Googlebot SCC0202 Project

Parte II

1 - Introdução

Este projeto foi desenvolvido sob a tutoria do Prof. Dr. Rudinei Goularte e pelo pós-graduando Rodrigo M. Kishi do Programa de Aperfeiçoamento do Ensino durante a disciplina SCC0202 - Algoritmos e Estruturas de Dados I, lecionada durante o 2º semestre de 2018 no Instituto de Ciências Matemáticas e Computação do campus de São Carlos da Universidade de São Paulo.

Os alunos participantes do grupo responsável pelo desenvolvimento e documentação deste projeto foram Bruno Baldissera, Bruno Gazoni e João Villaça, cujos números USP são respectivamente 10724351, 7585037 e 10724239.

1 - Introduction

This project was developed under the mentorship of Professor Rudinei Goularte and the postgraduate student Rodrigo M. Kishi of the Program for the Improvement of Teaching during the discipline SCC0202 - Algorithms and Data Structures I, taught during the 2nd semester of 2018 at the Institute of Mathematical Sciences and Computing of the São Carlos campus of the University of São Paulo.

The students participating in the group responsible for the development and documentation of this project were Bruno Baldissera, Bruno Gazoni and João Villaça, whose USP numbers are respectively 10724351, 7585037 and 10724239.

2 - Motivação

A motivação principal deste projeto foi o desenvolver os conhecimentos passados em aula. Os tópicos incluidos foram tipos abstratos de dados, bibliotecas, listas e ponteiros.

2 - Motivation

The main motivation of this project was to develop the knowledge passed in class. Topics included were abstract data types, libraries, lists, and pointers.

3 - Resumo

O programa consiste reproduzir em miniatura um motor de busca que classifica diversos site em características específicas. Intitulado de Mini Googlebot, o projeto foi inspirado em um algoritmo real da Google que capta informações de todos os cantos da internet, formando um sistema de busca em uma grande base de dados de forma consistente

3 - Abstract

The program consists of reproducing in miniature a search engine that ranks several websites in specific characteristics. Entitled Mini Googlebot, the project was inspired by a real algorithm from Google that captures information from all corners of the internet, forming a search engine in a large database consistently.

4 - Licença

A distribuição deste programa está regido sob as normas estabelecidas pela licensa “The Unlicense”. A licença pode ser conferida integralmente abaixo:

4 - License

The distribution of this program is governed by the norms established by the license "The Unlicense". The license can be given in full below:

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

For more information, please refer to <http://unlicense.org>

Este software está disponível para qualquer pessoa como um software livre. Para mais informações, acesse o link da organização que formalmente dispõe esta no link referido ao final da licença anexa acima.

This software is available to anyone as a free software. For more information, access the link of the organization that formally has this in the link referred at the bottom of the license attached above.

5 - Características Fundamentais

Este software foi completamente desenvolvido utilizando a linguagem de programação C em seu padrão ANSI C89. Não há garantia de que o programa será executado corretamente em uma versão da linguagem que nao seja ANSI-C89.

As plataformas de desenvolvimentos utilizadas foram o sistema operacional GNU/Linux na distribuição Lubuntu e o GitHub para compartilhamento de código. Tambem nao ha garantia de que o programa seja executado corretamente em outros sistemas operacionais.

O compilador utilizado foi o GCC versão 4.8. Esta versão é a mais indicada caso o usuário deseje compilar o código fonte. Além dos arquivos disponíveis que compõem o programa e esta documentação, há uma Makefile que pode ser utilizada para compilar e executar o programa desde que somente os códigos-fonte do programa estejam isolados juntamente do arquivo `client.c` em um diretório. A makefile deve ser executada pelo console do Linux sob o comando `make` quando o usuário já estiver visualizando o diretório previamente mencionado.

A makefile utilizada passa o parâmetro `-ansi` para o compilador, garantindo que o código foi feito sob os padrões ANSI.

5 - Fundamental Characteristics

This software was completely developed using the C programming language in its standard ANSI version. There is no guarantee that the program will run correctly on a language version other than ANSI-C89.

The development platforms used were the GNU / Linux operating system in the Lubuntu distribution and GitHub for code sharing. There is also no guarantee that the program will run correctly on other operating systems.

The compiler used was GCC version 4.8. This version is most suitable if you want to compile the source code. In addition to the available files that make up the program and this documentation, there is a Makefile that can be used to compile and run the program as long as only the program source code is isolated with the `client.c` file in a directory. The makefile must be run by the Linux console under the `make` command when the user is already viewing the previously mentioned directory.

