*Programming in C# (NPRG035 and NPRG038)*

Project: **Web crawler + indexer**

Tijana Stanković

**Motivation**

The amount of information that people generate is huge. In modern time, most of them are on the web, organized as **websites**.

As we know, a website is a set of web pages that are composed of text and images and are interconnected by html links.

Due to the huge number of websites, manually searching for information is practically impossible. That’s the job of web **search engines -** tools that enable users to find information on the internet by entering keywords or phrases.

Due to their significance, I think creating a small equivalent of a web search engine should be both a challenging and interesting project.

**Project**

A search engine essentially consists of two main components: a **web crawler** and an **indexer**.

The web crawler, also known as a spider, automatically navigates the internet, collecting information from web pages.

The indexer then organizes and stores this information in an index, which the search engine uses to quickly find relevant results when a user enters a query.

So, my project would consist of two main parts:

* a web crawler component and
* an indexer component

**Problems**

Since I want to create a small equivalent of a search engine, I first need to describe some challenges and limitations:

* Due to the vast number of websites, some of them with hundreds or thousands of pages, it is clear that, given limitations in my computing resources and time, my web crawler component cannot consider searching the entire web (as large search engines like Google, Bing, and Yahoo do).
* For the same reasons, the amount of information that this small search engine can store is (very) limited, so my indexer can create relativily small index with limited set of keywords.

**Possible solution**

To make this small search engine possible, we can limit the search to a specific set of websites, with a restricted depth of branching (following links that point to other pages), and by indexing only a predefined set of keywords.

This way, we would create a specialized search engine that could, for example, search certain sports sites only and look for information about certain athletes.

**Project description**

My idea is to create a program that would allow users to interactively define and change a set of initial web pages from which the search begins, the scope of the search (the maximum number of links the web crawler can follow starting from the initial page), and the set of keywords it looks for.

Then, from time to time, the user will be able to initiate the search process to create or update the information in the index. During the search process, an internal database will be formed in memory, which will serve as an index for quick data retrieval.

Based on the information in the index, the user will be able to quickly find and list web pages that contain one or more keywords.

**Details**

The program would be started as:

spider [<db\_filename>[.sdb]]

The program parameter is the name of the file that contains the database of web pages. This is an internal format understood and used only by this program, where information about web pages would be stored. If this parameter is not specified, the default name would be **spider\_db.sdb.** If the extension is not specified, the default would be **.sdb** („spider database“).

So, when the program starts, the following happens:

1. Data is loaded from the file whose name is specified as described above, and a database of web pages information and keywords connected with them, and with indexes is formed in the memory (or, if the file does not exist, a new, empty file with an empty database is created).
2. The program then displays basic information about the program (version, etc.) and the command for displaying help.
3. The program finally goes to interactive mode where the user can enter commands which the program will execute.

**List of commands:**

* **HELP**Displays the HELP page with a list of commands.
* **ABOUT**Displays information about the program.
* **EXIT**

Exit the program.

* **SAVE [<new\_spider\_db\_filename>]**

Saves the current state of the memory (database) to the local file.

* **ADD <name> <web-page-url> <internal-depth> <external-depth>**

Defines a new crawling **starting point** (URL from which the search begins) with the specified name.  
Using this command, we can define multiple starting points and use some or all of them.  
When the program find a new link on the page, it can follow that link and continue searching on that page. The maximum number of links the program can follow in this manner, starting from the first page, is defined by the parameters <internal-depth> (refers to pages within the same website as the initial page) and <external-depth> (refers to pages on other websites).

* **ADD KEYWORD <keyword>**Adds a new keyword to the database or updates an existing one.
* **REMOVE <name>**Removes a starting point with the specified name from the database.
* **REMOVE KEYWORD <keyword>**Removes the specified keyword from the database.
* **LIST**Displays database statistics.
* **LIST KEYWORDS**  
  Lists all existing keywords in the database.
* **LIST NAMES**  
  Lists all existing starting points in the database.
* **PSCAN [<number-of-threads>]**

Sets the maximum number of threads used during the web crawling process, enabling or disabling parallel crawling based on the specified value of parameter:

* + Value = 1: Parallel web crawling is disabled. Crawling is performed sequentially.
  + Value > 1: Parallel web crawling is enabled, using up to the specified number of threads.

If the parameter is omitted, it shows the parallel crawling status and max number of threads.

The web crawling process (both sequential or parallel) is started later using the SCAN command.

* **SCAN [<name-list>] [<keyword-list>]**Initiates the web crawling process for the specified starting point names and keywords.  
  If <name-list> (<keyword-list>) is not specified, all names (keywords) in the database will be used.  
  During the crawling process, the web crawler will start from the URLs of starting points specified by <name-list> and search those web page(s) for any of keywords present in the <keyword-list>.   
  If it finds a keyword, it will create a connection between the web page and the keyword in the database index.   
  Also, if it finds a HTML link on the page and if it is allowed by the internal and external depth parameters (specified when the starting point is defined), the web crawler will follow these links and continue searching on those pages in the same manner.
* **SCAN KEYWORDS [<keyword-list>] [<name-list>]**Same as the SCAN command, but we specify keywords first, then names.
* **FIND <keyword>**

Find and list URL of all pages in the database that contain the specified keyword.

* **LOG**During the web crawling process, crawling detail are logged in the log file.   
  Using this command we can manipulate with log file (e.g. set a log filename, clear log file etc).  
  The log file is a plain text file that can be opened and viewed in any text editor (e.g., Notepad).