Multithreaded Web Crawler

1. Introduction

This project aims to develop a multithreaded web crawler in Java, which will explore a website starting from an initial URL, gather all reachable HTTP links, and recursively download content from each link up to a specified depth. The crawler runs each link download in a separate thread for parallel processing, enabling efficient and fast crawling. Additionally, the crawler logs essential information about visited links, such as URL, size, and crawl time, and presents this data in a browsable index with an interactive GUI.

2. Project Objectives

- **Design a Multithreaded Web Crawler**:
  - Implement a recursive process to crawl web pages, extracting links and following them up to a specified depth.
- **Parallelize the Downloading Process**:
  - Use Java's multithreading capabilities (e.g., Thread, ExecutorService) to download links in parallel.
- **Information about Crawled Links**:
  - Maintain a table that tracks URLs, their sizes, and crawl times.
- **Browsable Index**:
  - Create an interactive GUI to display crawled links and associated data.

3. Technical Approach

3.1 Web Crawling

- **Starting URL**:
  - The crawler begins with a specified URL, using HttpURLConnection to fetch webpage content.
- **Link Extraction**:
  - HTML links are extracted using regular expressions to parse <a href> tags.
- **Depth Limitation**:
  - Recursive crawling is limited to a maximum depth (e.g., 2), as specified in the implementation, to avoid overloading.

3.2 Multithreading

- **Thread Per Link**:
  - Each URL is assigned to a thread for downloading, managed by an ExecutorService with a fixed thread pool of size 10.
- **Thread Pooling**:

    - `ExecutorService` is used to manage concurrent tasks efficiently, avoiding overhead from creating too many threads.
- **Synchronization**:
    - Proper synchronization ensures thread safety while accessing shared resources, such as the database and visited URLs set.

3.3 Information Collection

- **Log Structure**:
    - Data collected includes:
        - URL: The address of the crawled page.
        - Size: Number of extracted links (representing the page size).
        - Crawl Time: Timestamp of when the page was crawled.
- **Log Storage**:
    - Data is stored in a MySQL database with a schema that includes fields for URL, size, and crawl time.

3.4 Interactive GUI

- **Browsable Index**:
    - The GUI, built using Java Swing, presents crawled data in a tree structure.
    - Crawled links are organized hierarchically, allowing users to navigate and explore connections visually.

4. Tools and Technologies

- **Java Core Libraries**:
    - `HttpURLConnection` for HTTP requests.
    - `ExecutorService` for managing the thread pool.
- **HTML Parsing**:
    - Regular expressions are used for link extraction.
- **Data Storage**:
    - MySQL database for storing crawled data.
- **Interactive GUI**:
    - Java Swing for creating a user-friendly interface with a dynamic tree structure.

5. Implementation Steps

1. **Step 1**:
    - Set up the project with basic single-page crawling and link extraction functionality.
    - Research and implement URL fetching using `HttpURLConnection`.
2. **Step 2**:
    - Add multithreading using `ExecutorService` to handle concurrent downloads.
    - Design and initialize the MySQL database schema for storing crawl data.

3. **Step 3**:
    - Implement recursive crawling with depth control and validate link extraction results.
    - Optimize thread handling and manage shared resources like the visited URLs set.
4. **Step 4**:
    - Develop the GUI to display crawled data dynamically in a tree structure.
    - Integrate the GUI with the database to fetch and display crawled links.
5. **Step 5**:
    - Perform testing and optimization to handle edge cases (e.g., broken links, timeouts).
    - Finalize project documentation and polish the GUI for better usability.

