**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Реализация структуры данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Иванов Д.В. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Пименов П.В. | | |
| Группа 3343 | | |
| Тема работы: Реализация структуры данных | | |
| Исходные данные:  Вариант на оценку «хорошо» № 2 | | |
| Содержание пояснительной записки: разделы «Содержание», «Введение», «Задание», «Исследование», «Описание реализованной программы», «Анализ полученных результатов», «Заключение», «Список использованных источников», «Приложение А» | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 06.11.2024 | | |
| Дата сдачи реферата: 19.12.2024 | | |
| Дата защиты реферата: 19.12.2024 | | |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Иванов Д.В. |

**Аннотация**

В ходе курсовой работы создана программа, реализующая хранение данных о абитуриентах, проходящих отбор в ВУЗ. Исходя из требований задания выбрана и реализована подходящая по параметрам структура данных. Для быстрого добавления данных используется оптимизированный двумерный динамический массив. Результатом работы является класс StudentArray, отвечающий за хранение данных о абитуриентах. Кроме того, созданы юнит-тесты для различных сценариев работы со структурой данных, простой интерфейс командной строки для демонстрации работы со структурой данных, а также функция визуализации структуры данных.

**содержание**

**введение**

Цель курсовой работы – выбор и реализация структуры данных, оптимально подходящей под требования поставленного задания. Для выполнения этой цели необходимо решить следующие задачи:

1. Сравнить различные структуры данных, которые могут подойти для выполнения задания.
2. Изучить недостатки и преимущества этих структур.
3. На основе полученной информации выбрать структуру данных.
4. Реализовать выбранную структуру данных.

**1. задание**

Вариант 2

В магазине мясопродуктов продается бройлер. Покупатели предпочитают тушки побольше, поэтому всегда выбирают курицу максимального веса. Каждый день в магазин поступает новая партия бройлеров, которую можно представить в виде списка весов тушек. Прежде владелец для каждого покупателя взвешивал каждую доступную тушку и выбирал из них максимальную по весу. Со временем он понял, что это неоптимальный вариант.

Помогите владельцу с решением данной проблемы путем выбора и реализации оптимальной структуры данных для хранения весов тушек.

**2. ИССЛЕДОВАНИЕ**

Для решения поставленной задачи необходимо выбрать оптимальную структуру данных. Первичный отбор среди всех структур данных осуществлялся по двум критериям, сформулированным из условия задачи: быстрая вставка информации о студенте (здесь и далее имеется в виду абитуриент) и удобство в получении данных по убыванию баллов. В результате отсеивания наименее оптимальных для задачи структур, осталось две: макс-куча и оптимизированный двумерный массив. Исследование этих двух структур показало, что вставка в макс-кучу имеет временную сложность алгоритма вставки порядка O(logn), а получение данных по убыванию является простой задачей – необходимо получать данные из корня макс-кучи. В массив же вставка выполняется за O(1), но получение данных по убыванию является чуть более сложной задачей, чем для макс-кучи. Было решено использовать оптимизированный двумерный массив, так как в условии задачи четко указано, что необходима быстрая вставка и наличие возможности получения данных по убыванию, то есть, скорость последнего не важна. Стоит отметить, что под «оптимизированным двумерным массивом» в данном случае понимается «хэш-таблица с открытой адресацией (ключ – количество баллов студента) и методом цепочек для разрешения коллизий **[1]**». Однако, другой метод разрешения коллизий для структуры будет неоптимальным по времени выполнения (поскольку число студентов с одинаковым числом баллов может быть достаточно большим), ведь в таком случае будет необходимо часто расширять хэш-таблицу. В связи с этим, такую структуру я назвал «оптимизированный двумерный массив». Его специфика такова: есть массив, который отвечает за хранение массивов студентов. Каждая ячейка «внешнего» массива хранит массив студентов с конкретным числом баллов. Если есть студент с определенным числом баллов, ячейка «внешнего» массива инициализируется, в противном случае остается неинициализированной (None). При добавлении студента в структуру, осуществляется доступ к массиву студентов с конкретным числом баллов, и уже в него добавляется студент. Под этим и можно понимать «оптимизацию», поскольку место под «внутренние» массивы не выделяется при создании структуры.

