Лабораторная работа № 3 по курсу дискретного анализа: Исследование качества программ

Выполнил студент группы М8О-207Б-21 Меркулов Фёдор.

Условие

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.

Дневник выполнения работы

В данной работе я решил воспользоваться утилитами gprof и valgrind.

В начале запущу программу с утилитой gprof. Эта утилита помогает увидеть время работы всех используемых, а также она подсчитывает количество их вызовов и процентное соотношение работы конкретной функции от работы всей программы.

Скомпилируем код с помощью команды g++ main.cpp -pg -o main. -pg флаг, позволя.щий выводить необходимые данные в специальный файл gmon.out.

Сгенерируем файл t.txt, состоящий из 1000000 команд на поиск, вставку и удаление.

Запустим ./main < t.txt.

После выполнения просмотрим gmon.out как gprof main

```
index % time    self  children    called     name
                                <spontaneous>
[1]     99.9    0.04    9.32                 main [1]
                0.00    4.00 1330270/1330270     Tree::Search(String const&) [2]
                2.58    0.00 1670542/1670542     operator>>(std::istream&, String&) [6]
                0.01    1.72  340272/340272      Tree::Delete(String const&) [7]
                0.01    0.94  329459/329459      Tree::Insert(Element&) [10]
                0.00    0.04       1/1           Tree::~Tree() [17]
                0.01    0.01 2000002/2787502     Element::Element() [23]
                0.00    0.00       1/1           Tree::Tree(int) [29]
                0.00    0.00       1/2787503     String::String() [26]
                0.00    0.00 2000002/2787502     Element::~Element() [37]
                0.00    0.00       1/2787503     String::~String() [36]
-----------------------------------------------
                0.00    4.00 1330270/1330270     main [1]
[2]     42.7    0.00    4.00 1330270         Tree::Search(String const&) [2]
                1.42    2.58 1330270/1330270     TreeNode::Search(String const&) [3]
-----------------------------------------------
```

```
                9899035           TreeNode::Search(String const&) [3]
          1.42    2.58 1330270/1330270     Tree::Search(String const&) [2]
[3]    42.7    1.42    2.58 1330270+9899035 TreeNode::Search(String const&) [3]
          0.11    1.60 18691098/31938470     operator<(String const&, String const&) [5]
          0.03    0.84 9750885/11909335     operator==(String const&, String const&) [9]
                9899035           TreeNode::Search(String const&) [3]
-----------------------------------------------
          0.00    0.00   16142/43863947     operator>(String const&, String const&) [28]
          1.02    0.00 11909335/43863947     operator==(String const&, String const&) [9]
          2.74    0.00 31938470/43863947     operator<(String const&, String const&) [5]
[4]    40.1    3.76    0.00 43863947         String::Lexis(char const*, char const*) [4]
-----------------------------------------------
          0.04    0.54 6269601/31938470     TreeNode::InsertForNoFull(Element const&) [11]
          0.04    0.60 6977771/31938470     TreeNode::Delete(String const&) [8]
          0.11    1.60 18691098/31938470     TreeNode::Search(String const&) [3]
[5]    31.3    0.19    2.74 31938470         operator<(String const&, String const&) [5]
          2.74    0.00 31938470/43863947     String::Lexis(char const*, char const*) [4]
-----------------------------------------------
          2.58    0.00 1670542/1670542     main [1]
[6]    27.5    2.58    0.00 1670542         operator>>(std::istream&, String&) [6]
-----------------------------------------------
          0.01    1.72  340272/340272      main [1]
[7]    18.5    0.01    1.72  340272         Tree::Delete(String const&) [7]
          0.67    1.05  340272/340272      TreeNode::Delete(String const&) [8]
          0.00    0.00   16134/157500      TreeNode::~TreeNode() [22]
-----------------------------------------------
                2536364           TreeNode::Delete(String const&) [8]
          0.67    1.05  340272/340272      Tree::Delete(String const&) [7]
[8]    18.3    0.67    1.05  340272+2536364 TreeNode::Delete(String const&) [8]
          0.04    0.60 6977771/31938470     operator<(String const&, String const&) [5]
