ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

«Структуры и алгоритмы обработки данных»

на тему

ТЕРНАРНОЕ ДЕРЕВО ПОИСКА

|  |  |
| --- | --- |
| Выполнил студент | Кулик Павел Евгеньевич |
|  | Ф.И.О. |

|  |  |
| --- | --- |
| Группы | ИС241 |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работу принял |  | ст. преп. Кафедры ВС Д. М. Берлизов |
|  | подпись |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Защищена |  | Оценка |  |
|  |  |  |  |

Новосибирск – 2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc151082784)

[ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ 4](#_Toc151082785)

[1. Вставка ключа в дерево 4](#_Toc151082786)

[2. Удаление ключа из дерева 5](#_Toc151082787)

[2.1 Вспомогательные операции для удаления 7](#_Toc151082788)

# ВВЕДЕНИЕ

На сегодняшний день разработано и исследовано множество различных структур данных, каждая из которых обладает своими особенностями и может эффективно применяться в подходящих задачах.

В рамках данной курсовой работы мною была изучена и реализована такая структура данных как тернарное дерево поиска. Впервые эта структура была описана в статье под названием "A tree for associative searching" («Дерево для ассоциативного поиска»), написанной Найджелом П. Калдером (Nigel P. Chapman) и Эрнестом С. Коузенсом (Ernest S. Couden). Эта статья была опубликована в журнале "Communications of the ACM" в июне 1971 года (Volume 14, Issue 6, Pages 427-432).

Основная суть тернарного дерева поиска заключается в том, что каждый узел хранит в себе один символ ключа, представляющего собой строку и может иметь до трёх дочерних узлов, а именно нижний (low), равный (equal) и верхний (high). Нижний и верхний узлы используются по тому же принципу, что левый и правый в бинарном дереве поиска, то есть в нижний дочерний узел попадает символ, который располагается ближе к началу алфавита, чем его родитель, а в верхний дочерний узел попадает символ, который находится дальше от начала алфавита, чем его родитель. Равный узел служит для того, чтобы хранить в себе те символы, которые являются продолжением хранящегося в дереве ключа. Помимо символа ключа и указателей на дочерние узлы в каждом узле хранится так же метка, хранящая в себе одно из двух возможных состояний узла, которая нужна для того, чтобы проверить, является ли символ, хранящийся в узле, завершающим символом ключа или нет.

# ОПИСАНИЕ ОСНОВНЫХ ОПЕРАЦИЙ

## 1. Вставка ключа в дерево

Для того, чтобы вставить в дерево новый ключ, требуется сначала найти то место, с которого следует начать вставку. Если дерево пустое, то новый ключ становится корнем дерева, а вот если в дереве уже были ключи, то начинается поиск узла node, с которого начнётся добавление нового ключа.

Изначально узлу node присваивается значение корня дерева. Во время поиска места под новый ключ, первый символ ключа сравнивается с теми символами, которые уже хранятся в дереве. Если символ нового ключа меньше (находится ближе к началу алфавита), чем тот ключ, который уже есть, то узел node заменяется на его нижний дочерний узел, а если символ больше (дальше от начала алфавита), чем тот, что уже добавлен, то узел node заменяется на верхний дочерний узел. При каждой замене узла node в переменную prev сохраняется его предыдущее состояние, а в переменную indicator сохраняется информация о том, на какой из дочерних узлов был заменён узел node для того, чтобы в дальнейшем можно было связать новые узлы с теми, что уже были добавлены. Если возникает такая ситуация, что первый символ ключа уже был добавлен в дерево, то узел node заменяется на свой равный дочерний узел, а исследуемый символ ключа заменяется на его следующий символ, после чего поиск продолжается дальше.

Если в ходе поиска будут найдены все символы ключа, то узел prev, в котором в таком случае хранится последний символ ключа, будет помечен как узел с завершающим символом вне зависимости от того, какая у него до этого была метка, и алгоритм завершит свою работу. Если до запуска алгоритма узел prev был помечен как узел без ключа, то теперь в дереве будет новый ключ, для которого уже были готовы все узлы, а если метка окончания ключа уже стояла, то повторная метка никак не повлияет на дерево.

После того, как место для новых узлов найдено, начинается добавление новых узлов в дерево. При этом первый новый узел может быть по отношению к его родителю как равным, так и нижним или верхним, а вот все следующие за ним узлы будут равными дочерними узлами.

