Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Дискретный анализ»

Студент: А. В. Барсов
Преподаватель: Н. С. Капралов
Группа: М8О-307Б-19
Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №2

Задача: Необходимо создать программную библиотеку, реализующую указанную структуру данных, на основе которой разработать программу-словарь. В словаре каждому ключу, представляющему из себя регистронезависимую последовательность букв английского алфавита длиной не более 256 символов, поставлен в соответствие некоторый номер, от 0 до $2^{64} - 1$.

Программа должна обрабатывать строки входного файла до его окончания. Каждая строка может иметь следующий формат:

+ **word** 34 — добавить слово «word» с номером 34 в словарь. Программа должна вывести строку «OK», если операция прошла успешно, «Exist», если слово уже находится в словаре.

- word — удалить слово «word» из словаря. Программа должна вывести «OK», если слово существовало и было удалено, «NoSuchWord», если слово в словаре не было найдено.

word — найти в словаре слово «word». Программа должна вывести «OK: 34», если слово было найдено; число, которое следует за «OK:» — номер, присвоенный слову при добавлении. В случае, если слово в словаре не было обнаружено, нужно вывести строку «NoSuchWord».

! Save /path/to/file — сохранить словарь в бинарном компактном представлении на диск в файл, указанный параметром команды. В случае успеха, программа должна вывести «OK», в случае неудачи выполнения операции, программа должна вывести описание ошибки (см. ниже).

! Load /path/to/file — загрузить словарь из файла. Предполагается, что файл был ранее подготовлен при помощи команды Save. В случае успеха, программа должна вывести строку «OK», а загруженный словарь должен заменить текущий (с которым происходит работа); в случае неуспеха, должна быть выведена диагностика, а рабочий словарь должен остаться без изменений. Кроме системных ошибок, программа должна корректно обрабатывать случаи несовпадения формата указанного файла и представления данных словаря во внешнем файле.

Для всех операций, в случае возникновения системной ошибки (нехватка памяти, отсутствие прав записи и т.п.), программа должна вывести строку, начинающуюся с «ERROR:» и описывающую на английском языке возникшую ошибку.

Вариант №4: B-tree

1 Описание

Требуется написать реализацию словаря (ассоциативного массива) с помощью Вдерева. Помимо простого добавления, поиска и удаления требуется реализовать запись на диск и чтения с диска всего словаря.

Согласно [3], B-деревом называется структура данных, сбалансированное, сильно ветвистое дерево поиска.

Сбалансированность означает, что длина любых двух путей от корня до листьев различаются не более, чем на единицу.

Ветвистость дерева — свойство каждого узла ссылаться на большое число узломпотомков.

B-дерево обладают следующими свойствами:

1. В каждом узле ключи упорядочены для быстрого доступа к ним. Корень содержит от 1 до $2*t-1$ ключей. Любой другой узел содержит от $t-1$ до $2*t-1$. Здесь t - параметр дерева, не меньший 2.

2. Любой узел, содержащий n элементов вида $K_1, ..., K_n$, имеет $n+1$ потомков. При этом

   (a) Первый потомок содержит числа из интервала $(-\infty, K_1)$

   (b) Для $2 \leq i \leq n$, i потомок содержит числа из интервала $(K_i - 1, K_i)$

   (c) $n+1$ потомок содержит числа из интервала $(K_n, +\infty)$

3. Глубина всех листьев одинакова.

Свойство 3 означает, что дерево идеально сбалансированно.

Поиск в B-дереве похож на поиск в обычном бинарном дереве поиска. Если ключ содержится в узле, то он найден. Иначе определяем интервал и идём к соответствующему потомку, пока не найдём элемент.

2 Консоль

```
[artem@IdeaPad solution]$ make
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm -c main.cpp
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm main.o -o
solution
[artem@IdeaPad solution]$ cat tests/test15.txt
+ VQLIWVJN 11446046014801225453
-MJDITKHS
vqliwvjn
+ WLNEALJG 16496512296278855712
+ KTUSCZXR 5329757113570982924
LGZOEPPN
-NFIDJQON
+ DASNZYXQ 87382904958914278 28
vqliwvjn
RIVSWIGB
+ OIYBELBR 2561675325280022408
-SKCFWQTM
+ YXYJSFNQ 3424789100534423791
+ AVQAOHIT 8248868615972596447
+ EEUGEMPB 17894478951972822102
[artem@IdeaPad solution]$ ./solution <tests/test15.txt
OK
NoSuchWord
OK: 11446046014801225453
OK
OK
NoSuchWord
NoSuchWord
OK
OK: 11446046014801225453
NoSuchWord
OK
NoSuchWord
OK
OK
OK
```

3 Выводы

В ходе выполнения лабораторной работы я вспомнил прицип построения бинарного дерева поиска, изучил и реализовал структуру данных B-дерево.

Основной трудностью была работа с указателями, справиться с этим мне помогла утилита Valgrind.

Вызвало затруднение запись дерева в файл, потому что B-дерево нельзя так же просто индексировать, как бинарное дерево поиска (id — индекс вершины, $2 * id$ — индекс левого потомка, $2 * id + 1$ — индекс правого потомка). Я решил индексировать B-дерево, считая его полным. Такая программа работала очень долго, когда она записывала в файл всю информацию о пустом узле. Но после оптимизации мне удалось сжать файл с деревом до пары десятков мегабайт.

Список литературы

[1] Дональд Кнут *Искусство программирования.* Перевод и издатели: В. Штонда, Г. Петриковец, А. Орлович.

[2] *Справочник по алгоритмам и структурам данных.*
URL: `https://e-maxx.ru/` .

[3] *B-дерево.* - Википедия
URL: `https://ru.wikipedia.org/wiki/B-дерево` .