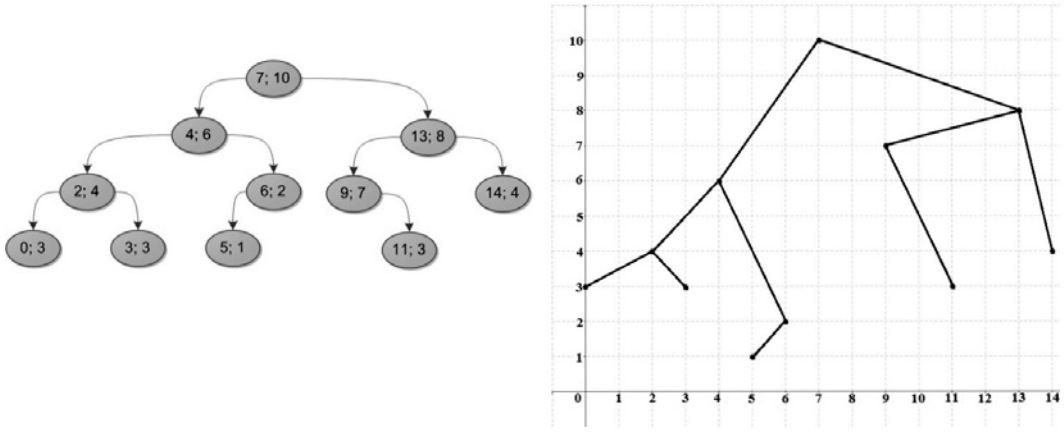

Рис.1. Почему дерево называется декартовым?

5. Применение

Для построения массивов применяют декартовы деревья по неявному ключу для того, чтобы с легкостью в дальнейшем выполнять следующие операции с массивом:

1) слить два массива в один;
2) разделить массив на два;
3) вставить элемент внутрь массива на требуемую позицию за $O(\log_2 N)$, а не за $O(N)$, как без помощи дерева поиска;
4) удалить из массива элемент, стоящий на данной позиции;

5) за $O(\log_2 N)$ можно также выполнять все те же множественные запросы на подотрезках массива (сумма/максимум/минимум/наличие или количество меток и т.д.);
6) за $O(\log_2 N)$ на подотрезках массива выполнять следующие операции: прибавление константы, покраску, установку в единое значение и т.д.;
7) переворот подотрезка, то есть перестановка его элементов в обратном порядке;
8) циклический сдвиг.

6. Основные операции с деревом

Для операции вставки и удаления узла из дерева, которые будут описаны далее, понадобятся операция сложения и деления. Операция сложения принимает на вход два декартовых дерева L и R. На выходе результат их сложения. Корнем будущего дерева станет узел с наибольшим приоритетом. Сравним приоритеты корней двух исходных деревьев; пусть для однозначности приоритет у левого корня больше, а ключ в нем равен x. Все дерево R окажется в правом поддереве нового корня. Точно так же левое поддерево старого корня *L.Left* имеет все ключи, меньшие x, и должно остаться левым поддеревом, а правое поддерево *L.Righ* должно оказаться справа. Рекурсивно вызываем операцию для *L.Right* и дерева R, и возвращенное ею дерево используем как новое правое поддерево. На рис. 2 пунктиром показано правое поддерево результирующего дерева после операции сложения и связь от нового корня к этому поддереву [3].

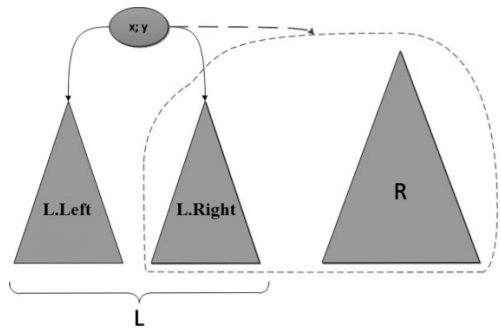

Рис. 2. Сложение деревьев

Теперь об операции деления. На вход ей поступает декартово дерево и некий ключ x_0. На выходе – два дерева, причем в одном из них (L) все элементы с ключами, меньшими x_0, а в другом (R) — с большими. Корень исходного дерева окажется в L, если его ключ меньше x_0, иначе – в R. Предположим, для однозначности, что ключ корня оказался меньше x_0. На рис. 3 изображен результат такого деления. Левое поддерево корня полностью сохранится без изменений, а вот правое уменьшится — из него придется убрать элементы с ключами, большими x_0, и вынести в дерево R, а остаток ключей сохранить как новое правое поддерево L. Возьмем правое поддерево и рекурсивно разрежем его по тому же ключу x_0 на два дерева L' и R'. После чего становится ясно, что L' станет новым правым поддеревом дерева L, а R' и есть непосредственно дерево R — оно состоит из тех и только тех элементов, которые больше x_0 [3].

