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: В. П. Будникова
Преподаватель: А. А. Кухтичев
Группа: М8О-209Б-19

Дата:
Оценка:
Подпись:

Москва, 2020

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.

Используемые интрументы: утилита Valgrind, Xcode Instrument: Time Profiler

1 Описание

Valgrind

Как сказано в [1]: «Valgrind — инструментальное программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти.» Так как при реализации B-Дерева я пользовалась утилитой Valgrind, то в конечной программе у меня не обнаружилось ошибок и утечек памяти, к сожалению, я не сохранила файлы с первоначальным кодом, поэтому искусственно сделаем ошибки. «Забудем» освободить память, при удалении дерева. Входные данные - 100000 строчек с операциями вставки, удаления, поиска, загрузки в файл, выгрузки из файла
Что выдает Valgrind:

```
==4792==
==4792== HEAP SUMMARY:
==4792== in use at exit: 101,261,168 bytes in 32,777 blocks
==4792== total heap usage: 59,559 allocs, 26,782 frees, 209,375,992 bytes allocated
==4792==
==4792== LEAK SUMMARY:
==4792== definitely lost: 38,958,304 bytes in 12,551 blocks
==4792== indirectly lost: 62,272,448 bytes in 20,062 blocks
==4792== possibly lost: 12,416 bytes in 4 blocks
==4792== still reachable: 4,096 bytes in 1 blocks
==4792== suppressed: 13,904 bytes in 159 blocks
==4792== Rerun with --leak-check=full to see details of leaked memory
==4792==
==4792== For lists of detected and suppressed errors, rerun with: -s
==4792== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```

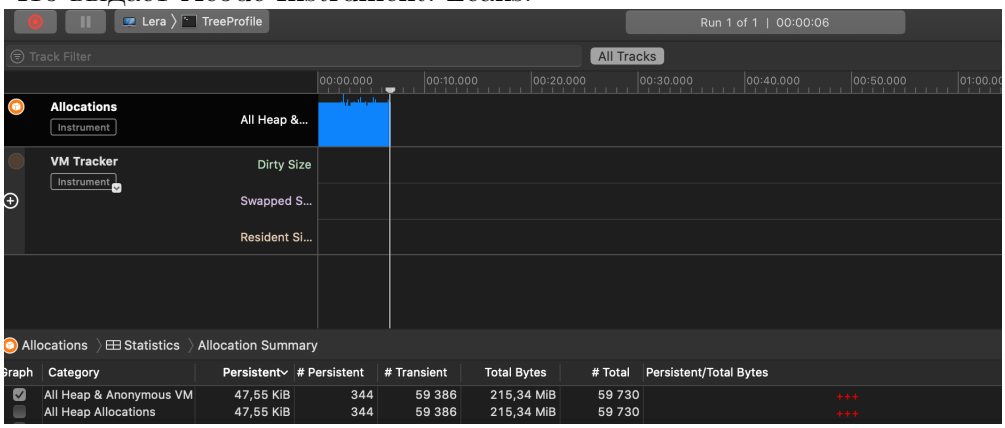
Согласно [2]: «Definitely lost означает, что valgrind нашел область памяти, на которую нет указателей, т.е. программист не освободил память, при выходе указателя за область видимости. Possibly lost показывает, что найден указатель, указывающий на часть области памяти, но valgrind не уверен в том, что указатель на начало области памяти до сих пор существует (это может происходить в тех случаях, когда программист вручную управляет указателями).Still reachable обычно означает, что valgrind нашел указатель на начало не освобожденного блока памяти» Valgrind обнаружил только утечки памяти, так как при работе с деревом, записи, удалении, поиске в моей программе происходит работа не с указателями, а с значением длины массивов, которые автоматически регулируются при добавлении элемента в массив, или удаления из него. Но обнаружились утечки памяти, исправим ошибку.
Что выдает Valgrind:

