Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Иркутский государственный университет»

**Институт математики экономики и информатики**

РЕФЕРАТ

на тему:

Двоичные деревья.

Работу выполнила

студент 3 курса

Кузина Ксения Андреевна

Иркутск-2018

**Содержание**

**1)** Введение

2) Понятие структуры данных «двоичное дерево»

3) Назначение

4) Принципы работы и алгоритм получения элемента

5) Алгоритм добавления элемента

6) Алгоритм удаления элемента

7) Обход дерева

8) Заключение

9) Список литературы

Введение

Целью данной работы является изучение структуры данных «двоичные деревья», ее назначения, принципов работы, алгоритмов обхода дерева.

Задачи:

1. Узнать принципы работы двоичных деревьев.
2. Понять как реализуются основные алгоритмы на двоичных деревьях.

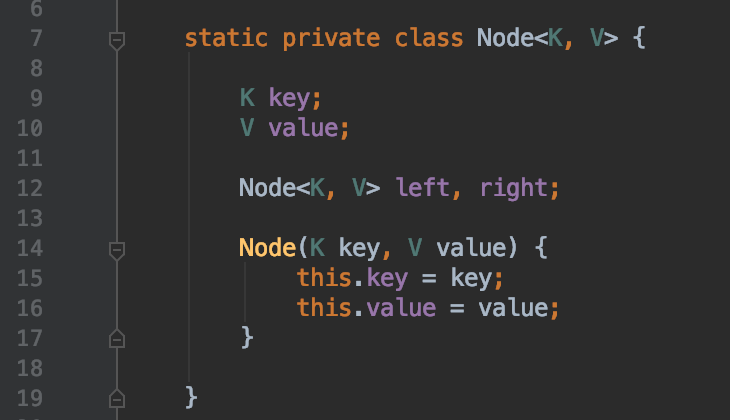
Понятие структуры данных «двоичное дерево».

Двоичное дерево — иерархическая структура данных, в которой каждый узел имеет не более двух дочерних узлов. У дерева есть особая вершина - корень, у которой нет родительских узлов. Вершины, у которых нет дочерних узлов называют листьями.

Каждой вершине дерева можно сопоставить свое поддерево, считая ее корнем этого поддерева.

Левым и правым поддеревьями вершины называют поддеревья с корнями в левом и правом дочернем узле этой вершины соответственно.

Данные хранятся в узлах дерева. В программах узлы дерева представляют структурой данных, хранящей ключ, значение и две ссылки на левый и правый дочерний узел.

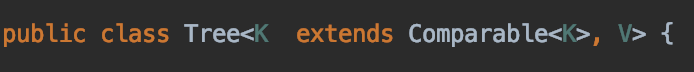


Назначение

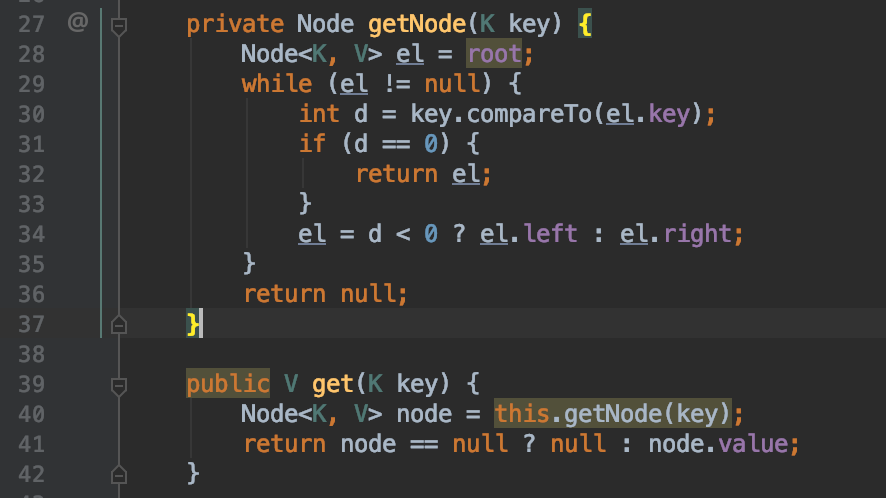
Деревья используют для более быстрого выполнения операций. Операции выполняются за O(h), где h - максимальная глубина дерева (расстояние от корня до вершины). В среднем h = log(n), а в худшем случае h = n, где n - количество вершин дерева.

Основным преимуществом данной структуры данных является высокая эффективность реализации алгоритмов поиска и сортировки.