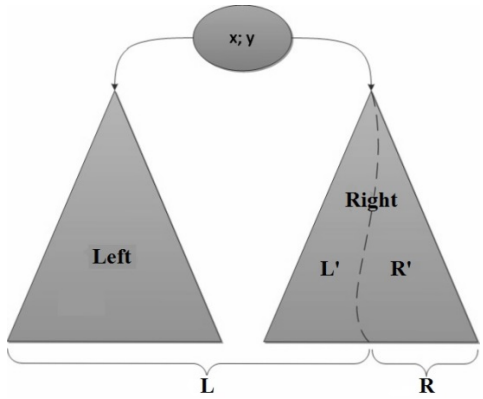

Рис. 3. Деление дерева

Теперь, когда мы знаем об операциях сложения и деления, с их помощью можем делать основные операции с деревьями поиска: добавление элемента в дерево и удаление его.

Для добавления ключа x в дерево необходимо выполнить следующие шаги (иллюстрация шагов на рис. 4):

1) разделить дерево по ключу x;
2) создать из данного ключа дерево M из единственной вершины (x,y), где y – только что сгенерированный случайный приоритет;
3) объединим по очереди L с M, то, что получится – с R.

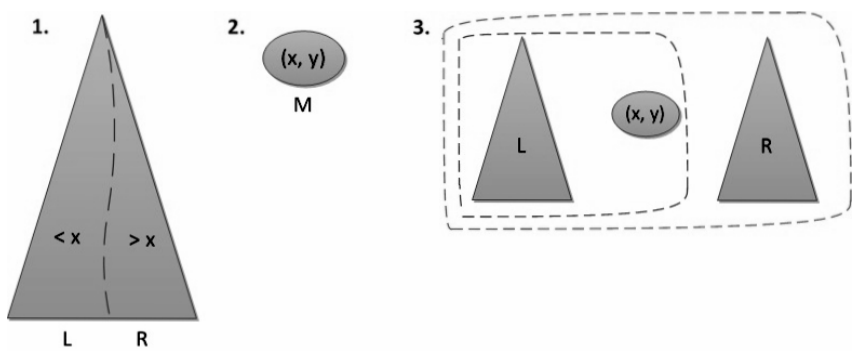

Рис. 4. Вставка элемента

При удалении ключа x из дерева необходимо выполнить следующие шаги (иллюстрация шагов на рис. 5):

1) разделить дерево по ключу x–1;
2) разделить правое поддерево по ключу x. В результате, после выполнения этого шага, левое поддерево правого исходного дерева и есть искомый элемент;
3) теперь просто объединим снова левое дерево с правым, без среднего, и дерамида осталась без ключа x.

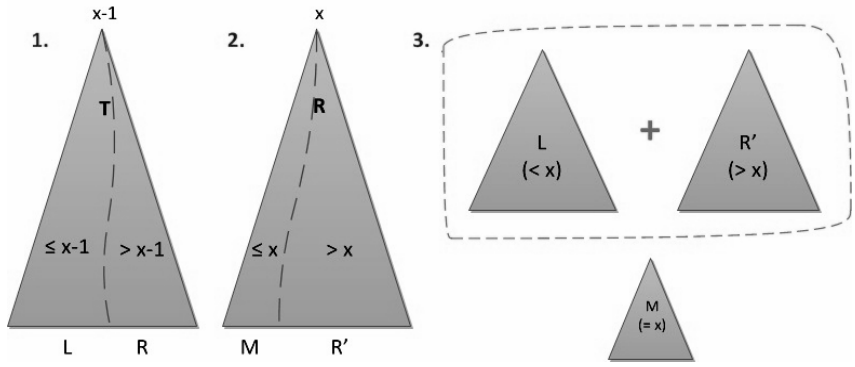

Рис. 5. Удаление вершины

7. Обучающая программа

Алгоритм работы. После запуска программы и выбора операции «Старт», программа считывает с документа лежащий в нем массив ключей и приоритетов, и строит по шагам (при последовательном выполнении операции «Вставить» осуществляется следующий шаг) декартово дерево.

На рис. 6 проиллюстрирован пример добавления узла с ключом 22 и приоритетом 11. Слева до, справа после добавления этого узла.

