Институт информационных технологий, математики и механики

Направление подготовки: Фундаментальная информатика и информационные технологии

**Отчет по лабораторной работе №1**

**«Сравнение способов хранения данных**

**на примере хранения и использования полиномов»**

**Выполнил**:студент группы 3822Б1Фи3

Сдобнов Владимир Александрович

**Проверила**:

ассистент кафедры МОСТ ИИТММ,

Усова М.А.

**Содержание**

[Введение 3](#_Toc165537870)

[Постановка задачи 4](#_Toc165537871)

[Руководство программиста 4](#_Toc165537872)

[Описание структуры программы 4](#_Toc165537873)

[Описание структур данных и алгоритмов 5](#_Toc165537874)

[Полиномы 5](#_Toc165537875)

[Бинарное дерево общего вида 5](#_Toc165537876)

[Бинарное дерево поиска 5](#_Toc165537877)

[Неупорядоченные таблицы (на массиве и на списке) 5](#_Toc165537878)

[Упорядоченные таблицы (на массиве и на списке) 5](#_Toc165537879)

[Хеш-таблицы (с открытой адресацией и методом цепочек) 5](#_Toc165537880)

[Куча 6](#_Toc165537881)

[Красно-черное дерево 6](#_Toc165537882)

[АВЛ-дерево 6](#_Toc165537883)

[Вычислительный эксперимент 6](#_Toc165537884)

[Описание вычислительных экспериментов 6](#_Toc165537885)

[Результаты вычислительных экспериментов 6](#_Toc165537886)

[Представление полиномов 6](#_Toc165537887)

[Хранение набора полиномов 9](#_Toc165537888)

[Заключение 9](#_Toc165537889)

[Литература 9](#_Toc165537890)

[Приложения 10](#_Toc165537891)

[Приложение 1. Функция бинарного поиска 10](#_Toc165537892)

Введение

Во время разработки программисты очень часто сталкиваются с проблемой выбора подходящей структуры хранения данных. Техническое задание почти всегда накладывает некоторые ограничения на скорость выполнения каких-либо операций с данными в структуре, таких как вставка, поиск и удаление, а также на затрачиваемой на это памятью.

Под различные ограничение существуют разные способы хранения информации в программах. Для оценки их эффективности вычисляются временный и пространственные сложности алгоритмов, используемых в системах данных. Временная сложность – время необходимое для завершения алгоритма, а пространственная сложность – требуемый объем памяти или дискового пространства. Однако оценка эффективности структуры хранения зависит не только от количества, но и от их типа.

Данная лабораторная работа направлена на изучение классических структур данных и их сравнение в терминах вычислительной сложности. В качества хранимых данных используются полиномы в общей аналитической форме, так как они широко используются в приложениях. Для проведения исследования разработаны программы обработки полиномов и структуры хранения данных.

В работе запланированы эксперименты, направленные на определение фактической сложности работы с полиномами, а также на оценку сложности работы различных разработанных систем хранения данных с полиномами.

Постановка задачи

В постановке задачи приводятся формулировки решаемых задач, а также приводятся функциональные и не функциональные требования.

Под полиномом понимается выражение вида:

или в более общем виде

где – степень полинома, – коэффициенты полинома (действительные), i, j, k – степени монома (целые).

В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективное представление полиномов, а также различные способы их хранения.

Предполагается, что в качестве структуры представления полинома будет использоваться массив. Полиномы будут допускать выполнение следующих операций над ними:

1. Сложение с мономом,
2. Сложение с полиномом,
3. Вычитание монома,
4. Вычитание полинома,
5. Умножение на константу,
6. Умножение на моном,
7. Умножение на полином,
8. Деление не константу,
9. Деление на моном.

В качестве структур хранения набора полиномов будут использоваться основные общеизвестные структуры данных, обладающие различными достоинствами и недостатками:

* Бинарное дерево общего вида,
* Бинарное дерево поиска,
* Неупорядоченные таблицы (на массиве и на списке),
* Упорядоченные таблицы (на массиве и на списке),
* Хеш-таблицы (с открытой адресацией и методом цепочек),
* Красно-черное дерево,

В числе операций над структурами хранения, которые будут исследованы:

* вставка,
* поиск,
* удаление.

Задача лабораторной работы измерить фактическую сложность операций над структурами хранения, разработанными самостоятельно, с теоретической сложностью данных операций и сложностью данных алгоритмов в библиотеке STL, а также определить эффективность представления полиномов.

Руководство программиста

В начале проводятся какие-то общие факты о вашем программном модуле. Можно нарисовать блок-схему работы вашего программного комплекса.

