Домашнее задание

Требуется реализовать “словарь” (динамический ассоциативный массив) - структуру, позволяющую добавлять и искать элементы по ключу - с помощью указанного в задании алгоритма, язык программирования по выбору. Программа должна загружать записи из входного файла, добавлять их в словарь по указанному ключу, затем производить поиск записей в словаре по ключам (ключи для поиска также могут загружаться из отдельного файла; желательно наличие как присутствующих, так и отсутствующих в словаре ключей).

Требуется замерить время добавления всех записей в словарь и время поиска записей в словаре по заданным ключам. Полученное время сравнить со временем работы с аналогичной стандартной структурой в выбранном языке программирования (в C++: std::map/std::multimap для бинарных деревьев поиска и std::unordered_map/ std::unordered_multimap для хэш-таблиц). Дополнительно время поиска сравнить со временем поиска тех же данных в линейном массиве/списке (т.е. когда словарь реализован как линейный массив или список пар “ключ-значение”). В качестве входных файлов использовать файлы из вспомогательных материалов.

Для бинарных деревьев поиска также найти высоту получившегося дерева, для хэш-таблицы со списками - минимальную, максимальную и среднюю длину списка, для хэш-таблицы с прямой адресацией - процент заполненности таблицы и минимальное, максимальное и среднее число проверок при поиске элементов.

Формат записей во входном файле (по одной записи на строку): `Имя Фамилия Возраст`. Имя, фамилия - строка, возраст - число от 1 до 100.

При вставке нового элемента, если словарь допускает элементы с совпадающими ключами и в словаре уже есть элемент с таким ключом, в словаре остаются оба элемента. Поиск по ключу в этом случае находит первый совпавший элемент. Иначе, если элементы с совпадающими ключами не допускаются, новый элемент замещает старый.

В отчет включить:

- постановку задачи
- описание использованных алгоритмов и структур данных
- результаты тестов (таблицы, графики)
- выводы
- текст программы включать в отчет не обязательно, но требуется показать работающую программу на занятии

Вспомогательные материалы

Вспомогательные материалы можно использовать как основу для выполнения задания. Они находятся в репозитории https://gitlab.com/georgy-schukin/mpiaa, директория searching.

В файлах .h и .cpp находится исходный код тестового примера на языке C++. Для выполнения задания нужно заменить стандартную реализацию словаря (std::map) и функции doBuild и doSearch на свои собственные.

Для генерации входных файлов нужно запустить скрипт inputgen.py (должен быть установлен язык программирования Python). Для генерации ключей для поиска нужно запустить скрипт keygen.py.

В случае использования ключа, отличного от ключа в примере (фамилия), нужно изменить реализацию метода makeKey класса Person и генерацию ключей в keygen.py.

Тестовый пример принимает на вход два аргумента: имя входного файла с записями и имя файла с ключами для поиска. При отсутствии аргументов будут использованы значения по умолчанию.

Варианты заданий

Номер варианта для выполнения = свой номер в списке группы % количество вариантов

1. Хэш-таблица со связными списками для разрешения коллизий. Ключ: фамилия. Совпадающие ключи не допускаются.
2. Хэш-таблица с открытой адресацией. Ключ: фамилия + первая буква имени. Совпадающие ключи не допускаются.
3. Красно-черное дерево. Ключ: фамилия + возраст. Совпадающие ключи допускаются.
4. 2-3 дерево. Ключ: фамилия. Совпадающие ключи не допускаются.
5. 2-3-4 дерево. Ключ: фамилия + первая буква имени. Совпадающие ключи допускаются.
6. АВЛ-дерево. Ключ: фамилия + возраст. Совпадающие ключи допускаются.
7. Хэш-таблица со связными списками для разрешения коллизий. Ключ: фамилия + первая буква имени. Совпадающие ключи допускаются.

8. Хэш-таблица с открытой адресацией. Ключ: фамилия. Совпадающие ключи допускаются.
9. Красно-черное дерево. Ключ: фамилия + первая буква имени. Совпадающие ключи не допускаются.
10. 2-3 дерево. Ключ: фамилия + последняя буква имени. Совпадающие ключи допускаются.
11. 2-3-4 дерево. Ключ: фамилия + последняя буква имени. Совпадающие ключи не допускаются.
12. АВЛ-дерево. Ключ: фамилия. Совпадающие ключи допускаются.
13. Хэш-таблица со связными списками для разрешения коллизий. Ключ: фамилия + возраст. Совпадающие ключи не допускаются.
14. Хэш-таблица с открытой адресацией. Ключ: фамилия + последняя буква имени. Совпадающие ключи допускаются.
15. Красно-черное дерево. Ключ: фамилия. Совпадающие ключи допускаются.
16. 2-3 дерево. Ключ: фамилия + возраст. Совпадающие ключи не допускаются.
17. 2-3-4 дерево. Ключ: фамилия. Совпадающие ключи допускаются.
18. АВЛ-дерево. Ключ: фамилия + первая буква имени. Совпадающие ключи не допускаются.
19. Хэш-таблица со связными списками для разрешения коллизий. Ключ: фамилия + последняя буква имени. Совпадающие ключи не допускаются.
20. Хэш-таблица с открытой адресацией. Ключ: фамилия + возраст. Совпадающие ключи не допускаются.
21. Красно-черное дерево. Ключ: фамилия + последняя буква имени. Совпадающие ключи не допускаются.
22. 2-3 дерево. Ключ: фамилия + первая буква имени. Совпадающие ключи допускаются.
23. 2-3-4 дерево. Ключ: фамилия + возраст. Совпадающие ключи не допускаются.
24. АВЛ-дерево. Ключ: фамилия + последняя буква имени. Совпадающие ключи допускаются.

Контрольные вопросы

1. Хэш-таблица со связными списками для разрешения коллизий.
2. Хэш-таблица с открытой адресацией.
3. Хэш-функции. Свойства и примеры.
4. Бинарные деревья поиска. Свойства БДП. Сбалансированные деревья поиска.
5. Красно-черные деревья поиска.
6. 2-3 деревья поиска.
7. Для заданной последовательности из 10-15 чисел или слов построить (используя значение элемента в качестве ключа) обычное бинарное дерево поиска, а также:
    a. красно-черное дерево
    b. 2-3 дерево
    c. 2-3-4 дерево

    d. АВЛ-дерево
8. Для заданной последовательности из 10-15 чисел или слов построить (используя значение элемента в качестве ключа; хэш-функция по выбору):
    a. хэш-таблицу со связными списками
    b. хэш-таблицу с открытой адресацией

Литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: Построение и анализ, 3-е издание:
    - Глава 11, Хеширование и хеш-таблицы
    - Глава 12, Бинарные деревья поиска
    - Глава 13, Красно-черные деревья
2. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы:
    - Глава 4, Основные операторы множеств
    - Глава 5, Специальные методы представления множеств
3. Седжвик Р. Фундаментальные алгоритмы на C++:
    - Часть 4, Поиск, главы 12-14