Особенность алгоритма вставки ключей в тернарное дерево поиска заключается в том, что если в дереве уже хранится часть нового ключа, то те символы, которых не было в дереве, будут добавлены в нижнее или верхнее поддерево первого узла с несовпадающим символом, следующего за последним узлом с совпадающим символом. Таким образом, если узел является нижним или верхним по отношению к своему родителю, то символ, хранящийся в родительском узле, точно не является частью ключа, часть которого хранится в дочернем узле, но при этом другая часть ключа может хранится ближе к корню.

## 2. Удаление ключа из дерева

Удаление ключа является более сложной операцией, чем добавление. Во время удаления могут возникать различные неправильные состояния дерева, каждое из которых необходимо правильно обработать.

Удаление начинается с поиска узла, в котором хранится первый символ удаляемого ключа. Это действие происходит с помощью вспомогательной операции “start\_of\_key” (все вспомогательные операции будут описаны после операции удаления). Если такой узел был найден, то он сохраняется в переменную node, при этом в переменную prev сохраняется его родительский узел (если только первый символ ключа не хранится в корне, иначе в переменной prev ничего не хранится), а в переменную indicator сохраняется его отношение к родительскому узлу. Если такой узел не был найден, значит удаляемого ключа нет в дереве и алгоритм завершает свою работу.

Далее происходит поиск всех остальных символов ключа с использованием вспомогательной операции “end\_of\_key”. Помимо поиска данная операция сохраняет в массив key\_trace все узлы, которые лежат на пути от узла, хранящего в себе первый символ ключа до узла, хранящего в себе последний символ. Некоторые узлы на пути могут хранить в себе символы, которые не являются частью ключа из-за особенностей вставки ключей. Помимо этого, в массив indicators сохраняются отношения узлов к их родительским узлам. После выполнения “end\_of\_key” в переменную node сохраняется узел, хранящий в себе последний символ ключа (если он был найден). Если такой символ не был найден, значит соответствующий ключ отсутствует в дереве и алгоритм завершает свою работу.

После того, как все узлы, относящиеся к удаляемому ключу, были найдены, начинается последовательное удаление узлов, которое происходит, c помощью операции “free\_key”, начиная с последнего найденного узла, до тех пор, пока не встретится узел, принадлежащий более чем одному ключу или пока не будет удалён весь ключ, кроме первого символа ключа (узел с первым символом требуется удалять особенным образом). Также после выполнения операции “free\_key” в last\_node и last\_sym сохраняются индекс узла родителя последнего удалённого узла и индекс символа ключа, который ещё хранится в дереве.

Проверка осуществляется следующим образом: если какой-либо узел на пути от узла, хранящего в себе первый символ ключа, к узлу, хранящему в себе последний символ, помечен как узел, хранящий в себе последний символ какого-либо ключа или имеет дочерние узлы, кроме удаляемых, или является узлом, хранящим в себе первый символ удаляемого ключа или символ, которого нет в удаляемом ключе, то удаление прекращается.

После того, как были удалены все узлы, которые не нарушали условия, в переменную node сохраняется родитель последнего удалённого узла и начинается проверка нарушений свойств дерева, которые могли возникнуть в ходе удаления.

Первое условие проверяет, нет ли у node равного дочернего узла. Если такой узел есть, значит последний удалённый узел был нижним, либо верхним дочерним узлом node. В таком случае более никаких действий не требуется и алгоритм завершает свою работу.

Следующее условие проверяет, происходит ли удаление корня при удалении ключа из дерева. Если складывается такая ситуация, значит переменная prev ничего не хранит, при этом были удалены все символы ключа, кроме первого и первый символ ключа не является завершающим символом целого ключа.

Если условие выполняется и происходит удаление корня, то вызывается вспомогательная операция “tst\_restore\_prop\_any\_case”, которая перестраивает дерево для того, чтобы обозначить новый корень и сохранить бинарные свойства дерева. Подробности этой вспомогательной операции будут описаны позже. Далее алгоритм завершает свою работу.

Если не происходит удаление корня, то проверяется тот случай, при котором узел node помечен как конец ключа и не имеет дочерних узлов. В таком случае более никаких действий не требуется и возвращается корень дерева.

После этого происходит последняя проверка с одним условием, при котором node может быть узлом, хранящим первый символ ключа, либо любой другой символ. Если node хранит в себе первый символ ключа, то происходит восстановление свойств дерева вспомогательной операцией “tst\_restore\_prop”, которая принимает на вход индикатор, узел node и его родителя из массива key\_trace. В противном случае на вход подаются значения первого элемента indicators, переменой node и prev. После чего происходит завершение операции удаления.

## 2.1 Вспомогательные операции для удаления