The makefile also calls the compiler with the parameter `-ansi`, ensuring that the code was made following the ANSI patterns.

6 - Obtendo o programa

O programa pode ser obtido no repositorio <https://github.com/ZeroDoisBCC018/Algoritmos-e-Estruturas-de-Dados> tanto por cópia, `fork` ou download de uma pasta ZIP. Os arquivos `LICENSE` e `README.md`

estão no repositorio apenas pelo padrão de criação de repositórios da plataforma. Durante o período de desenvolvimento, o arquivo `test.csv` (também disponível no repositório) foi utilizado na bateria de testes do Googlebot.

6 - Getting the program itself

The program can be obtained from the <https://github.com/ZeroDoisBCC018/Algoritmos-e-Estruturas-de-Dados> repository either by copying, `fork` or downloading a ZIP folder. The `LICENSE` and `README.md` files are in the repository only by the default platform repository creation. During the development period, the test.csv file (also available in the repository) was used in the Googlebot test battery.

7 - Padrão de entrada (Database)

A base de dados utilizada no Googlebot deve seguir exatamente o padrão que está no arquivo de teste. Os sites são objetos com as seguintes características: código, nome, relevância, link da página principal e palavras-chave.

- Os códigos devem ser números inteiros entre 1 e 9999,
- O nome deve ter entre 1 e 50 caracteres,
- A relevância deve ser um número inteiro entre 0 e 1000,
- O link da página principal deve ter entre 1 e 100 caracteres,
- Cada site pode conter até 10 palavras-chave, cada uma de 1 a 50 caracteres.

Como o arquivo deve seguir o padrão .csv, os valores devem ser separados por vírgulas, e as strings (palavras) devem estar entre aspas duplas (opcional). O cabeçalho (primeira linha) do arquivo .csv deve conter os nomes das características atribuídas aos sites, em ordem.

7 - Input pattern (Database)

The database used in Googlebot should follow exactly the pattern that is in the test file. Sites are objects with the following characteristics: code, name, relevance, main page link and keywords.

- *Codes must be integers between 1 and 9999,*
- *Name must be between 1 and 50 characters,*
- *The relevance must be an integer between 0 and 1000,*
- *The home page link must be between 1 and 100 characters,*
- *Each site can contain up to 10 keywords, each of 1 to 50 characters.*

Because the file must follow the .csv pattern, the values must be separated by commas, and the strings must be enclosed in double quotation marks (optional). The header (first line) of the .csv file should contain the names of the characteristics assigned to the sites, in order.

8 - Biblioteca e segmentos de código-fonte

Todo o código dedicado ao projeto, desde as estruturas de dados até funções e algoritmos relacionados foram unificados em uma única biblioteca chamada `googlebot.h`, cujo código-fonte está em um arquivo de mesmo cunho: `googlebot.c`. Um cliente de teste com a apresentação do programa mostrada diretamente no terminal está no arquivo `client.c`, que possui a referência a uma função `main`.

8 - Source code segments and library

All code dedicated to the project, from data structures to functions and related algorithms, was unified into a single library called `googlebot.h`, whose source code is in a file of the same kind: `googlebot.c`. A test client with the program presentation shown directly on the terminal is in the `client.c` file, which has the reference to a `main` function.

9 - Escolha da estrutura de dado e seus algoritmos relacionados

Nesta 2ª parte do projeto, o grupo arquitetou o Googlebot utilizando duas estruturas de dado diferentes: uma lista simplesmente encadeada e uma árvore de prefixos, também conhecida como Trie. A lista é uma forma prática de armazenar os sites durante o uso do programa e a Trie oferece a busca de palavras de forma muito eficiente, que é a operação principal desta parte do projeto.

Quando a busca pelas palavras-chave é realizada no programa, a palavra é buscada na Trie, cujo o último nó aponta para todos os sites que possui aquela palavra-chave. A sugestão de sites também fica eficiente desta forma.

A lista é utilizada para inserir os sites e, caso o cliente queira, ordená-los por relevância ou código para atualizar seu banco de dados ou mostrá-lo diretamente no terminal. Ambas as estruturas são dedicadas aos tipos de dado utilizados nas operações do programa e são implementadas com uso de memória primária alocada dinamicamente.

