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

Выполнил студент группы 08-207 МАИ *Лебедев Иван.*

Условие

1. 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 отладки

1. gprof

Для профиляции программы утилитой gprof необходимо скомпилировать её с флагом -pg, затем запустить исполняемый файл с какими- либо тестами, в результате чего создаётся файл gmon.out с данными о времени работы функций программы. Эти данные можно представить в виде таблицы с помощью самой утилиты gprof.

На тестах в 200 000 строк я получаются следующие результаты:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 16.69      0.06     0.06 24669945     0.00     0.00  TNode::Height()
 16.69      0.12     0.06  3857705     0.02     0.02  TNode::FixHeight()
 16.69      0.18     0.06   106239     0.57     1.49  TAVLtree::Insert(TN
 13.91      0.23     0.05   106240     0.47     1.43  TAVLtree::DeleteNod
 11.13      0.27     0.04                             main
  8.35      0.30     0.03  6548415     0.00     0.01  TNode::Balance()
  5.56      0.32     0.02    58003     0.35     0.40  TAVLtree::DeleteMin(
  2.78      0.33     0.01  3213775     0.00     0.05  TAVLtree::Balancing(
  2.78      0.34     0.01   162650     0.06     0.11  TAVLtree::TurnLeft(T
  2.78      0.35     0.01   106239     0.09     0.09  TNode::TNode(char c
  2.78      0.36     0.01    29647     0.34     0.34  TAVLtree::Search(TN
  0.00      0.36     0.00   159315     0.00     0.05  TAVLtree::TurnRight(
```

Если запустить программу с тестом в 400 000 строк то результаты будут следующими:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 33.38      0.30     0.30 43602602     0.00     0.00  TNode::Height()
 16.69      0.45     0.15   215697     0.00     0.00  TAVLtree::Insert(TNo
 12.24      0.56     0.11                             main
  9.46      0.65     0.09    60547     0.00     0.00  TAVLtree::Search(TNo
  7.79      0.72     0.07  6810106     0.00     0.00  TNode::FixHeight()
  7.79      0.79     0.07   129655     0.00     0.00  TAVLtree::DeleteNode
  5.56      0.84     0.05 11586142     0.00     0.00  TNode::Balance()
  3.34      0.87     0.03  5688010     0.00     0.00  TAVLtree::Balancing(
  1.11      0.88     0.01   214877     0.00     0.00  TNode::~TNode()
  1.11      0.89     0.01    65591     0.00     0.00  TAVLtree::DeleteMin(
  1.11      0.90     0.01        1    10.02    13.99  TAVLtree::DeleteTree
  0.56      0.90     0.01    60547     0.00     0.00  TAVLtree::Search(cha
  0.00      0.90     0.00   283203     0.00     0.00  TAVLtree::TurnLeft(T
  0.00      0.90     0.00   277845     0.00     0.00  TAVLtree::TurnRight(
```

По этим данным можно понять, что больше всего времени занимает функция Height(), а также Insert() и Search().

2. valgrind

Valgrind - программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти. Программа запускается с помощью valgrind –leak-check=full –show-leak-kinds=all, и подаются на вход тесты. В результате работы программы можно видеть, что вся выделенная память очищается, и нет утечек и ошибок памяти.

```
==5074== 72,704 bytes in 1 blocks are still reachable in loss record 2 o
==5074==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memchec
linux.so)
==5074==    by 0x4EC3EFF: ??? (in /usr/lib/x86_64-linux-gnu/libstdc++.so
==5074==    by 0x40104E9: call_init.part.0 (dl-init.c:72)
==5074==    by 0x40105FA: call_init (dl-init.c:30)
==5074==    by 0x40105FA: _dl_init (dl-init.c:120)
==5074==    by 0x4000CF9: ??? (in /lib/x86_64-linux-gnu/ld-2.23.so)
==5074==
==5074== LEAK SUMMARY:
==5074==    definitely lost: 0 bytes in 0 blocks
==5074==    indirectly lost: 0 bytes in 0 blocks
==5074==      possibly lost: 0 bytes in 0 blocks
==5074==    still reachable: 81,676 bytes in 2 blocks
==5074==         suppressed: 0 bytes in 0 blocks
==5074==
==5074== For counts of detected and suppressed errors, rerun with: -v
==5074== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Выводы

Утилитой gprof можно выявить места в программе в которых она работает дольше всего, и точечно ускорить её, тем самым не тратя время на и так быстрые функции. Утилитой valgrind можно узнать об утечках памяти, об освобождении уже освобожденной памяти и многих других ошибках. Что упрощает нахождение множества ошибок.