Основная часть руководства программиста состоит из нескольких частей.

Описание структуры программы

Про модули (статические библиотеки, тесты, как собирается и прочее).

Основные классы, их поля, функционал. СХЕМЫ СИСТЕМ КЛАССОВ!

Так как проект большой ОПИСАНИЕ ФУНКЦИЙ МОЖНО НЕ ПРИВОДИТЬ (в последствие рекомендуется оформлять самодокументирующийся код, чтобы прикладывать ссылку на автоматически генерирующуюся документацию или брать описание оттуда).

Описание структур данных и алгоритмов

Во второй части приводится описание использованных или реализованных алгоритмов. Для описания алгоритма вначале приводится текстовое описание. Затем по необходимости приводятся блок схемы и/или псевдокод. Сложные алгоритмы следует сопровождать примерами!

ВЕЗДЕ ПРИВЕСТИ ТЕОРЕТИЧЕСКИЕ СЛОЖНОСТИ!

ДЛЯ СЛОЖНЫХ СТРУКТУР ДАННЫХ ЛУЧШИЙ КОММЕНТАРИЙ — ЭТО КАРТИНКА!

Полиномы

Класс представления полинома:

Polinom

TVector<Monom> monoms;

TVector<Lexem> variebles;

Поле monoms содержит в себе все ненулевые мономы, упорядоченные по степеням, из которых состоит полином. Поле variebles хранит лексемы с именами переменных и их значениями. Для данного класса реализовано несколько конструкторов, помимо инициализации и копирования:

1. По вектору лексем. Данный конструктор предполагается использовать для создания полиномов с одинаковыми наборами переменных. В поле monoms создается пустой вектор.
2. По строке. Данный конструктор необходим для ручного ввода полиномов.

В классе так же реализованы операторы сложения полинома с мономом и полиномом, вычитания монома и полинома, умножения на константу, моном и полином, деления на константу и моном.

Сложение: базовой операцией является сложение с мономом. Добавлена отдельная функция приоритетной вставки, которая бинарным поиском ищет место, где должен находится данный моном. Если в это месте находится моном с тем же набором степеней, то они складываются. Если данное место занято другими мономом, то производится вставка в вектор. Теоретическая сложность О(logN) Сложение с полиномом — это сложение данного полинома со всеми мономами другого полинома в цикле. Теоретическая сложность O(mlogN), где m – число мономов во втором полиноме

Вычитание: сложение полинома с элементом с противоположным знаком. Теоретическая сложность аналогична сложению

Умножение: при умножении на константу или моном, все мономы в полиноме умножаются на данный элемент. Теоретическая сложность О(N). При умножении полиномов данный полином умножается на каждый моном другого полинома и полученные произведения складываются в результирующий полином. Теоретическая сложность О(m \* N \* logN).

Деление: при делении каждый моном полинома делится на элемент. Сложность аналогична умножению.

Бинарное дерево общего вида

В данной работе используется однонаправленное бинарное дерево. Дерево состоит из звеньев каждое из которых хранит данные и указывает на другие два звена (или nullptr). Для данного дерева реализована вставка элемента, удаление элемента по указателю и поиск элемента.

Вставка: при вставке элемента дерево обходится в ширину до ближайшего звена с хотя бы одним пустым указателем, после чего создается звено, хранящее новый элемент и указатель на него вставляется вместо пустого указателя. Теоретическая сложность O(N).

TNodeTree<T>

TNodeTree<T>\* left;

TNodeTree<T>\* right;

T data;

TBinarTree<T>

TNodeTree<T>\* root;

Поиск: при поиске используется алгоритм рекурсивного обхода в глубину. Функции возвращает указатель на звено, в котором хранятся данные. Если такого звена нет возвращает nullptr. Теоретическая сложность О(N).

Удаление: при удалении сначала по значению находится указатель на звено. Затем с помощью обхода в ширину находится последнее вставленное звено (далее replacment) и передает данные в звено под удаление. После чего поиском в глубину находится предок replacement и указатель на замену обнуляется. Теоретическая сложность O(N).

Бинарное дерево поиска

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Неупорядоченные таблицы (на массиве и на списке)

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Упорядоченные таблицы (на массиве и на списке)

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Хеш-таблицы (с открытой адресацией и методом цепочек)

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Куча

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Красно-черное дерево

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

АВЛ-дерево

Общее описание …

**Вставка**. …

**Поиск**. …

**Удаление**. …

Вычислительный эксперимент

В этом разделе приводятся описание проводимых численных экспериментов и анализ полученных результатов.