**3. ОПИСАНИЕ РЕАЛИЗОВАННОЙ ПРОГРАММЫ**

**3.1. Описание структуры данных**

Класс Student представляет собой сущность студента. Каждый студент имеет имя и количество баллов. При создании экземпляра класса осуществляется проверка имени и количества баллов на корректность (имя должно быть непустым, а количество баллов – неотрицательным).

Описание полей:

* name – имя студента
* score – количество баллов студента

Класс StudentArray представляет реализованную по заданию структуру данных. Это двумерный оптимизированный массив, о котором говорилось в Исследовании. В конструкторе принимает максимальное количество баллов, которое может быть у студента. Стоит отметить, что в любом случае максимальное число баллов есть у списка абитуриентов на каждое направление. Введение этого параметра позволяет оптимизировать структуру и не тратить время на ее расширение. В случае, когда максимальное количество баллов меньше 1 (если оно 0, то в таком случае в структуре нет смысла), или когда при вставке студента в структуру его количество баллов больше максимального (некорректный аргумент), выбрасывается ошибка.

Описание полей:

* \_\_base – «внешний» массив, который содержит массивы студентов
* \_\_current\_index – индекс текущего студента. Инициализируется при создании итератора
* \_\_current\_unit – индекс текущей ячейки (внутреннего массива). Инициализируется при создании итератора

Описание методов:

* append – добавляет студента в массив. Если ячейка с массивом студентов с данным количеством баллов не была инициализирована, то метод инициализирует ее
* \_\_find\_nearest\_unit – вспомогательный метод для итерирования по структуре. Смещает \_\_current\_unit и \_\_current\_index на следующего студента
* \_\_iter\_\_ – инициализирует итератор по структуре (в порядке убывания баллов студентов)
* \_\_next\_\_ – возвращает следующего студента по ходу итерации. Выбрасывает StopIteration при достижении конца структуры

**3.2. Описание тестов**

Для тестирования корректности работы структуры созданы юнит-тесты, проверяющие различные сценарии работы со структурой, в том числе «крайние случаи». Например, проверяется выбрасывание ошибки при различных некорректных аргументах, вывод студентов в порядке убывания, а также поведение пустой структуры. Используется библиотека pytest, запуск тестов выполняется командой «pytest tests.py».