Принципы работы

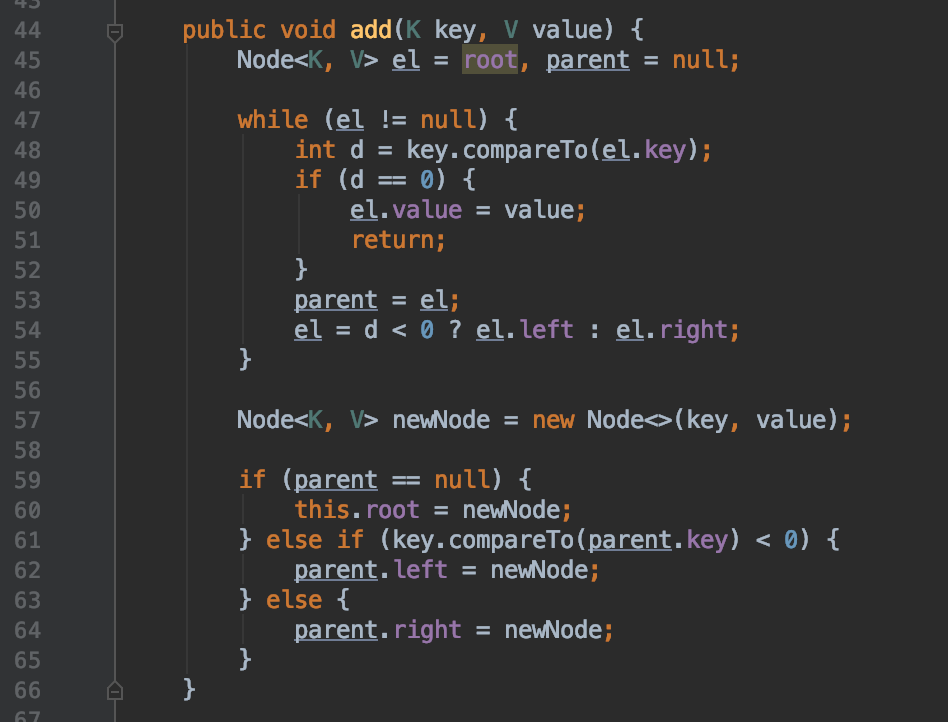
Ключи в дереве должны быть такими, чтобы их можно было сравнивать между собой.

В двоичном дереве применяется следующее правило для хранения данных: Если вершине дерева соответствует некоторый ключ, то левому дочернему узлу соответствует ключ, меньший данного, а правому - больший.

Такое упорядочивание позволяет легко получать значения по конкретным ключам: Будем сравнивать ключ с ключом текущей вершины и если он больше перейдем в правую вершину, а если меньше в левую, ну а если они равны, то нужная вершина найдена.

Добавление элемента

Для добавления значения с некоторым ключом нужно либо найти существующий узел с этим ключом и заменить его значение на новое, либо создать новый узел с таким ключом.

Таким образом, сначала происходит обход дерева и если в процессе этого обхода находится нужный ключ, то его значение заменяется и происходит возврат из метода. При обходе запоминается родительский узел текущего узла, для того, чтобы повторно не обходить дерево при отсутствии заданного ключа в нем.

Если ключ не найден, то создается новый узел, который становится корнем, если родительского узла нет, левым узлом текущего родительского узла, если данный ключ меньше родительского ключа, иначе правым.

