Здравствуйте. Тема курсовой работы: «Структура данных “Splay-дерево”»

**Научный руководитель: кандидат физико-математических наук, доцент Русакова Маргарита Сергеевна**

**Слайд 2:**

В настоящее время с целью организации и систематизации данных разрабатываются **разные алгоритмы и структуры данных, которые позволяют упростить процесс поиска данных.**

Структурой данных, позволяющей быстрее находить те данные, которые использовались недавно, является разработанное в 1983 году Дэниелом Слейтером и Робертом Тарьяном **Splay-дерево**.

**Splay-дерево** используют при кэшировании данных, маршрутизации в сетях и в алгоритмах сжатия данных.

**Слайд 3:**

Целью работы является изучение алгоритмов обработки Splay-деревьев и особенностей их применения при решении практических задач.

Для достижения поставленной цели необходимо решить следующие задачи, которые представлены на слайде:

– изучить структуру данных Splay-дерева, ее особенности и свойства;

– провести анализ, оценку трудоемкости и асимптотической сложности алгоритмов работы со Splay-деревьями;

– провести проектирование, разработку и тестирование приложения, использующего Splay-дерево для поиска информации о поездах на железнодорожном вокзале.

**Слайд 4:**

Splay-дерево — это самобалансирующаяся структура данных, в которой последний ключ, к которому осуществлялся доступ, всегда помещается в корень.

При этом Splay-дерево не обеспечивает идеальную балансировку и на отдельных запросах может работать даже линейное время.

Поднятие узла в корень происходит с помощью «поворотов».

Каждый узел дерева содержит:

– ссылку на левое поддерево;

– ссылку на правое поддерево;

– полезную информацию.

**Слайд 5:**

Сравнивая существующие структуры данных стоит отметить, что в отличие от АВЛ-дерева, Splay-дереву не нужно хранить дополнительную информацию (например, высота в АВЛ-дереве), что делает его эффективным по памяти.

**После каждого запроса Splay-дерево меняет свою структуру**.

**Асимптотическая сложность** Splay-дерева в большинстве случаев составляет .

Однако использование Splay-дерева избегают в тех случаях, когда требуется строгое соблюдение времени выполнения, так как в худшем случае асимптотическая сложность равна .

**Слайд 6:**

Добавление нового элемента в Splay-дерево происходит по правилам бинарного дерева поиска, однако после добавления узла он поднимается в корень.

Добавление нового узла состоит из нескольких этапов:

1) спуск по дереву до места вставки узла;

2)добавление узла в дерево;

3)поднятие добавленного узла в корень.

**Слайд 7:**

Поиск узла в Splay-дереве осуществляется в три этапа:

1) поиск искомого узла;

2) поднятие искомого узла в корень;

3) метод возвращает ссылку на искомый узел.

**Слайд 8:**

Удаление узла в Splay-дереве осуществляется в три этапа:

1) поиск удаляемого узла;

2) поднятие узла в корень;

3) удаление узла.

**Слайд 9:**

Поднятие узла в корень в Splay-дереве осуществляется с помощью поворотов.

В Splay-дереве используется 6 видов поворотов:

малый левый поворот (zig);

– малый правый поворот (zag);

**Слайд 10:**

– большой левый поворот (zig-zig);

– большой правый поворот (zag-zag);

– левый-правый поворот (zig-zag);

– правый-левый поворот (zag-zig).

**Слайд 11:**

Для программной реализации решения рассматриваемой задачи был выбран язык программирования C++ и редактор кода Visual Studio Code.

В курсовой работе было разработано приложение, в котором:

* Программно реализована структура Splay-дерева;
* Реализована программа для поиска информации о поездах на железнодорожном вокзале.

**Слайд 12:**

**Постановка задачи:** Имеется автоматизированная информационная система на железнодорожном вокзале, которая содержит сведения об отправлении поездов дальнего следования и организована в виде Splay-дерева. Для каждого поезда указываются номер, станция назначения, время отправления.

**Разработать программу, которая обеспечит:**

**–** первоначальный ввод данных в систему и формирование Splay-дерева;

– добавление информации о поездах;

– удаление информации о поездах;

– поиск информации о поезде по его номеру.

**Слайд 13:**

На слайде представлены результаты проверки операции добавления, поиска и удаления.

На вход подается файл с информацией о поездах: номер поезда, следующая станция и время отправления.

Ключом каждого узла является номер поезда. Далее совершаются такие действия, как добавление и удаление информации о поездах, а также поиск информации о поезде по его номеру. Каждое действие сопровождается поднятием в корень узла, над которым оно выполняется. Таким образом узлы, к которым несколько раз происходит обращение, находятся в корне дерева, и их поиск занимает минимальное кол-во времени и имеет асимптотическую сложность .

Вывод информации о всех поездах осуществляется полным перебором дерева. Таким образом, асимптотическая сложность этого действия .

Добавление узла происходит по правилам бинарного дерева поиска. После размещения узла в дерево он поднимается в корень.

Поиск происходит по центральному перебору узлов (родительский узел, левое поддерево, правое поддерево). Найденный узел поднимается в корень. Средним значением асимптотической сложности является , в лучшем случае — , в худшем случае — .

Удаление происходит поиском удаляемого узла и поднятием его в корень с последующим удалением.