6. Code Implementation Overview

6.1 Key Components of the Code

Multithreaded Crawling

```
public void startCrawl(String startUrl) {
    linkQueue.add(new Link(startUrl, 0));

    while (!linkQueue.isEmpty() || !executor.isTerminated()) {
        Link link = linkQueue.poll();
        if (link == null) continue;
        String url = link.url;
        int depth = link.depth;

        if (depth > MAX_DEPTH || visitedUrls.contains(url)) continue;

        visitedUrls.add(url);
        executor.execute(() -> {
            try {
                List<String> links = downloadAndExtractLinks(url);
                System.out.println("Crawled: " + url + ", Found " + links.size() + " links");
                saveToDatabase(url, links.size());

                SwingUtilities.invokeLater(() -> {
                    CustomTreeNode parentNode = findNode(root, url);
                    if (parentNode == null) {
                        parentNode = new CustomTreeNode(url);
                        root.addChild(parentNode);
                    }
                    for (String linkUrl : links) {
                        CustomTreeNode childNode = new CustomTreeNode(linkUrl);
                        parentNode.addChild(childNode);
                    }
                    treeModel.setRoot(convertToSwingTreeNode(root));
                    treeModel.reload();
                });

                if (depth < MAX_DEPTH) {
                    for (String linkUrl : links) {
                        if (!visitedUrls.contains(linkUrl)) {
```

```
                    linkQueue.add(new Link(linkUrl, depth + 1));
                }
            }
        }
    } catch (IOException e) {
        System.err.println("Failed to crawl: " + url + " - " + e.getMessage());
    }
});
}

waitForCompletion();
}
```

GUI Integration

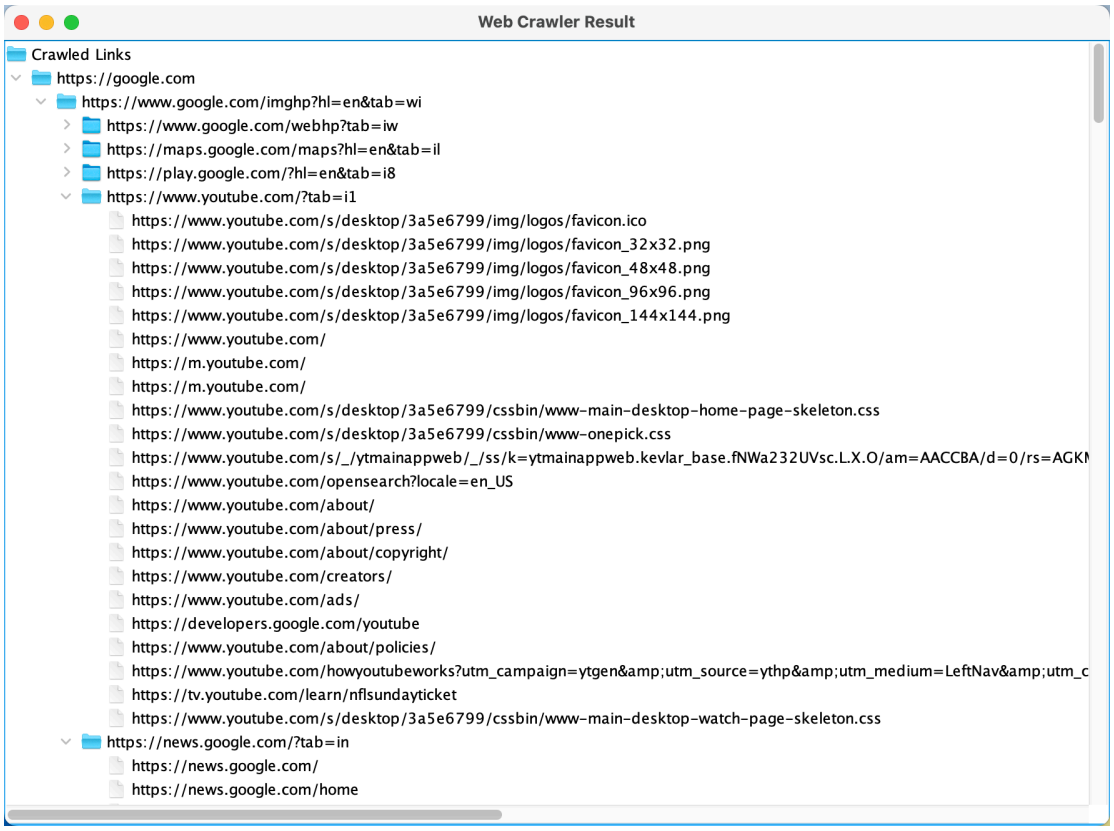
```
private void createAndShowGUI() {
    JFrame frame = new JFrame("Web Crawler Result");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JTree tree = new JTree(treeModel);
    JScrollPane scrollPane = new JScrollPane(tree);
    frame.add(scrollPane);

    frame.setSize(400, 600);
    frame.setVisible(true);
}
```