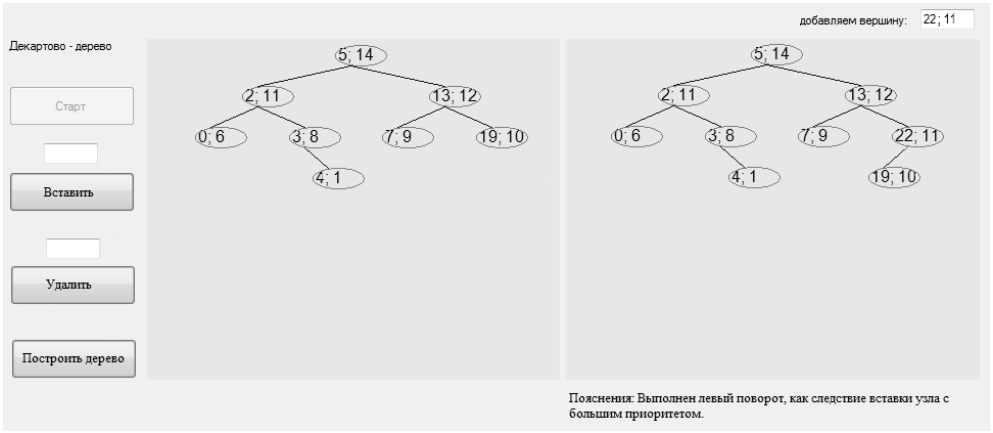

Рис. 6. Интерфейс программы. Режим обучения

В программе возможна вставка узла «вручную», для этого необходимо в специальное поле записать нужное число и выполнить действие «Вставить». Также пользователь может удалить любую вершину, чтобы посмотреть поведение декартового дерева. Для этого в специальное поле необходимо ввести это число и выбрать операцию «Удалить». Для построения дерева целиком из узлов, хранящихся в текстовом документе, необходимо сразу выбрать пункт «Построить дерево».

После изучения декартового дерева пользователь может перейти в контролирующий режим и проверить свои знания. На рис. 7 проиллюстрирован один из шагов режима контроля. Система запрашивает место для нового узла и, если необходим поворот, относительно какой вершины поворот и направление поворота. После окончания построения система анализирует введенный вариант построения дерева с правильным. В заключении пользователь видит сообщение о результатах и два дерева: свое и эталонное.

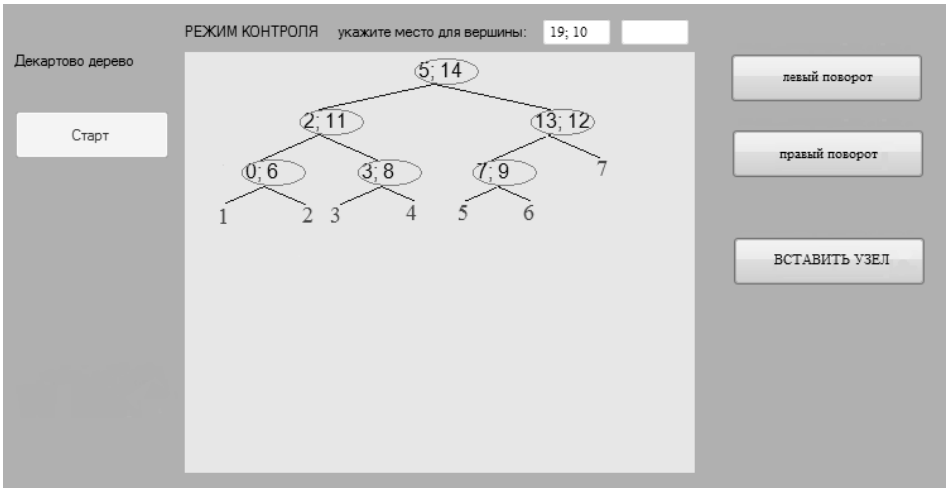

Рис. 7. Интерфейс программы. Режим контроля

Заключение

В ходе данной работы изучено декартового дерева поиска и создана обучающая программа, позволяющая изучить декартово дерево, а также проверить свои знания. Алгоритм построения дерева был запрограммирован на языке C++, а интерфейс в визуальной среде Visual Studio.

ЛИТЕРАТУРА

1. *Raimund Seidel* Randomized Search Trees: 1996. – 33 с.

2. Алгоритмы: построение и анализ [пер. с англ.] / Т. Кормен [и др.]. –2-е изд. – М. [и др.]: Вильямс, 2005. – 1290 с.: ил.

3. Декартово дерево [Электронный ресурс] // Электрон. дан. – [Б. м.], 2014. – URL: http://habrahabr.ru/post/101818/ (дата обращения 10.10.2014).

4. *Сибирякова В.А.* Введение в язык С: учеб. пособие / В.А. Сибирякова, О.И. Голубева. –Томск : [б. и.], 2011. – Ч. 2. – 49 с.

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ПАРАМЕТРОВ МЕТОДА ВСПОМОГАТЕЛЬНЫХ ИСТОЧНИКОВ НА СЕЧЕНИЯ РАССЕЯНИЯ ТОНКОГО ДИЭЛЕКТРИЧЕСКОГО ЦИЛИНДРА

Е. П. Полин
Томский государственный университет
E-mail: sena7or@sibmail.com

Введение

Значительный интерес для исследователей представляет изучение рассеяния электромагнитных волн в резонансной частотной области тонким диэлектрическим цилиндром. Этот интерес обусловлен необходимостью решения таких практически важных проблем как влияние диэлектрических объектов на характеристики антенн, снижение радиолокационной заметности и др.

Сама по себе задача электромагнитного рассеяния на тонком диэлектрическом цилиндре не является новой. Например, она рассматривалась в работе [1]. Чаще всего для ее решения использовался метод интегральных уравнений, которые затем решались методом моментов с применением различного вида сеток. Такая техника является чрезвычайно громоздкой и является оправданной для объемных неоднородных диэлектрических тел. Для тонких рассеивателей представляется целесообразным использовать более простые методы решения, изначально учитывающие специфику рассеивателей.