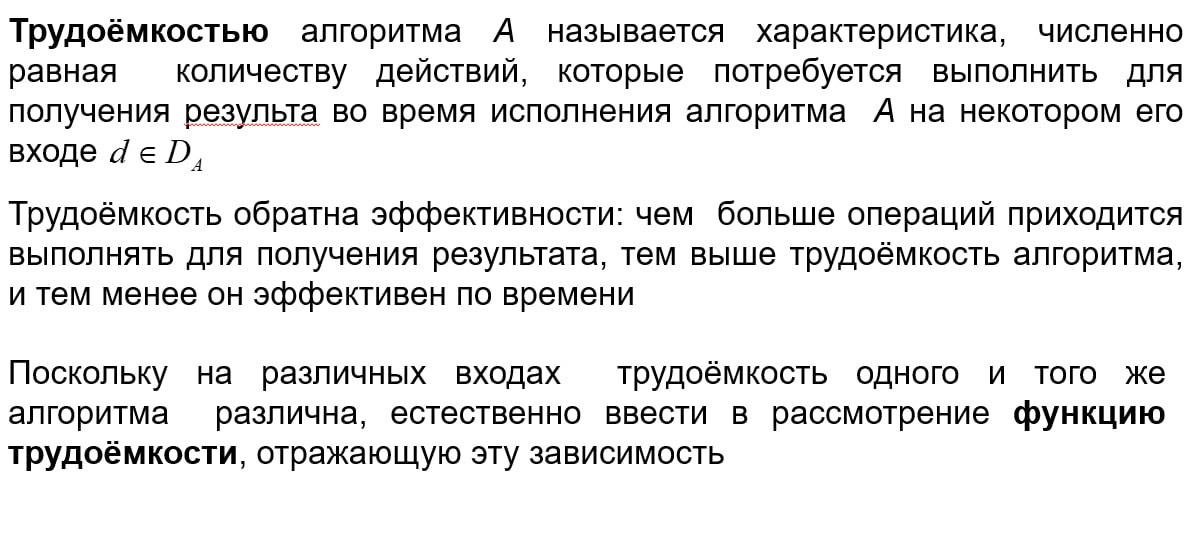
**Слайд 14:**

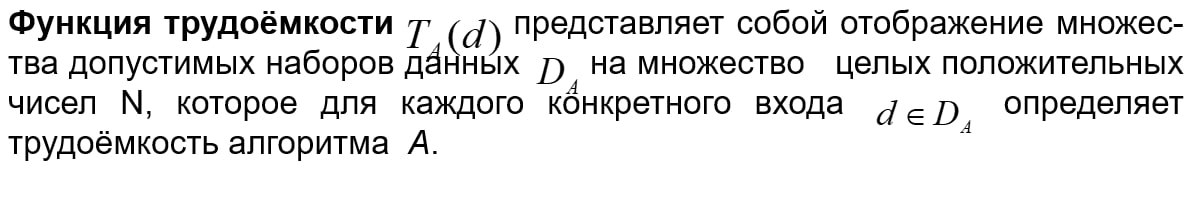
В процессе выполнения курсовой работы было сделано следующее:

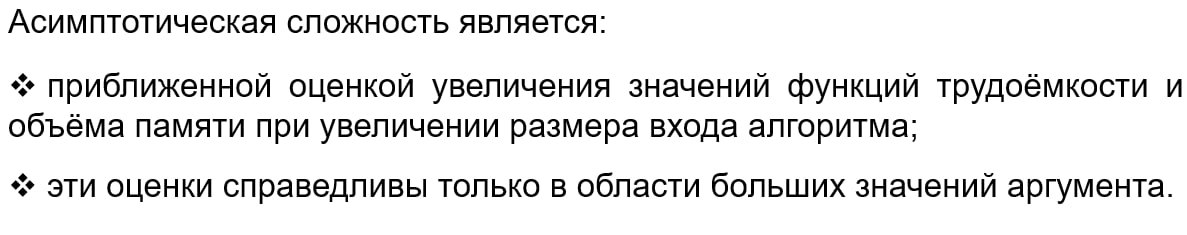
* Изучена структура данных Splay-дерева~~, ее особенности и свойства~~;
* Проведен анализ алгоритмов добавления, поиска и удаления элемента в Splay-дереве;
* Проведена оценка трудоемкости и асимптотической сложности алгоритмов работы со Splay-деревьями;
* Спроектировано, разработано и протестировано приложение, использующее Splay-дерево для поиска информации о поездах на железнодорожном вокзале.

**Слайд 15:** Благодарю за внимание!

**Редактор кода Visual Studio Code** предоставляет широкий спектр возможностей для разработки программного обеспечения и высокую производительность.

**Язык С++ — мощный универсальный язык программирования**, обеспечивающий высокую производительность и контроль над системными ресурсами. Область применения включает его использование для разработки программного обеспечения, приложений реального времени, создания операционных систем и драйверов. 





**Кэш –** сверхбыстрая память, которая является связующим между оперативной памятью и процессором.

**Дерево** — структура данных, элементы которой, называемые вершинами (узлами), связаны отношениями подчиненности, когда одному элементу может быть подчинено несколько, но при этом сам он может быть подчинен только одному [1].

**Бинарное дерево поиска** — иерархическая структура данных, в которой каждый узел имеет значение и ссылки на узлы левых и правых поддеревьев.

**Поддерево** — часть древообразной структуры данных, которая может быть представлена в виде отдельного дерева.

**Корневой узел** — узел на самом верхнем уровне, на котором начинается выполнение большинства операций над деревом (не имеющий родительских узлов).

**Родительский узел** — узел, которому подчиняются дочерние узлы.

**Дочерний узел** — узел, который подчиняется одному родительскому узлу.

**Листовой узел** — узел, не имеющий дочерних элементов.