```
==4858==
==4858== HEAP SUMMARY:
==4858== in use at exit: 18,000 bytes in 160 blocks
```

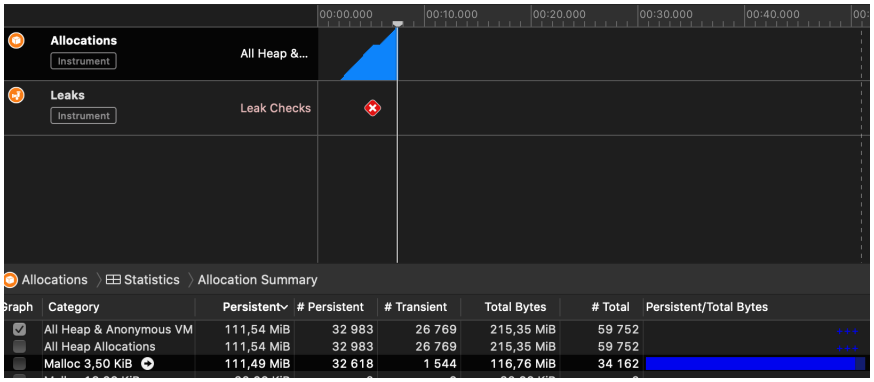
```
==4858== total heap usage: 59,559 allocs, 59,399 frees, 209,375,992 bytes allocated
==4858==
==4858== LEAK SUMMARY:
==4858== definitely lost: 0 bytes in 0 blocks
==4858== indirectly lost: 0 bytes in 0 blocks
==4858== possibly lost: 0 bytes in 0 blocks
==4858== still reachable: 4,096 bytes in 1 blocks
==4858== suppressed: 13,904 bytes in 159 blocks
==4858== Rerun with --leak-check=full to see details of leaked memory
==4858==
==4858== For lists of detected and suppressed errors, rerun with: -s
==4858== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 1 from 1)
```

Но Valgrind все равно показывает утечки памяти. Во время написания лабораторной работы я жолго не понимала, почему происходит такой вывод. я обратилась к одногруппнику за помощью и попросила протестировать его мою программу на Linux, утечек у него не обнаружилось. Проблема в том, что оригинального valgring под mac OC не существует, поэтому я устанавливала кастомный. Попробуем проверить в Xcode Instrument: Leaks.

Что выдает Xcode Instrument: Leaks:

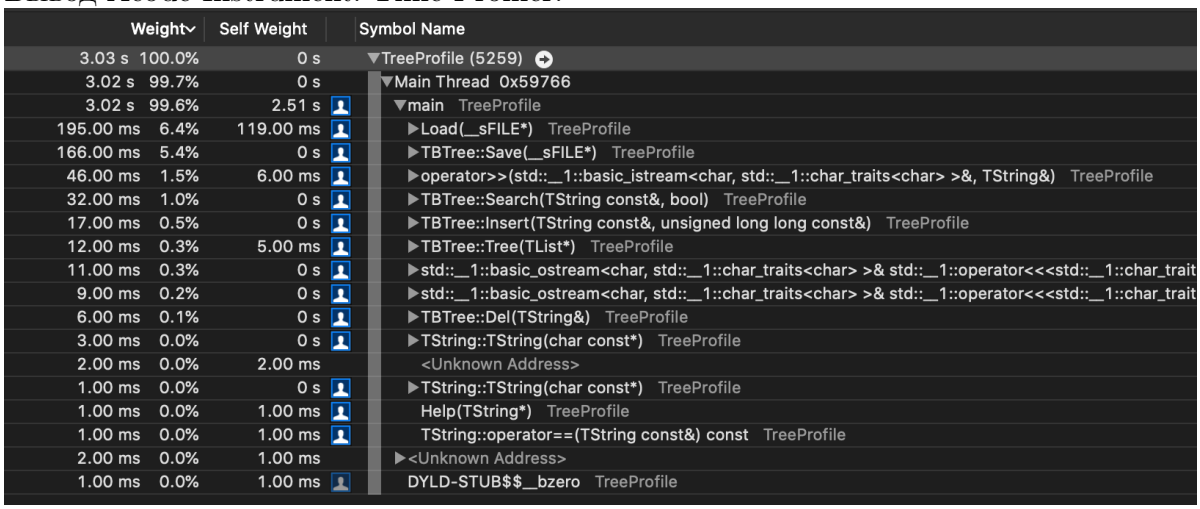


Что выдает Xcode Instrument: Leaks, если не освободить память при удалении дерева:



Xcode Instrument: Time Profiler

Так как утилиты gprof я не обнаружила для mac OC, я решила воспользоваться Xcode Instrument: Time Profiler. Согласно [3] «Time profiler is an important instrument which helps in achieving better performance in iOS applications. It provides details about time consumed by each method in your source code, giving able space to alter the logic applied.» Перевод: Профилировщик времени - важный инструмент, помогающий повысить производительность приложений iOS. Он предоставляет подробную информацию о времени, затраченном на каждый метод в вашем исходном коде, давая возможность изменить применяемую логику.
Вывод Xcode Instrument: Time Profiler:



Weight	Self Weight	Symbol Name
3.03 s 100.0%	0 s	▼TreeProfile (5259)
3.02 s 99.7%	0 s	▼Main Thread 0x59766
3.02 s 99.6%	2.51 s	▼main TreeProfile
195.00 ms 6.4%	119.00 ms	▶Load(__sFILE*) TreeProfile
166.00 ms 5.4%	0 s	▶TBTree::Save(__sFILE*) TreeProfile
46.00 ms 1.5%	6.00 ms	▶operator>>(std::__1::basic_istream<char, std::__1::char_traits<char> >&, TString&) TreeProfile
32.00 ms 1.0%	0 s	▶TBTree::Search(TString const&, bool) TreeProfile
17.00 ms 0.5%	0 s	▶TBTree::Insert(TString const&, unsigned long long const&) TreeProfile
12.00 ms 0.3%	5.00 ms	▶TBTree::Tree(TList*) TreeProfile
