ИКН НИТУ МИСИС

Комбинаторика и теория графов

Деревья поиска, их представление в компьютере

Исполнитель:

Лосев В.Л. БИВТ-23-18

(<https://github.com/SomethingWF/Bin_Search_tree>)

Москва 2024 год

**Определения**

Дерево – связный, ациклический граф, обладающий следующими свойствами:

1. Конечный связный граф является деревом т. и т.т., когда число ребер (m) и число вершин в графе (n) связаны соотношением m = n - 1.
2. Граф является деревом т. и т.т. две его различные вершины можно соединить единственной простой цепью.
3. Не содержит кратных ребер и петель.
4. Любое дерево однозначно определяется расстояниями (длиной наименьшей цепи) между его концевыми (степени 1) вершинами.
5. Любое дерево является двудольным графом

Бинарное дерево – дерево, степени вершин которого не превосходят трех для неориентированных графов или двух для ориентированных.

В рамках данной работы было реализовано два вида деревьев поиска:

Двоичное дерево поиска – двоичное дерево, для которого выполняются следующие дополнительные условия:

1. Оба поддерева – левое и правое – являются двоичными деревьями поиска.
2. У всех узлов левого поддерева произвольного узла X значения ключей данных меньше либо равны, нежели значение ключа данных самого узла X.
3. У всех узлов правого поддерева произвольного узла X значения ключей данных больше, нежели значение ключа данных самого узла X.

АВЛ-дерево – сбалансированное по высоте двоичное дерево поиска, для каждой вершины которого высота двух ее поддеревьев различается не более чем на 1.

Вышеописанные структуры данных поддерживают следующие операции:

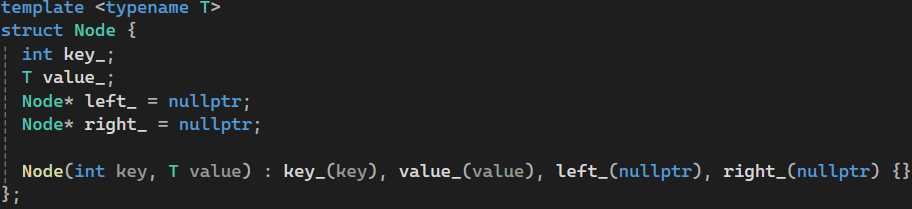
1. search(k) – поиск узла, в котором хранится пара (key, value) с key = k.
2. insert(k, v) – добавление в дерево пары (key, value) = (k, v).
3. remove(k) – удаление узла, в котором хранится пара (key, value) с key = k.

**Перечень инструментов**

Языком программирования, использовавшимся для реализации структур данных, хранящих графы, является C++. В частности, стандартная библиотека шаблонов(STL).

**Описание реализации**

Для реализации узлов бинарного дерева поиска была реализована структура Node, со полями ключа, значения, ссылок на левого и правого потомка узла. Также в коде используются шаблоны, для обеспечения возможности создания деревьев с разными типами данных.

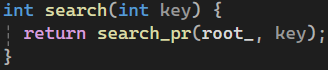


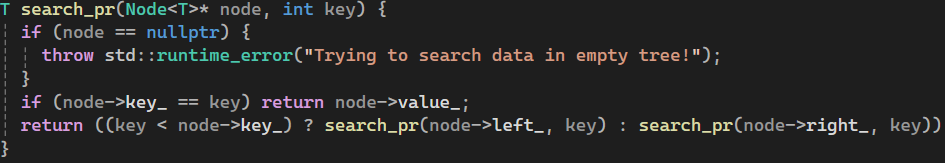
У самого же дерева поиска единственным полем является указатель на корневой узел дерева.



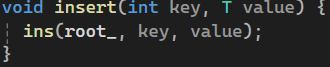
Все операции в двоичных деревьях поиска были реализованы с помощью рекурсии. Поэтому непосредственная реализация рекурсивных функций находится в private-секции класса, а в методах, доступных пользователю, происходят вызовы этих самых функций с указанием корневого узла в качестве точки старта рекурсии.

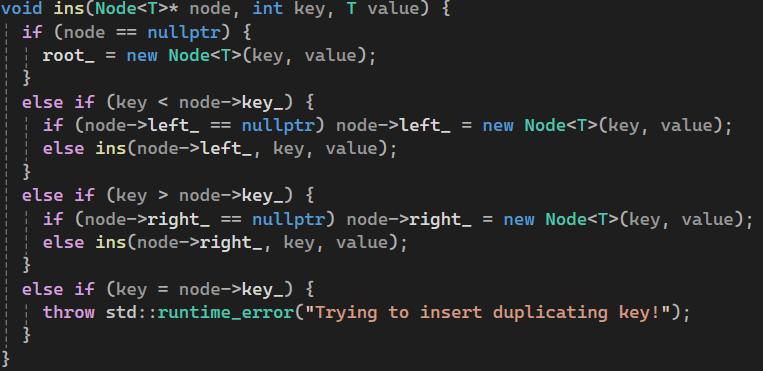
Операция поиска элемента осуществляется путем сравнения ключа текущего узла с искомым. Если искомый ключ меньше – происходит вызов функции для левого потомка текущего узла, если больше – для правого. Также предусмотрен механизм выброса исключения в случае поиска несуществующего ключа.





Операция вставки элемента также основана на сравнении ключа текущего узла с искомым. Особенными случаями являются вставка элемента в пустое дерево и попытка вставки повторяющегося ключа, что вызовет исключение.





Операция удаления помимо сравнения ключей и соответствующего ему рекурсивного вызова, рассматривает два сценария действий.

Первый – у удаляемого узла ноль или один потомок. Предварительно сохранив адрес памяти удаляемого узла, приоритетно выбирается потомок, содержащий какие-либо данные, после чего его адрес сохраняется в текущий узел, который будет передан функции, вызвавшей текущей, поддерживая тем самым связность дерева. После, ввиду особенностей программирования на C++, вручную удаляем искомый узел из динамической памяти с помощью предварительно сохраненного адреса.

Во втором случае у удаляемого узла в потомках имеется два поддерева. В рассмотренной реализации производится поиск узла с наибольшим ключом в левом поддереве. Его данные, в том числе ключ, передаются удаляемому узлу, а ранее найденный наибольший узел левого поддерева удаляется согласно первому сценарию.

