Search and Find key word from “Oracle”

Savvas Rostantis

Contents

File structure

The project contains the files below:

- **Source:**
  - Main.cpp
  - Tree_Functions.cpp
  - List.cpp
  - Bloom_Filter.cpp
- **Headers**:
  - Tree.h
  - Oracle.h
  - List.h
  - Bloom_Filter.h
  - Hash.h
  - Libhash.a
  - Liboracle.a

This project is about search a hidden word using specific data structure and a filter. Tree_Functions.cpp (Tree.h) Contains the search tree alongside with the implementations and tree functions . List.cpp (List.h) contains a simple linked list and finally Bloom_Filter.cpp (Bloom_Filter.h) has the implantation of the filter for the word search .Oracle.h, Hash.h ,Libhash.a and Liboracle.a are libraries that contain the hidden word that we are going to find .

Compiling Overview

Cmake is utilize for the compilation process. A bash file for the exexution can be used. This file is responsible for the cmake execution and for the project execution. Open a bash command line and run:

- ./Run -k *Integer1* *Integer2* *Integer3*
- ./Run ,for clean up

1. Integer1: Number of functions

2. Integer2: Bloom Filter size in bytes

3. Integer3: Tree Depth

Execution Overview

The following structures were used as part of the exercise: A tree, a Bloom, a list of when the tree was filled, an auxiliary list for printing and deleting the tree, and a list of words to enter the tree after checking the filter. In main we call the oracle which gives us the tables with its words, then the oracle gives NULL or the tree returns a shutdown terminates (win, defeat). The tables pass to the tree. They filter by Bloom and with the help list we create new filtered tables, insert them into the tree and return the next word to main. Once the tree is filled, we use another list that makes the tree (again filtered). When the filter fills up, it stops filtering. Listing when empty means that there are no more words to look for and terminates main. At the end main prints some statistics of the program, deletes the structures, prints the tree and terminates the program. If oracle returns NULL, the list will be lost.

Data structure

List

The list contains pointers to the node of the tree in order to print and delete it.

Bloom Filter

The filter consists of a table with a number of hash functions and an indication of whether it has been filled. It includes initializing, the table in bytes, checking whether it has been filled, deleting it, returning the rate it has filled for statistics, and checking the import word. Bitwise operators and a mask are used for this control. We first execute hash functions to find the byte and the bit, that is, the position, which is in the table. We make a byte mask in the appropriate position (e.g. byte 2-> bit 4 -> mask = [to array [2] <-00001000). We check if the byte is changed, if so then we entered the table, if not no login and returns accordingly to the tree.

Tree

The tree includes a Bloom Filter, a list of when it is full, a list of words to save after filtering, and a list of auxiliaries for deleting and printing it. The tree is responsible for initializing other structures and for extracting information from main statistics and deleting them. The tree receives from the main word tables, passes the words through the filter and, depending on the output of the filter, saves them in a utility list. Then creates a node that will have the test word and binds it to a list of markers for his children according to the list after filtering, and finally the tree is updated so that the next word entry points to the appropriate node for their entry. When the tree reaches a given depth it uses a list to store words. When the list is empty it will send a signal to main to end in defeat. It prints out all messages for each case and is responsible for any other configuration settings.

Execution example

Compiling:

```
linux06:~/SavvasRostantis_1115201000149-project1> make
g++ -g main.o Tree_Functions.o List.o Bloom_Filter.o List.h oracle.h Tree.h Bloom_Filter.h hash.h libhash.a liborac
le_v2.a  -o invoke-oracle
linux06:~/SavvasRostantis_1115201000149-project1> ./invoke-oracle -k 5 120000 7
```

Execution example:

```
>>Seed[121622]  :pi^yoxfgu
>>Seed[121623]  :pj^yoyfhu
>>Seed[121624]  :pj^yowfgu
>>Seed[121625]  :pk^yoxfhu
>>Seed[121626]  :qj_ttxgft
>>Seed[121627]  :pm^xozfju
>>Seed[121628]  :pm^xozfiu
>>Seed[121629]  :shaxruifu
>>Seed[121630]  :ri`wqvhgu
>>Seed[121631]  :qj_xpwghu
>>Seed[121632]  :pi^wovfgu
>>Seed[121633]  :pj^xowfhu
>>Seed[121634]  :ri`yovfgu
>>Seed[121635]  :qj_yowfhu
>>Seed[121636]  :tibyqvhgu
>>Seed[121637]  :sjaypwghu
>>Seed[121638]  :ulcvqyhjp
>>Seed[121639]  :ulcwpygjp
>>Seed[121640]  :tmbxpzgkp
>>Seed[121641]  :qm_xpzgkp
>>Seed[121642]  :rm`xqzhkp
>>Seed[121643]  :uhcwpughp
>>Seed[121644]  :uhcvquhgp
>>Seed[121645]  :uhcxquhhp
>>Seed[121646]  :uhcxpugip
>>Seed[121647]  :uhcyquhip
>>Seed[121648]  :uhcypugjp
>>Seed[121649]  :thbxpugip
>>Seed[121650]  :thbyquhip
>>Seed[121651]  :shaypugjp
>>Seed[121652]  :umcytzkku
>>Secret word founded !!!You win!!!...
>>Secret word :!!!!umcytzkku!!!!
>>----------[Statistics]----------
>>--->Words used                  : 121652 words
>>--->Depth size of tree          : 7 levels
>>--->Words saved to tree         : 86 words
>>--->Number of hash functions    : 5 functions
>>--->Size of bloom filter        : 120000 bytes
>>--->Percentage Tree_W/Total_W   : 0.0706935%
>>--->Percentage Bloom filter used: 51.798%
>>--->Helper word list has        : 17439 words
>>--->Helper word Percentage      : 14.3352%
>>--->Execution time              : 130.09 sec
>>----------[Statistics]----------
>>Printing the word Tree...
>>Root :[ ]
>>---------->kids[1]
>>---------->(1):C
>>node>(1) :/
```