Удаление элемента

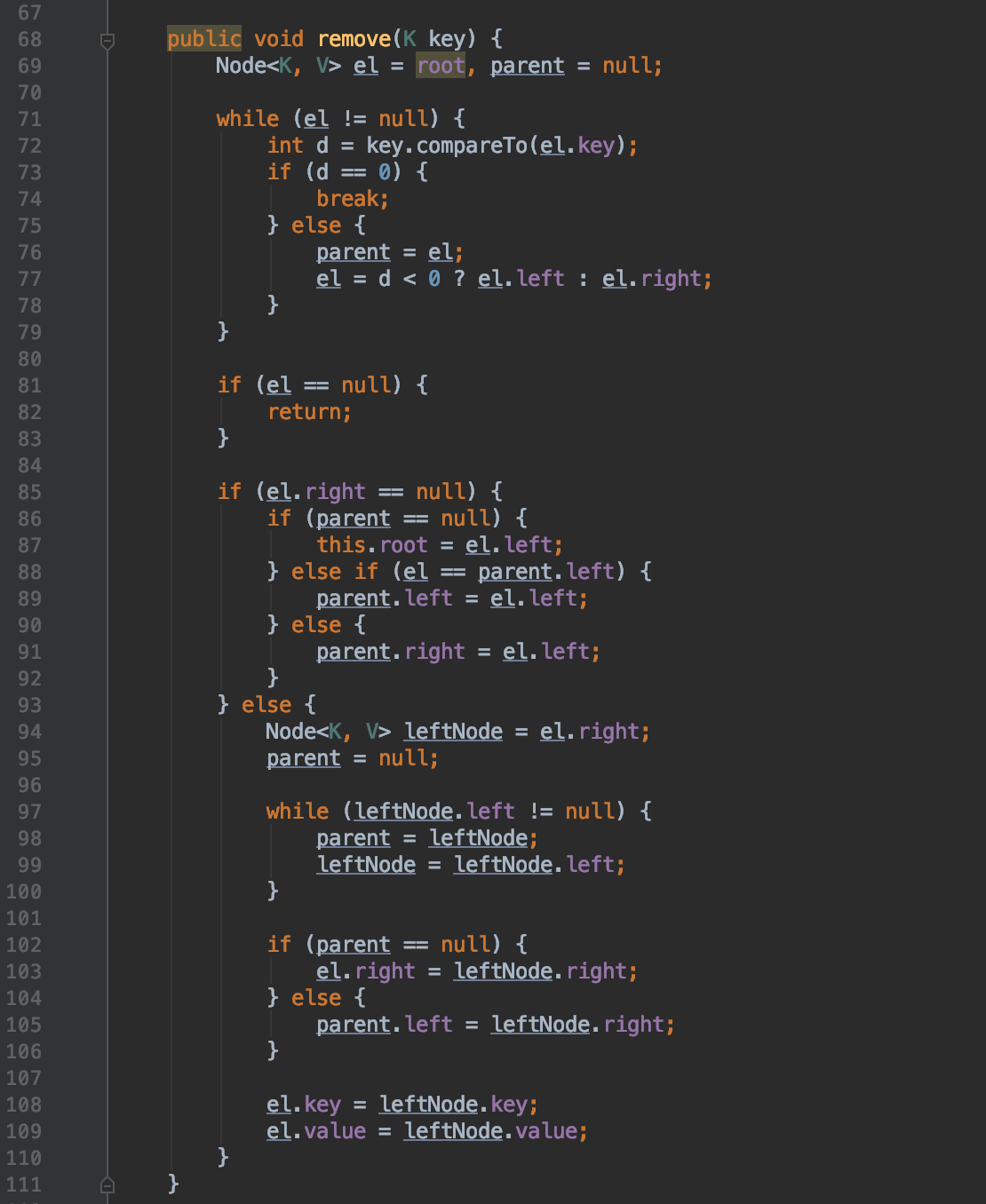
Для удаления узла нужно не только удалить его, но и сохранить структуру дерева.

Сначала ищем узел с заданным ключом, запоминая родительский узел.

Если узел не найден, то выходим из метода.

Если у удаляемой вершины нет правого дочернего узла, то ее левое поддерево станет левым поддеревом ее родительского узла.

Если же у удаляемой вершины есть правый дочерний узел, то найдем в правом поддереве минимум, ясно что он находится в самой последней левой вершине. Заменим удаляемый узел на найденный.

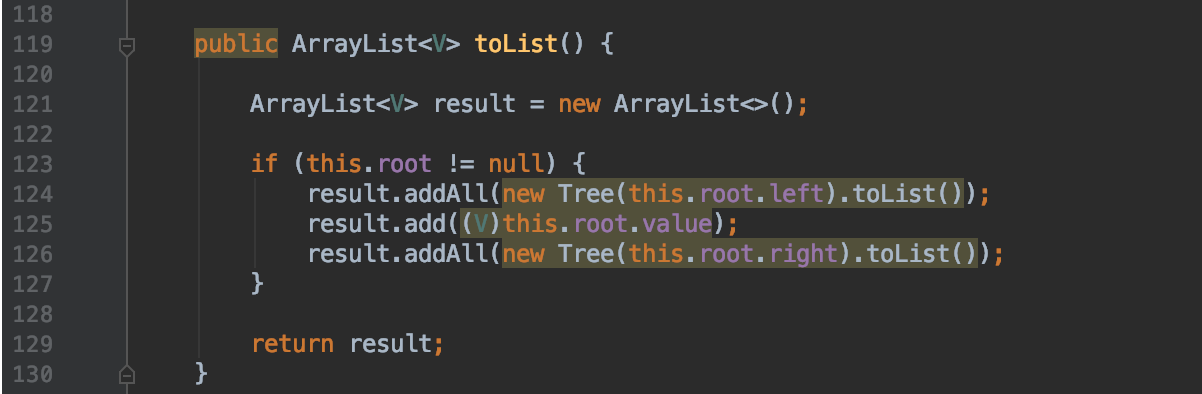


Обход дерева

Есть три операции обхода дерева, отличающиеся порядком обхода узлов.

1. inorder - левое поддерево => вершина => правое поддерево
2. preorder - вершина => левое поддерево => правое поддерево
3. postorder - правое поддерево => вершина => левое поддерево

На картинке проиллюстрирован обход дерева в порядке inorder, для создания списка значений. Для обхода левого и правого поддеревьев используется рекурсия.

Для замены операции обхода нужно поменять местами строки 124, 125, 126.

Заключение

Изучение структуры данных «двоичное дерево» оказалось очень полезным.

На мой взгляд, cамой сложной операцией является удаление элемента. А скорость работы и эффективное выполнение алгоритмов - главные плюсы хранения данных в виде двоичного дерева.

Список литературы

1. <https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0>
2. <https://habr.com/post/65617/>