Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Дискретный анализ»

Студент: Д. С. Пивницкий
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-19

Дата: 01.10.2020
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №3

Задача: Для реализации словаря из предыдущей лабораторной работы необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочётов, требуется их исправить.

Результатом лабораторной работы является отчёт, состоящий из:

Дневника выполнения работы, в котором отражено что и когда делалось, какие средства использовались и какие результаты были достигнуты на каждом шаге выполнения лабораторной работы.

Выводов о найденных недочётов.

Сравнение работы исправленной программы с предыдущей версии.

Общих выводов о выполнении лабораторной работы, полученном опыте.

1 Описание методов отладки

Для проверки корректности работы программы и отыскания утечек памяти я использовал утилиту valgrind.

Если необхоимо было выполнить один тест, то команда выглядела следующим образом:

```
(py37) ~ /DA/lab2 valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes
--verbose --log-file=valgrind-out.txt ./lab2ex <test2 && tail -n5 valgrind-out.txt
```

Уже в ходе отладкки я, осознав небходимость в собственном генераторе тестов, создал таковой, и лишь благодарая ему смог отыскать и исправить все (регистрируемые чекером) ошибки в программе.

Созданный для проверки работы прогрмаммы генератор тестов работает следующим образом:

На первом этапе создется три строки со случайными именами файлов, в которые будет производится схранения дерева.

На втором - выполняется генерация первого тестового файла. В нем 50доступа являются командами добавления, остальные – команды поиска и удаления. Таким образом, размер дерева наращивается. Все слова являются простыи комбинациями из 2-3 символов, чтобы обеспечить частые повторения. Периодически в последовательность команд добавляются операции загрузки и сохранения дерева в один из трех файлов.

На третьем этапе генерируется схожий тестовый файл, однако распределение команд доступа в нем иное: 75не происходит. Таким образом при достаточной длине файла размер дерева медленно уменьшается до 0. При этом не перестают выполняться периодические сохранения и загрузки.

Для работы с этим тестером я создал дополнительный bash-скрипт, который выполняет генерацию 30 тестов и проверку корректности работы прогрммы. В этой реализации генератор состоит из двух программ, каждая из которых генерирует один файл с тестами.

```
1 #!/bin/bash
2 for ((i=1;i<= 30;i++))
3 do
4 ./gen.out && ./gen-2.out && valgrind --leak-check=full --show-leak-kinds=all --track-
    origins=yes --verbose --log-file=valgrind-out.txt ./lab2ex < test_of_absolutia >
    res-test&& tail -n5 valgrind-out.txt && valgrind --leak-check=full --show-leak-
    kinds=all --track-origins=yes --verbose --log-file=valgrind-out.txt ./lab2ex <
    second_test_of_absolutia > res-test && tail -n5 valgrind-out.txt
5 done
```

2 Дневник разработки

Дневник содержит в себе хронологию основных действий по отладке программы и список найденных ошибок

Дневник разработки		
Дата	Действие	Коментарий
20.01.2021	Реализаяция и отладка основных компонентов программы	В этот промежуток времени программа еще не работоспособна, производится содание классов дерева и узла, разработка парсера команд.
23.01.2021	Предварительная отладка	Создается простенькая программа, конвертирующая английйский текст в последовательность команд (на добавление, удаление и поиск слов). Ошибок не найдено.
23.01.2021	Проверка на предельных значениях	Производится ручная проверка на предельных длинах слов и значениях поля value. Ошибок не найдено.
25.01.2021	Первый залив на чекер	Наблюдается ошибка выполнения в тесте 4
28.01.2021	Обнаружена ошибка в алгоритме поиска узла в дереве	Алгоритм поиска не учитывает возможность того что дерево пусто. После исправления программа начинает проходить тест 4 почти до конца, однако в итоге все равно ломается.
01.02.2021	Выполнена существенная оптимизация	В ходе отладки я решил заменить используемый до этого контейнер std::string на char*, отчего скорость работы программы выпросла почти вдвое.

02.02.2021	Выполнена существенная оптимизация	Я заменил инструмент вывода на экран с std::cout на printf, отчего скорость работы программы увеличилась в полтора раза.
07.02.2021	Создание своего генератора тестов	Ввиду того что все предыдущие попытки отыскания ошибок не увенчались успехом, принимаю решение создать собственный генератор тестов, который сможет смоделировать большую часть допустимых последовательностей ввода.

08.02.2021	Обнаружена ошибка в алгоритме балансировки после удаления элемента	В ходе работы со сгенерированными тестами обнаружено, что несмотря на то что программа всегда корректно завершается, приблизительно в 30случаев valgrind регистрирует обширную утечку памяти, по общему виду напоминающую утерю целой ветви дерева. При последующем анализе программы было обнаружено, что в определенных ситуациях при выполнении перебалансировки после удаления элемента не срабатывает рекурсивный вызов из-за чего функция возвращает в качестве нового корня элемент в центре дерева, а остальная часть структуры теряется. Ошибка исправлена. После исправления «ошибка выполнения на тесте 4» сменилась «на неправильный ответ на тесте 4».

08.02.2021	Обнаружена ошибка в алгоритме поиска узла в дереве	Так как для проверки равенства строк использовалась собственная функция strequal(), не учитывающая регистр букв в строках, а для, непосредственно, сравнения использовалась функция strcmp(), учитывающая регистр, порой strcmp() пускала поиск по неверному пути, отчего результат не соответствовал истине. Для обнаружения ошибки не потребовалось использование каких-либо специальных средств, Она была найдена в ходе мысленного анализа результатов выполнения тестов чекера. Ошибка исправлена путем переписывания strequal() таким образом, чтобы она покрывала функционал strcmp(). После исправления иных ошибок чекером обнаружено не было.

3 Выводы

Пожалуй, основной вывод, который я сделал за время отладки программы для ЛР2 заключается в том, что вместо того чтобы две недели пытаться найти ошибку различными альтернативными путями лучше сразу написать тестер, ибо рано или поздно делть его все равно придется. К тому же, я заметил у себя склонность к поиску сложных и неочевидных ошибок, при том что, как нетрудно заметить, моя программа содержит исключительно до предела простые ошибки, и то что я за время отладки насквозь пронизал ее путями выброса исключений о каждом несработавшем внутри new malloc-е никак в итоге не помогло. В ходе выполнения этой лабораторной работы я также поближе познакомился с программой valgrind и научился более ли менее успешно отыскивать утечки памяти. Так как за время поиска олибок выполнения, я заодно натолько, насколько мог оптимизировал работу своей программы, никаких ощутимых проблем связанных со скоростью ее работы я не испытал.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Поразрядная сортировка — Вики университета ITMO.*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Цифровая_сортировка` (дата обращения: 01.10.2020).