          0.10    0.12  305629/305629      TreeNode::Balance(int) [14]
          0.01    0.19 2158450/11909335     operator==(String const&, String const&) [9]
                2536364           TreeNode::Delete(String const&) [8]
-----------------------------------------------
          0.01    0.19 2158450/11909335     TreeNode::Delete(String const&) [8]
          0.03    0.84 9750885/11909335     TreeNode::Search(String const&) [3]
[9]    11.3    0.04    1.02 11909335         operator==(String const&, String const&) [9]
```

```
          1.02    0.00 11909335/43863947     String::Lexis(char const*, char const*) [4]
-----------------------------------------------
          0.01    0.94  329459/329459      main [1]
[10]    10.2    0.01    0.94  329459       Tree::Insert(Element&) [10]
          0.09    0.84  329459/329459      TreeNode::InsertForNoFull(Element const&) [11]
          0.00    0.00   16142/141357      TreeNode::SplitChild(int) [19]
          0.00    0.00   16142/157500      TreeNode::TreeNode(int, bool) [24]
          0.00    0.00   16142/16142       operator>(String const&, String const&) [28]
-----------------------------------------------
                    2456439           TreeNode::InsertForNoFull(Element const&) [11]
          0.09    0.84  329459/329459      Tree::Insert(Element&) [10]
[11]    10.0    0.09    0.84  329459+2456439 TreeNode::InsertForNoFull(Element const&) [11]
          0.04    0.54 6269601/31938470     operator<(String const&, String const&) [5]
          0.00    0.23  329459/329459      String::operator=(String const&) [13]
          0.02    0.02  125215/141357      TreeNode::SplitChild(int) [19]
          0.00    0.00  478714/2325124     Element::operator=(Element&) [21]
                    2456439           TreeNode::InsertForNoFull(Element const&) [11]
-----------------------------------------------
          0.23    0.00  329459/329459      String::operator=(String const&) [13]
[12]     2.5    0.23    0.00  329459       String::Copy(char*, char const*) [12]
-----------------------------------------------
          0.00    0.23  329459/329459      TreeNode::InsertForNoFull(Element const&) [11]
[13]     2.5    0.00    0.23  329459       String::operator=(String const&) [13]
          0.23    0.00  329459/329459      String::Copy(char*, char const*) [12]
-----------------------------------------------
          0.10    0.12  305629/305629      TreeNode::Delete(String const&) [8]
[14]     2.3    0.10    0.12  305629       TreeNode::Balance(int) [14]
          0.05    0.01  157950/157950      TreeNode::GetFromLeftChild(int) [15]
          0.04    0.01   41919/41919       TreeNode::Merge(int) [16]
          0.01    0.00  105760/105760      TreeNode::GetFromRightChild(int) [25]
-----------------------------------------------
          0.05    0.01  157950/157950      TreeNode::Balance(int) [14]
[15]     0.6    0.05    0.01  157950       TreeNode::GetFromLeftChild(int) [15]
          0.00    0.01  631800/2325124     Element::operator=(Element&) [21]
-----------------------------------------------
          0.04    0.01   41919/41919       TreeNode::Balance(int) [14]
[16]     0.5    0.04    0.01   41919       TreeNode::Merge(int) [16]
```

```
          0.01    0.00   41919/157500      TreeNode::~TreeNode() [22]
          0.00    0.00  164672/2325124     Element::operator=(Element&) [21]
-----------------------------------------------
          0.00    0.04       1/1           main [1]
[17]    0.5   0.00    0.04       1         Tree::~Tree() [17]
          0.03    0.01       1/1           TreeNode::DeleteNode() [18]
-----------------------------------------------
                    99446           TreeNode::DeleteNode() [18]
          0.03    0.01       1/1           Tree::~Tree() [17]
[18]    0.5   0.03    0.01      1+99446    TreeNode::DeleteNode() [18]
          0.01    0.00   99447/157500      TreeNode::~TreeNode() [22]
                    99446           TreeNode::DeleteNode() [18]
-----------------------------------------------
          0.00    0.00   16142/141357      Tree::Insert(Element&) [10]
          0.02    0.02  125215/141357      TreeNode::InsertForNoFull(Element const&) [11]
[19]    0.4   0.02    0.02  141357        TreeNode::SplitChild(int) [19]
