Шаяхметов Искандер Мударисович
магистрант
КНИТУ имени А.Н. Туполева – КАИ
г. Казань, Республика Татарстан

Создание обучающей программы по методам обработки поисковых деревьев

Аннотация*: в данной статье автор представляет обучающее программное обеспечение (ПО) по методам обработки поисковых деревьев, а именно: добавление, удаление и поиск вершины. Областью применения данного ПО является кафедра Прикладной Математики и Информатики имени Ю.В. Кожевникова.*

Ключевые слова: *поисковые деревья, электронно-вычислительные машины, двоичное дерево поиска, ключ вершины, терминальная вершина*.

Создание обучающих систем на базе электронно-вычислительных машин (ЭВМ) – это один из перспективных способов повышения эффективности процесса обучения [1].

Целью данной работы является разработка обучающего программного обеспечение по методам обработки деревьев.

Объектом управления при обучении является обучаемый. Субъектами управления выступают преподаватель и обучающая система, т.к. она не заменяет, а дополняет преподавателя.

Двоичным деревом поиска (ДДП) называют дерево, все вершины которого упорядочены, каждая вершина имеет не более двух потомков (назовём их левым и правым), и все вершины, кроме корня, имеют родителя. Вершины, не имеющие потомков, называются листами. ДДП позволяет выполнять следующие основные операции:

– поиск вершины по ключу;

– вставка вершины;

– удаление вершины.

Двоичное дерево может быть логически разбито на уровни. Корень дерева является нулевым уровнем, потомки корня – первым уровнем, их потомки – вторым, и т.д. Глубина дерева это его максимальный уровень. Понятие глубины также может быть описано в терминах пути, то есть глубина дерева есть длина самого длинного пути от корня до листа, если следовать от родительской вершины до потомка. Каждую вершину дерева можно рассматривать как корень поддерева, которое определяется данной вершиной и всеми потомками этой вершины, как прямыми, так и косвенными. Поэтому о дереве можно говорить как о рекурсивной структуре. Эффективность поиска по дереву напрямую связана с его сбалансированностью, то есть с максимальной разницей между глубиной левого и правого поддерева среди всех вершин [2].

Необходимо помнить, что при наличии нескольких вершин с одинаковыми значениями ключа некоторые алгоритмы не будут работать правильно. Например, алгоритм поиска будет всегда возвращать указатель только на одну вершину. Эту проблему можно решить, храня элементы с одинаковыми ключами в одной и той же вершине в виде списка. В таком случае мы будем хранить в одной вершине несколько элементов, но данный случай в статье не рассматривается.

Идея поиска проста. Алгоритм поиска в ДДП по своей природе рекурсивен. При его описании проще всего использовать понятие поддерева. Поиск начинается с корня дерева, который принимается за корень текущего поддерева, и его ключ сравнивается с искомым. Если они равны, то, очевидно, поиск закончен. Если ключ, который мы ищем, оказался больше текущего, то, очевидно, что нужная вершина находится в правом поддереве, иначе – в левом. Далее эта операция повторяется для правого или левого поддерева.

Добавление вершины в ДДП сопряжено с некоторыми проблемами. После добавления ДДП должно сохранить свойство упорядоченности, а это значит, что вершину, куда попало добавлять нельзя. Поэтому, прежде чем вставлять вершину, необходимо подобрать для неё подходящее место, то есть такое место, после вставки в которое, дерево сохранит своё свойство упорядоченности.

Теперь рассмотрим удаление вершины из ДП. По сравнению с добавлением удаление реализуется более сложным алгоритмом, поскольку добавляемая вершина всегда является терминальной, а удаляться может любая, в том числе и нетерминальная. При этом может возникать несколько различных ситуаций.

Ситуация 1. Удаляемая вершина не имеет ни одного потомка, т.е. является терминальной. Удаление реализуется очень легко обнулением соответствующего указателя у родителя.

Ситуация 2. Удаляемая вершина имеет только одного потомка. В этом случае удаляемая вершина вместе со своим потомком и родителем образуют фрагмент линейного списка. Удаление реализуется простым изменением указателя у родительского элемента.

Ситуация 3. Пусть удаляемая вершина имеет двух потомков. Этот случай наиболее сложен, поскольку нельзя просто в родительской вершине изменить соответствующее ссылочное поле на адрес одного из потомков удаляемой вершины. Это может нарушить структуру дерева поиска. Например, замена вершины 6 на одного из ее непосредственных потомков 2 или 8 сразу нарушает структуру дерева поиска.

Существует специальное правило для определения вершины, которая должна заменить удаляемую вершину. Это правило состоит из двух взаимоисключающих действий:

– либо войти в левое поддерево удаляемой вершины и в этом поддереве спуститься как можно глубже, придерживаясь только правых потомков; это позволяет найти в дереве ближайшую меньшую вершину;

– либо войти в правое поддерево удаляемой вершины и спустится в нем как можно глубже придерживаясь только левых потомков; это позволяет найти ближайшую большую вершину.