O algoritmo de ordenação final utilizado no programa é semelhante a um Insertion Sort comum, um algoritmo largamente conhecido. Apesar de sua complexidade ser quadrática, o Insertion se mostrou uma opção adequada. Isso porquê a opção inicial do projeto, que era um Radix Sort utilizando um Bucket Sort como rotina apresentou alguns problemas, entre eles: a necessidade de listas auxiliares, que demandava mais operações e memória auxiliar do que o esperado, o código extenso, que dificultava a visualização e otimização do funcionamento, e a dificuldade de debugging, pois além do código extenso, o número e a variedade de operações tornou a projeção do algoritmo e a depuração muito mais extensa. Apesar da complexidade linear do Radix, muitas operações e o uso de memória auxiliar fizeram com que sua eficiência caísse vertiginosamente, além da

implementação, que complicada demais para os propósitos do projeto. Já o Insertion Sort funcionou bem, usando muito menos memória auxiliar, e por se basear apenas em comparações entre os dados guardados na lista, sendo mais rápido e simples de entender e depurar.

A complexidade da busca na Trie está relacionada somente com o número de letras da palavra buscada e dos acessos de ponteiros. Uma ótima opção para um programa que executará muitas buscas baseadas em strings. Outras árvores conhecidas e estudadas pelo grupo até o momento exigiria muito mais tempo para a projeção de buscas baseadas em strings. Apesar de usar certa memória auxiliar, ambas as estruturas de dados e o algoritmo de ordenação foram eficazes e completaram suas tarefas com eficiência satisfatória.

9 - Choice of data structure and its related algorithms

In this second part of the project, the group architected the Googlebot using two different data structures: a simply linked list and a prefix tree, also known as Trie. The list is a practical way of storing the sites while using the program and the Trie offers the word search very efficiently, which is the main operation of this part of the project.

When keyword search is performed in the program, the word is searched in Trie, whose last node points to all sites that have that keyword. The suggestion of websites is also efficient this way.

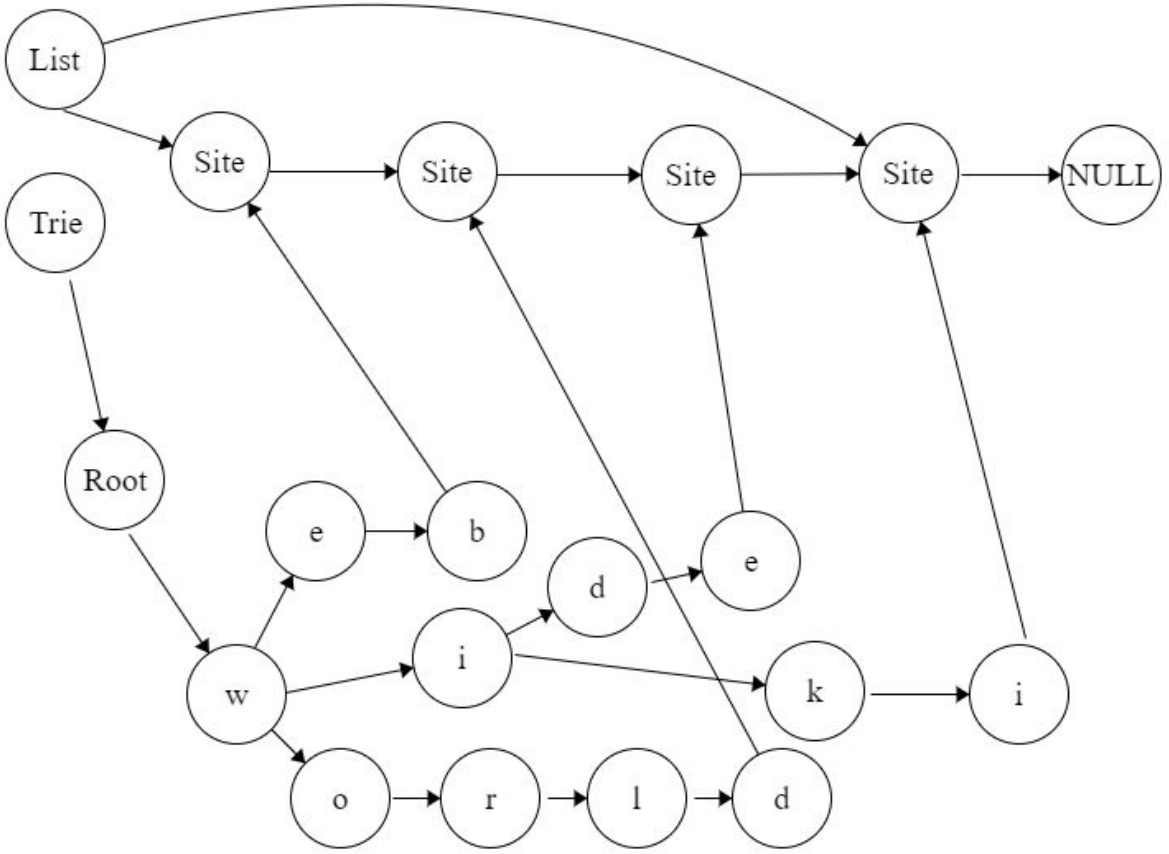
The list is used to insert the sites and, if the client wants, to order them by relevance or code to update your database or show it directly in the terminal. Both structures are dedicated to the data types used in program operations and are implemented using dynamically allocated primary memory.