Данная программа написана с помощью среды программирования Microsoft Visual Studio 2019 на языке C++. Проект выполнен с использованием модульных тестов (Google test framework) и средства автоматизации сборки ПО из исходного кода cmake. Исходный код проекта выложен на github [ссылка из списка литературы на ваш репозиторий].

Вычислительные эксперименты проводились на компьютере с процессором Intel Core i7 10750H CPU 2.6GHz.

Описание вычислительных экспериментов

Для создания тестовых наборов применялась функция генерации случайного полинома. Данная функция создает полином с заданным числом мономов со случайными степенями переменных в заданном диапазоне. В рамках эксперимента создавались полиномы с тремя переменными, коэффициенты мономов действительные числа в пределах от 1 до 1000, степени переменных в диапазоне от 0 до 100.

Для замера времени работы функций использовалась функция из стандартно библиотеки <time.h> - clock(). С ее помощью отмечалось время начала эксперимента, затем время окончания. Разность между временем начала и конца эксперимента и есть время проведения эксперимента.

Первым был проведен эксперимент по измерению эффективности представления полинома. В данном эксперименте генерировались два полинома с указанным в таблице 1 числом мономов, а затем над ними проводилась операция, указанная в заголовке таблицы 1.

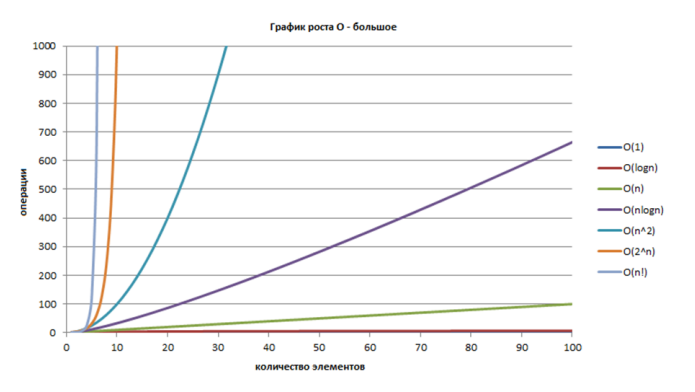
Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Количество мономов полинома** | **Сложение полиномов** | | **Умножение полиномов** | |
| **Время (сек.)** | **Замедление** | **Время (сек.)** | **Замедление** |
| 100 | 30 | - | 59 | - |
|
| 1000 | 278 |  | 592 |  |
|
| 10000 | 2659 |  | 5740 |  |
|
| 100000 | 25003 |  | 57674 |  |
|
| **Факт. сложность** |  | |  | |
| **Теор. сложность** |  | |  | |

Затем был проведен эксперимент по оценке эффективности различных структур хранения….

ПОСТРОИТЬ ГРАФИКИ, В КОТОРЫХ БЫЛО БЫ ВИДНО СЛОЖНОСТЬ, возможно потребуется добавить еще некоторое количество испытаний чтобы график был более точный. График разных видов роста предложен дальше.

Таблица 2



Провести короткий анализ промежуточных результатов.

На основе полученных результатов экспериментов сделать выводы об эффективности реализованного способа представления полинома.

Хранение набора полиномов

Таблица 4 Сравнение эффективности хранения набора полиномов в среднем случае

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Структура данных** | **Размер хранилища** | Вставка | | | | | | Поиск | | | | | | | Удаление | | | | | | | |
| **Время (сек.)** | | | **Замедление** | | | **Время (сек.)** | | **Замедление** | | | | | **Время (сек.)** | | | | **Замедление** | | | |
| Бинарное дерево общего вида | 100 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 1000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 10000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 100000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Бинарное дерево поиска | 100 |  |  | | | | |  | |  | | | | |  | | |  | | | | |
| 1000 |  |  | | | | |  | |  | | | | |  | | |  | | | | |
| 10000 |  |  | | | | |  | |  | | | | |  | | |  | | | | |
| 100000 |  |  | | | | |  | |  | | | | |  | | |  | | | | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Черно-красное дерево | 100 |  |  | | | | |  | | | |  | | |  | | | | |  | | |
| 1000 |  |  | | | | |  | | | |  | | |  | | | | |  | | |
| 10000 |  |  | | | | |  | | | |  | | |  | | | | |  | | |
| 100000 |  |  | | | | |  | | | |  | | |  | | | | |  | | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Неупорядоченная таблица на массиве | 100 |  | |  | | | |  | | |  | | | | | |  | |  | | | |
| 1000 |  | |  | | | |  | | |  | | | | | |  | |  | | | |
| 10000 |  | |  | | | |  | | |  | | | | | |  | |  | | | |
| 100000 |  | |  | | | |  | | |  | | | | | |  | |  | | | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Неупорядоченная таблица на листе | 100 |  | | | |  | |  | | | | | |  | |  | | | | | |  |
| 1000 |  | | | |  | |  | | | | | |  | |  | | | | | |  |
| 10000 |  | | | |  | |  | | | | | |  | |  | | | | | |  |
| 100000 |  | | | |  | |  | | | | | |  | |  | | | | | |  |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Хэш-таблица с шагом | 100 |  | | | | |  | |  | | | |  | | |  | | | | |  | |
| 1000 |  | | | | |  | |  | | | |  | | |  | | | | |  | |
| 10000 |  | | | | |  | |  | | | |  | | |  | | | | |  | |
| 100000 |  | | | | |  | |  | | | |  | | |  | | | | |  | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| Хэш-таблица с цепочкой | 100 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 1000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 10000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| 100000 |  | | |  | | |  | |  | | | | |  | | | |  | | | |
| **Факт. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |
| **Теор. сложность** |  | | | | | |  | | | | | | |  | | | | | | | |