11.00 ms 0.3%	0 s	▶std::__1::basic_ostream<char, std::__1::char_traits<char> >& std::__1::operator<<<std::__1::char_trait
9.00 ms 0.2%	0 s	▶std::__1::basic_ostream<char, std::__1::char_traits<char> >& std::__1::operator<<<std::__1::char_trait
6.00 ms 0.1%	0 s	▶TBTree::Del(TString&) TreeProfile
3.00 ms 0.0%	0 s	▶TString::TString(char const*) TreeProfile
2.00 ms 0.0%	2.00 ms	<Unknown Address>
1.00 ms 0.0%	0 s	▶TString::TString(char const*) TreeProfile
1.00 ms 0.0%	1.00 ms	Help(TString*) TreeProfile
1.00 ms 0.0%	1.00 ms	TString::operator==(TString const&) const TreeProfile
2.00 ms 0.0%	1.00 ms	▶<Unknown Address>
1.00 ms 0.0%	1.00 ms	DYLD-STUB$$__bzero TreeProfile

Как можно увидеть, много времени занимает сохранение и выгрузка из файла дерева, это логично, так как запись и чтение с диска занимает достаточное количество времени. Далее идет операторов вывода строк(а точнее моей структуры TString) так как чтобы вывести строку, мы проходится по кажлому ее элементу, а если строка очень длинная, то время будет затрачено много. Поиск занимает больше времени из времени выполнения всей программы, чем вставка, так как в файле было очень много повторяющихся ключей, и следовательно вставка не проходила до конца, а поиск элементов происходит путем обхода всего дерева.

2 Дневник отладки

1. Использовала утилиту Valgrind для отладки программы
2. Познакомилась с инструментом Xcode Instrument: Time Leaks
3. Использовала Time Leaks для проверки на утечки памяти программу
4. Познакомилась с инструментом Xcode Instrument: Time Profiler
5. Использовала Time Profiler для профилирования программы

3 Выводы

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я изучила средства диагностики и профилирования такие как Valgrind, Xcode Instrument: Time Leaks, Xcode Instrument: Time Profiler. К сожалению я потратила очень много времени, чтобы разобраться как профилировать программу на mac OC, но нашла выход в Xcode Instrument. По началу мне было очень непривычно, так как для меня эти инструменты были не знакомы. Но я узнала о очень многих инструментах доступных в Xcode Instrument. Очень интересно еще больше узнать о инструментах Time Leaks и Time Profiler, но времени на это не хватит, по крайней мере в этом семестре из-за большой загруженности.

Список литературы

[1] *Valgrind — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Valgrind` (дата обращения: 09.12.2020).

[2] *Valgrind*
URL: `https://www.opennet.ru/base/dev/valgrind_memory.txt.html` (дата обращения: 09.12.2020).

[3] *How to Time Profile*
URL: `https://www.bittudavis.com/post/time-profiler` (дата обращения: 09.12.2020).