Таблица 1

Алгоритм поиска вершины

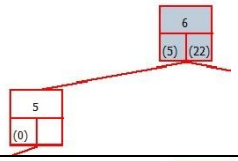
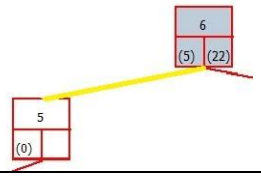
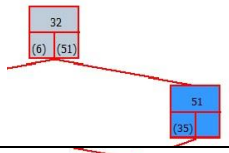
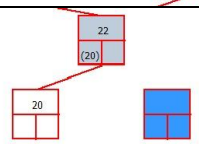
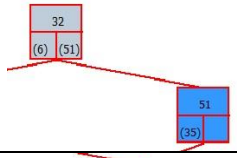
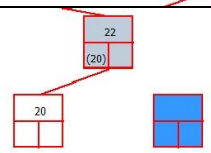
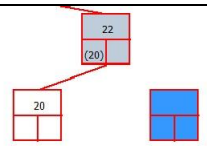
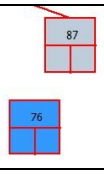
Начиная с корневой вершины для каждого текущего поддерева, выполняем следующие шаги:		
Шаги алгоритма	Визуализация	Пример реализации
Сравниваем ключ вершины с заданным значением x.	Сравниваемую вершину выделяем серым цветом.	
Если заданное значение меньше ключа вершины, переходим к левому потомку, иначе переходим к правому поддереву.	Если переход к левому потомку, то выделяем левое ребро, если к правому потомку, то правое ребро.	
Поиск прекращаем при выполнении одного из двух условий:		
Либо если нашли искомый элемент.	Выделяем найденный элемент, синим цветом.	
Либо если надо продолжать поиск в пустом поддереве, что является признаком отсутствия искомого элемента.	Показываем отсутствия искомого элемента, нарисовав пустую ячейку, синего цвета.	

Таблица 2

Алгоритм добавления вершины

Прежде всего, надо найти подходящее место для нового элемента. Будем считать, что в дерево могут добавляться элементы с одинаковыми ключами, и для этого с каждой вершиной свяжем счетчик числа появления этого ключа. В процесс поиска может возникнуть одна из двух ситуаций:		
Шаги алгоритма	Визуализация	Пример реализации
Найдена вершина с заданным значением ключа, просто увеличивается счетчик.	Выделяем найденный элемент, синим цветом.	
Нашли место в дереве для размещения новой вершины.	Выделяем место для добавления искомого элемента, синим цветом.	
Само добавление включает следующие шаги		
Выделение памяти для новой вершины.	Закрашиваем синим цветом.	
Формирование информационной составляющей	Записываем информационную часть в ячейку.	

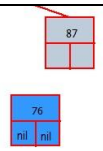
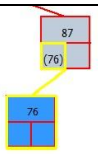
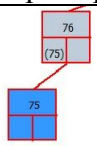
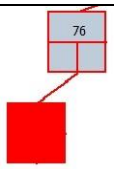
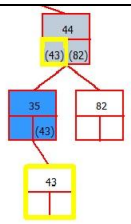
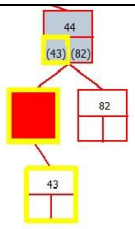
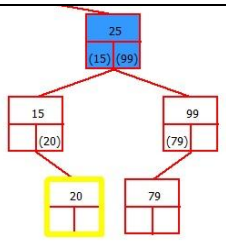
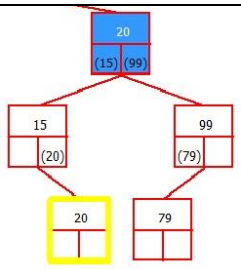
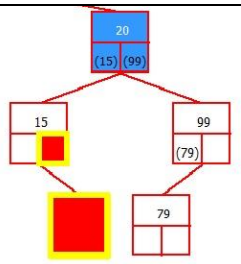
Формирование двух пустых ссылочных полей на будущих потомков.	Записываем в ссылочные поля, нули.	
Формирование в родительской вершине левого или правого ссылочного поля – адреса новой вершины.	Рисуем ребро, и выделяем ссылочного поля родителя желтым цветом, и заносим адрес в ссылочное поле.	

Таблица 3

Алгоритм удаления вершины

По сравнению с добавлением удаление реализуется более сложным алгоритмом, поскольку добавляемая вершина *всегда* является *терминальной*, а удаляться может *любая*, в том числе и нетерминальная. При этом может возникать несколько различных ситуаций.		
Шаги алгоритма	Визуализация	Пример реализации
Найдена вершина с заданным значением ключа.	Выделяем найденный элемент, синим цветом.	
Ситуация 1. Удаляемая вершина является терминальной. Обнулением соответствующего указателя у родителя.	Закрашиваем удаляемую вершину красным цветом.	
Ситуация 2. Удаляемая вершина имеет *только одного потомка*. Изменяем указатель у родительского элемента.	Выделяем потомка удаляемой вершины и соответствующего указателя у родителя желтым цветом.	
Удаляем вершину.	Закрашиваем удаляемую вершину красным цветом.	
Ситуация 3. Удаляемая вершина имеет *двух потомков*. Находим замену для удаляемой вершины.	Выделяем заменитель желтым цветом.	

Делаем замену.	Записываем вместо удаляемой вершины значения заменителя.	
Удаляем заменитель. Обнулением соответствующего указателя у родителя.	Закрашиваем заменителя красным цветом.	

Таким образом, в данной работе была создана программа, позволяющая обучить по теме «Поисковые деревья». Разработанное ПО является законченным и рабочим приложением. С помощью этого приложения можно провести поиск, добавление и удаление вершины дерева.

Список литературы

1. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD.: Пер. с анг. – М.: Ткачев Ф.В.: ДМК Пресс, 2010. - 274стр.

2. Козин А.Н. Структура и алгоритмы обработки данных, 2010. - 193стр.