+ ГРАФИКИ!!!

На основе полученных результатов экспериментов

* сделать выводы об эффективности разных способов хранения,
* выявить недостатки и достоинства использованных СД,
* порефлексировать о приложениях в которых могут быть использованы те или иные способы хранения,
* выбрать наиболее оптимальный способ хранения в данной задаче (не только с учётом эффективности, но и с учетом требуемых усилий по реализации и поддержке ПО).

Заключение

В заключении обобщаются выводы, полученные в результате выполнения численных экспериментов, а также приводятся цели, которых удалось достичь в процессе выполнения лабораторной работы.

Литература

Раздел отчета содержит список литературы, оформленный согласно рекомендациям ГОСТ.

Пример списка литературы:

1. Кнут Д. Э. Искусство программирования. Том 1. Основные алгоритмы = The Art of Computer Programming. Volume 1. Fundamental Algorithms / под ред. С. Г. Тригуб (гл. 1), Ю. Г. Гордиенко (гл. 2) и И. В. Красикова (разд. 2.5 и 2.6). — 3. — Москва: Вильямс, 2002. — Т. 1. — 720 с. — ISBN 5-8459-0080-8.
2. Кнут Д. Э. Искусство программирования. Том 2. Получисленные алгоритмы = The Art of Computer Programming. Volume 2. Seminumerical Algorithms / под ред. Л. Ф. Козаченко (гл. 3, разд. 4.6.4 и 4.7), В. Т. Тертышного (гл. 4) и И. В. Красикова (разд. 4.6). — 3. — Москва: Вильямс, 2001. — Т. 2. — 832 с. — ISBN 5-8459-0081-6.
3. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с. — ISBN 5-8459-0082-1.
4. Кнут Д. Э. Искусство программирования, том 4, A. Комбинаторные алгоритмы, часть 1 = The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1 / под ред. Ю. В. Козаченко. — 1. — Москва: Вильямс, 2013. — Т. 4. — 960 с. — ISBN 978-5-8459-1744-7.
5. Учебный курс «*Методы программирование-2: алгоритмы и структуры данных*». *© Гергель В. П.*
6. Учебное пособие «Программа общего курса и описание лабораторных работ» © Балло Л.В., Барышева И.В., Гергель В.П., Гришагин В.А., Долгов Г.А., Кулакова А.П, Стронгин Р.Г.
7. Сайт algolist.manual . Сортировка выбором <http://algolist.manual.ru/sort/select_sort.php>
8. Сайт algolist.manual . Сортировка пузырьком <http://algolist.manual.ru/sort/bubble_sort.php>
9. Сайт studopedia. Сортировка вставками <https://studopedia.ru/5_159643_sortirovka-viborom.html>
10. Сайт intuit.ru. Сортировка Хоара <https://www.intuit.ru/studies/courses/648/504/lecture/11472?page=2>

Приложения

В качестве приложения приводится код. Как правило вставляется НЕ ВЕСЬ КОД, а только существенно важные куски кода.

Приложение 1. Интерфейс Таблиц

UnorderedTableList<TKey, TValue>

TList<TPair<TKey, TValue>> data;

UnorderedTableVec<TKey, TValue>

TVector<TPair<TKey, TValue>> data;

TPair<TKey, TValue>

TKey \_key;

Tvalue \_val

TQueueOnList<T>

TList<T> list;

Monom

TVector<float> powers;

Float coefficient;

Lexem

LexemTypes \_type;

Float \_value;

Std::string name;

TList<T>

Size\_t count;

TNode<T>\* head;

TNode<T>\* tail

TVector<T>

Int \_size;

Int \_count;

T\* pMem;