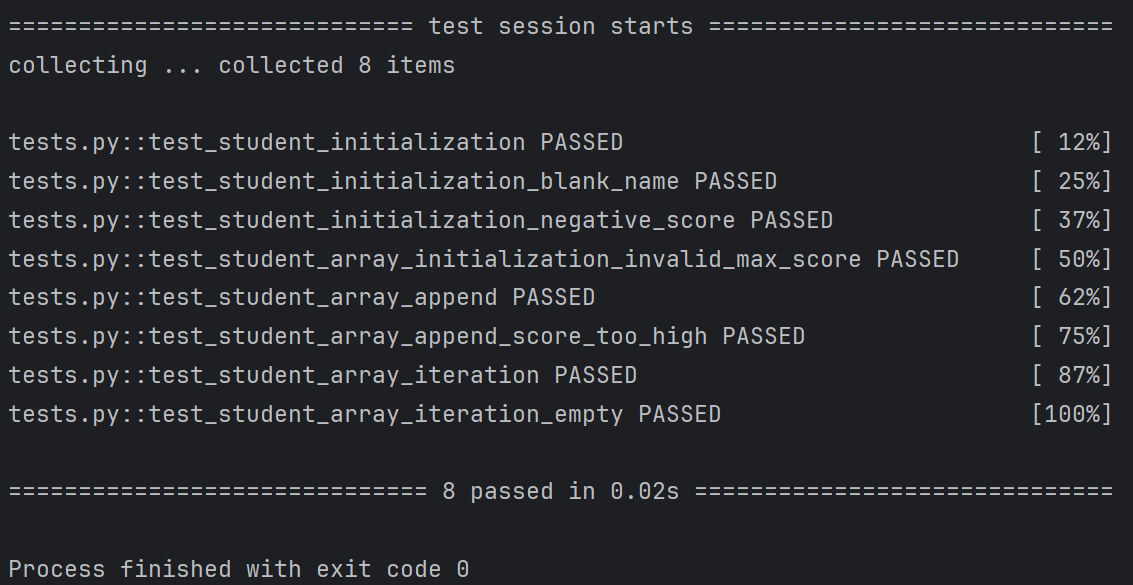


Рисунок 1 – Результат выполнения тестов

**3.3. Описание главной функции**

Для демонстрации взаимодействия со структурой создан небольшой интерфейс командной строки. Создается экземпляр структуры с выбранным количеством максимальных баллов. Делается запрос команды у пользователя. Доступные команды: help (вывести вспомогательное сообщение), append [name] [score] (добавить студента с именем name и score баллами), list (напечатать данные о студентах в порядке убывания баллов студентов), visualize (визуализировать структуру) и exit (выйти из программы). Запускается цикл, который работает до тех пор, пока команда не станет exit.

**3.4. Описание функции визуализации**

Создана функция визуализации структуры с помощью библиотеки Graphviz.

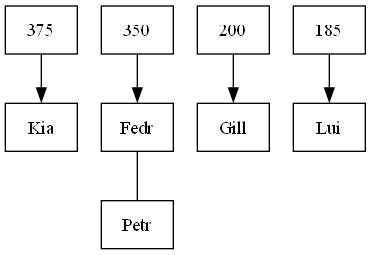


Рисунок 2 – Пример визуализации структуры

На рисунке блоки сверху показывают число баллов студентов в «колонке». От каждого блока с числом идет стрелка на блоки со студентами. Каждый блок со студентом содержит имя конкретного студента. Блоки со студентами в колонке связаны стрелками без наконечников. Каждая «колонка» содержит студентов только с конкретным числом баллов.

Полный разработанный программный код (структуры, главной функции, тестов и визуализации) см. в Приложении А.

**4. Анализ полученных РЕЗУЛЬТАТОВ**

Для исследования скорости работы структуры было измерено время работы операции вставки на различных объемах данных. Сводные результаты измерений представлены в Таблице 1. Все значения времени в таблице представлены в секундах.

Таблица 1 – Результаты исследования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Операция | Размер выборки | | | | |
| 100 | 1000 | 10000 | 100000 | 1000000 |
| Вставка | 0.000000210 | 0.000000178 | 0.000000210 | 0.000000192 | 0.000000176 |

Как видно из данных Таблицы 1, операция вставки студента в структуру при любом из представленных объемов данных занимает менее, чем 10-6 секунды, что позволяет сделать вывод о подтверждении теоретических данных и заключить, что вставка структуру выполняется за O(1) по времени. Это подтверждает заявленную в теоретическом исследовании гипотезу о эффективности структуры для решения задачи, поставленной в рамках варианта курсовой работы.

**заключение**

Работа успешно завершена. Поставленные цели успешно достигнуты, а задачи – выполнены. Результаты практических измерений времени работы структуры подтвердили ее заявленную эффективность. В процессе разработки был реализован весь необходимый функционал в соответствии с требованиями курсовой работы. Кроме того, создан простой интерфейс командной строки для демонстрации работы структуры, реализована функция визуализации структуры, написаны и выполнены юнит-тесты.

**список использованных источников**

1. Разрешение коллизий // IFMO URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5\_%D0%BA%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D0%B9 (дата обращения 12.12.2024) – Текст : электронный.

**приложение А**

**ИСХОДНЫЙ КОД ПРОГРАММ**