          0.01    0.01  141357/157500      TreeNode::TreeNode(int, bool) [24]
          0.00    0.01  611751/2325124     Element::operator=(Element&) [21]
-----------------------------------------------
          0.02    0.00 2325124/2325124     Element::operator=(Element&) [21]
[20]    0.2   0.02    0.00 2325124        String::Swap(String&) [20]
-----------------------------------------------
          0.00    0.00  164672/2325124     TreeNode::Merge(int) [16]
          0.00    0.00  438187/2325124     TreeNode::GetFromRightChild(int) [25]
          0.00    0.00  478714/2325124     TreeNode::InsertForNoFull(Element const&) [11]
          0.00    0.01  611751/2325124     TreeNode::SplitChild(int) [19]
          0.00    0.01  631800/2325124     TreeNode::GetFromLeftChild(int) [15]
[21]    0.2   0.00    0.02 2325124        Element::operator=(Element&) [21]
          0.02    0.00 2325124/2325124     String::Swap(String&) [20]
-----------------------------------------------
          0.00    0.00   16134/157500      Tree::Delete(String const&) [7]
          0.01    0.00   41919/157500      TreeNode::Merge(int) [16]
          0.01    0.00   99447/157500      TreeNode::DeleteNode() [18]
[22]    0.2   0.02    0.00  157500        TreeNode::~TreeNode() [22]
          0.00    0.00  787500/2787502     Element::~Element() [37]
-----------------------------------------------
          0.00    0.00  787500/2787502     TreeNode::TreeNode(int, bool) [24]
```

```
          0.01    0.01 2000002/2787502     main [1]
[23]    0.2   0.01    0.01 2787502       Element::Element() [23]
          0.01    0.00 2787502/2787503     String::String() [26]
-----------------------------------------------
          0.00    0.00       1/157500      Tree::Tree(int) [29]
          0.00    0.00   16142/157500      Tree::Insert(Element&) [10]
          0.01    0.01  141357/157500      TreeNode::SplitChild(int) [19]
[24]    0.2   0.01    0.01  157500       TreeNode::TreeNode(int, bool) [24]
          0.00    0.00  787500/2787502     Element::Element() [23]
-----------------------------------------------
          0.01    0.00  105760/105760      TreeNode::Balance(int) [14]
[25]    0.1   0.01    0.00  105760       TreeNode::GetFromRightChild(int) [25]
          0.00    0.00  438187/2325124     Element::operator=(Element&) [21]
-----------------------------------------------
          0.00    0.00       1/2787503     main [1]
          0.01    0.00 2787502/2787503     Element::Element() [23]
[26]    0.1   0.01    0.00 2787503       String::String() [26]
-----------------------------------------------
                                <spontaneous>
[27]    0.1   0.01    0.00             String::String(char const*) [27]
-----------------------------------------------
          0.00    0.00   16142/16142       Tree::Insert(Element&) [10]
[28]    0.0   0.00    0.00   16142       operator>(String const&, String const&) [28]
          0.00    0.00   16142/43863947     String::Lexis(char const*, char const*) [4]
-----------------------------------------------
          0.00    0.00       1/1         main [1]
[29]    0.0   0.00    0.00       1       Tree::Tree(int) [29]
          0.00    0.00       1/157500      TreeNode::TreeNode(int, bool) [24]
-----------------------------------------------
          0.00    0.00       1/2787503     main [1]
          0.00    0.00 2787502/2787503     Element::~Element() [37]
[36]    0.0   0.00    0.00 2787503       String::~String() [36]
-----------------------------------------------
          0.00    0.00  787500/2787502     TreeNode::~TreeNode() [22]
          0.00    0.00 2000002/2787502     main [1]
[37]    0.0   0.00    0.00 2787502       Element::~Element() [37]
          0.00    0.00 2787502/2787503     String::~String() [36]
```

```
-----------------------------------------------

           0.00    0.00       1/1         __libc_csu_init [54]

[38]     0.0    0.00    0.00       1         _GLOBAL__sub_I_main [38]

           0.00    0.00       1/1         __static_initialization_and_destruction_0(int, int) [39]

-----------------------------------------------

           0.00    0.00       1/1         _GLOBAL__sub_I_main [38]

[39]     0.0    0.00    0.00       1         __static_initialization_and_destruction_0(int, int) [39]
```