The final sorting algorithm used in the program is similar to a common Insertion Sort, a widely known algorithm. Although its complexity is quadratic, Insertion has proved to be a suitable option. That's why the initial project option, which was a Radix Sort using a Bucket Sort subroutine, presented some problems, among them: the need for auxiliary lists, which demanded more operations and auxiliary memory than expected, the extensive code, which made difficult optimization, and increased the difficulty of debugging, since besides the extensive code, the number and the variety of operations made the projection of the algorithm and the debugging much more extensive. Despite the linear complexity of Radix, many operations and the use of auxiliary memory have caused its efficiency to fall precipitously, in addition to the implementation, which is too complicated for the purposes of the project. Insertion Sort has worked well, using much less auxiliary memory, and is based only on comparisons between the data stored in the list, being faster and easier to understand and debug.

The complexity of the search in Trie is related only to the number of letters of the searched word and the pointers accesses. A great option for a program that will perform many string-based searches. Other trees known and studied by the group so far would require much more time for the projection of string-based searches. Despite using some auxiliary memory, both the data structures and the sorting algorithm were effective and completed their tasks with satisfactory efficiency.

Esquema das estrutura de dados utilizada:

Schema of the data structures used:



10 - Tratamento de erros e exceções

A maior parte das funções implementadas retorna algum código indicando sucesso ou erro. Há quatro macros criadas que ajudam no tratamento de erro através do retorno das funções: `ERROR`, `SUCCESS`, `FAIL`, `TRUE` e `FALSE`, sendo que os dois últimos códigos são subtipos “booleanos” adaptados ao C89. O programa possui mensagens de erro que serão impressas no buffer de console `stderr`, porém as mensagens de erro só são impressas caso um erro fatal (leia: que impede o programa de continuar) seja detectado. Nestes casos o comando `exit(EXIT_FAILURE)` finaliza o programa.

Na implementação das funções, não há exceções que foram tratadas. Isso significa que o cliente deve obedecer aos parâmetros e retornos das funções, já que caso estes não sejam seguidos corretamente, o código de erro será retornado, as funções serão desempilhadas e devido a isso, as variáveis perdem a estabilidade de seus valores, que seriam utilizados para executar funções em sequência, e o sistema aborta o processo, resultando no erro `Segmentation fault (Core dumped)`. Se o espaço no disco acabar, o programa também é encerrado para evitar heap overflow ou outras falhas que possam danificar o computador do cliente.

Os segmentos de código estão comentados e a interface da biblioteca está com os retornos, parâmetros e funcionalidades descritas. para entender o

funcionamento do programa em mais baixo-nível, acesse os segmentos disponibilizados no repositório oficial.

10 - Error Handling and exceptions

Most of the implemented functions return some code indicating success or error. There are four created macros that help in error handling by returning functions: `ERROR`*,* `SUCCESS`*,* `FAIL`*,* `TRUE` *and* `FALSE`*, with the last two codes being "boolean" subtypes adapted to C89. The program has error messages that will be printed in the* `stderr` *console buffer, but error messages will only be printed if a fatal error (read: which prevents the program from continuing) is detected. In these cases, the* `exit(EXIT_FAILURE)` *command ends the program.*

In the implementation of the functions, there are no exceptions that have been addressed. This means that the client must obey the parameters and returns of the functions, since if they are not followed correctly, the error code will be returned, the functions will be unsettled and because of this, the variables lose the stability of their values, which would be used to perform functions in sequence, and the system aborts the process, resulting in the `Segmentation fault (Core dumped)` *error. If disk space runs out, the program also shuts down to prevent heap overflow or other faults that could damage the client's computer.*

Code segments are commented and the library interface has the returns, parameters, and functionality described. To understand the operation of the program at a lower-level, access the available segments in the official repository.