

Задача X. Быстрый поиск

Цель работы – изучение, реализация и использование списков с пропусками.

Сдаваемые файлы:

- SkipList.hpp
- JournalNetActivity.hpp



Постановка задачи:

В рамках «Программы по усовершенствованной модернизации с помощью инновационных инноваций» полицейские решили разработать систему поддержки борьбы с киберпреступностью. Для увеличения оперативности в поиске неуемных кибервредителей необходимо при обнаружении взлома как можно быстрее узнать, кто пользовался маскирующими серверами во время атаки. Систему решено было опробовать сначала в метро, а затем и в масштабах всего города.

При совершении киберпреступления, взломщики, для большей скрытности, пользуются сервером verisicretproxi.com. Чтобы поймать преступника, необходимо найти в журнале список всех пользователей, которые пользовались этим сервером незадолго до преступления.

Например, взлом сайта awwapp.com был совершен 18.03.2015 в 17:05:20. Необходимо найти всех пользователей, которые пользовались verisicretproxi.com 18.03.2015 в интервале 17:04:50 - 17:05:40.

Все запросы пользователей к сайтам сохраняются в журнале journal, реализованном с помощью связного списка. Журнал запросов в метро очень большой, т.к. одновременно WiFi в метро пользуется очень большое количество пассажиров. Простой проход по списку занимает очень большое время. Лог настолько велик, что за время поиска, преступник успевал скрыться, сменить компьютер, внешность и взгляды на жизнь.

После некоторых раздумий проект было решено поручить студентам НИУ ВШЭ, которые, в стремлении изучить новую интересную структуру данных, реализовали систему меньше чем за неделю.

Логи даны в виде строк в формате <дата> <время> <пользователь> <запрос>:

20.03.2015 17:04:50 ivan78 vk.com
20.03.2015 17:04:52 maria33 vk.com
20.03.2015 17:04:59 ivan78 vk.com
20.03.2015 17:05:02 oleg_oruk vk.com
20.03.2015 17:05:02 notebook_3 vk.com
20.03.2015 17:05:17 smith vk.com
20.03.2015 17:05:18 smith verisicretproxi.com
20.03.2015 17:05:23 oleg_oruk verisicretproxi.com
20.03.2015 17:05:24 smith facebook.com

Существующая архитектура

1. Класс `timestamp` используется для представления временного штампа «дата+время».
 - a. Методы:
 - i. операторы `==`, `<=`, `<` сравнения;
 - ii. оператор `>>` ввода временного штампа;
 - iii. оператор `<<` вывода временного штампа.
2. Классы-шаблоны `NodeWithKeyAbstract/NodeWithKey` для представления узлов упорядоченных списков:
 - a. Поля:
 - i. `m_key` – ключ элемента;
 - ii. `m_value` – хранимое значение элемента;
 - iii. `m_next` – указатель на следующий элемент.
3. Классы-шаблоны `NodeSkipListAbstract/NodeSkipList` для представления узлов списков с пропусками:
 - a. Параметры шаблона
 - i. `numLevels` – максимальное число дополнительных уровней у элемента;
 - b. Поля
 - i. `m_nextjump` – массив для дополнительных разреженных уровней быстрого перемещения по списку;
 - ii. `m_levelHighest` – максимальный уровень указателей в данном узле
4. Класс `OrderedList` – обыкновенный упорядоченный список
 - a. Методы:
 - i. `void insert(Value value, Key key)` - добавление нового элемента с ключом `key` и значением `value`;
 - ii. `void remove(Node * nodeBefore)` – удаление элемента следующего за `nodeBefore`;
 - iii. `Node * findFirst(Key key)`– поиск первого элемента по ключу `key`
 - iv. `Node * findLastLessThan(Key key)` – поиск последнего элемента с ключом строго меньше чем `key`
5. Класс `NetActivity` для представления активности пользователей.
 - a. Поля:
 - i. `string m_user` – имя пользователя;
 - ii. `string m_host` – имя запрашиваемого сервера.
 - b. Оператор вывода `<<` в формате: **mary218 msdn.com**.
6. Класс `JournalNetActivity` – реализация журнала для хранения и поиска сетевой активности
 - a. Поля
 - i. `m_journal` – список для хранения лога
 - b. Методы:
 - i. `void dumpJournal(ostream & out)` - вывод всего журнала в поток `out`
 - ii. `void parseLog(string fullpath)` – чтение файла лога

Необходимо реализовать:

7. В классе `SkipList`:
 - a. Методы:
 - i. `void insert(Value value, Key key)` - добавление нового элемента с ключом `key` и значением `value`;

- ii. `void remove(Node * nodeBefore)` – удаление элемента следующего сразу за `nodeBefore`;
 - iii. `Node * findLastLessThan(Key key)` – поиск последнего элемента с ключом строго меньшим чем `key`
 - iv. `Node * findFirst(Key key)`– поиск первого элемента по ключу `key`
- 8. В классе `JournalNetActivity`:
 - a. Методы:
 - i. `void outputSuspiciousActivities(string site, const TimeStamp & from, const TimeStamp & to)` – вывод на экран всех запросов пользователей к сайту `site` в интервале от `from` до `to` включительно.