Видно, что большую часть времени занимают функции, связанные с работой дерева, например поиск, удаление и вставка, а также работа со строками, в частности лексикографическое сравнение строк.

Теперь запустим программу с утилитой valgrind, которая позволяет онаруживать RE и утечки памяти.

```
papik@papik-VirtualBox:~/DA2$ g++ -g main.cpp -o main

papik@papik-VirtualBox:~/DA2$ valgrind ./main

==5264== Memcheck, a memory error detector

==5264== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.

==5264== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info

==5264== Command: ./main

==5264==

==5264==

==5264== HEAP SUMMARY:

==5264==     in use at exit: 0 bytes in 0 blocks

==5264==   total heap usage: 12 allocs, 12 frees, 73,943 bytes allocated

==5264==

==5264== All heap blocks were freed -- no leaks are possible

==5264==

==5264== For lists of detected and suppressed errors, rerun with: -s

==5264== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Диагностика показала, что программа работает верно с памятью. Было произведено 12 аллокаций памяти из кучи суммой на 73,943 байт, затем ровно столько же освобождено. Также не возникло иных ошибок.

Сравнение работы программы с предыдущей версией

В процессе решения 2 лабораторной работы у меня происходила утечка связанная с использованием строк cin.tie(Null), ios_base::sync_with_stdio(false). Исправил я эту ошибку с помощью valgrind. Вывод для старой реализации:

```
papik@papik-VirtualBox:~/DA2$ g++ -g main.cpp -o main

papik@papik-VirtualBox:~/DA2$ valgrind --leak-check=full --show-leak-kinds=all ./main

==5374== Memcheck, a memory error detector

==5374== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.

==5374== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info

==5374== Command: ./main

==5374==

==5374==

==5374== HEAP SUMMARY:

==5374==     in use at exit: 122,880 bytes in 6 blocks

==5374==   total heap usage: 17 allocs, 11 frees, 195,799 bytes allocated

==5374==

==5374== 8,192 bytes in 1 blocks are still reachable in loss record 1 of 6

==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)

==5374==    by 0x4968D03: std::basic_filebuf<char, std::char_traits<char> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4966AD9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4918AA7: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x109430: main (main.cpp:527)

==5374==

==5374== 8,192 bytes in 1 blocks are still reachable in loss record 2 of 6

==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)

==5374==    by 0x4968D03: std::basic_filebuf<char, std::char_traits<char> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4966AD9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4918AC8: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x109430: main (main.cpp:527)

==5374==

==5374== 8,192 bytes in 1 blocks are still reachable in loss record 3 of 6

==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)

==5374==    by 0x4968D03: std::basic_filebuf<char, std::char_traits<char> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4966AD9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4918AE8: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x109430: main (main.cpp:527)

==5374==

==5374== 32,768 bytes in 1 blocks are still reachable in loss record 4 of 6

==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)

==5374==    by 0x496AB16: std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)

==5374==    by 0x4966CC9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
```

```
==5374==    by 0x4918B5D: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x109430: main (main.cpp:527)
==5374==
==5374== 32,768 bytes in 1 blocks are still reachable in loss record 5 of 6
==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==5374==    by 0x496AB16: std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x4966CC9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x4918B77: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x109430: main (main.cpp:527)
==5374==
==5374== 32,768 bytes in 1 blocks are still reachable in loss record 6 of 6
==5374==    at 0x483C583: operator new[](unsigned long) (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==5374==    by 0x496AB16: std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >::_M_allocate_internal_buffer() (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x4966CC9: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x4918B90: std::ios_base::sync_with_stdio(bool) (in /usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.28)
==5374==    by 0x109430: main (main.cpp:527)
==5374==
==5374== LEAK SUMMARY:
==5374==    definitely lost: 0 bytes in 0 blocks
==5374==    indirectly lost: 0 bytes in 0 blocks
==5374==      possibly lost: 0 bytes in 0 blocks
==5374==    still reachable: 122,880 bytes in 6 blocks
==5374==         suppressed: 0 bytes in 0 blocks
==5374==
==5374== For lists of detected and suppressed errors, rerun with: -s
==5374== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Выводы

В процессе выполнения третей лабораторной работы по курсу Дискретный анализ, я изучил и применил средства диагностики и профилирования, в частности gprof и valgrind. В связи с тем, что они популярны, становится ясно, что они нередко используются на практике. Valgrind помогает найти утечки памяти и места получения RE. Gprof позволяет узнать, какие функции дольше всего выполняет программа (в моём случае это поиск в дереве и лексикографическое сравнение строк). Эти утилиты хорошо помогают оптимизировать код, как по времени, так и по памяти. Учитывая, что это основные ресурсы компьютера, то эти утилиты являются хорошей опорой в улучшении качества работы и быстродействия программы.