6.2 Database Interaction

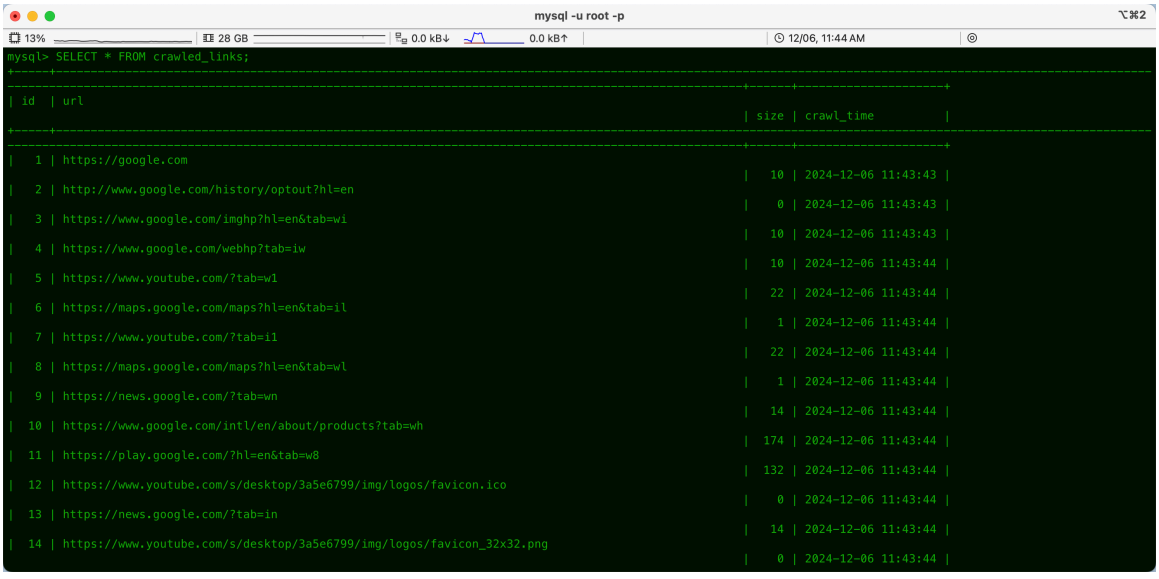
```
private void saveToDatabase(String url, int size) {
    try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {
        String sql = "INSERT INTO crawled_links (url, size, crawl_time) VALUES (?, ?, NOW())";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, url);
            pstmt.setInt(2, size);
            pstmt.executeUpdate();
        }
    } catch (SQLException e) {
        System.err.println("Failed to save to database: " + e.getMessage());
    }
}
```

7. Results and Evaluation

- **GUI Result Example**:
    - Below is a snapshot of the graphical user interface displaying the crawled links in a hierarchical tree structure:



- Performance metrics:
  - Maximum crawling depth achieved efficiently with a thread pool of size 10.
  - Average time for link extraction and storage operations.
- GUI functionality:
  - Dynamic tree structure visualizes hierarchical relationships between links.
- Database integration:
  - Successfully stores crawled link data with minimal latency.



8. Conclusion

- The multithreaded web crawler successfully demonstrates efficient crawling, data extraction, and presentation.
- Key learning outcomes include hands-on experience with multithreading, database integration, and GUI development.
- Future work could focus on enhancing error handling, scaling for larger websites, and adding advanced features like link prioritization.

9. References

［1] Oracle, "Class URL," [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/net/URL.html. [Accessed: Dec. 6, 2024].

[2] Stack Overflow, "Connect Java to a MySQL Database," [Online]. Available: https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database. [Accessed: Dec. 6, 2024].

[3] Oracle, "Creating a GUI with Swing," [Online]. Available: https://docs.oracle.com/javase/tutorial/uiswing. [Accessed: Dec. 6, 2024].

[4] E. H. Chua, "JDBC Basics," Nanyang Technological University, [Online]. Available: https://www3.ntu.edu.sg/home/ehchua/programming/java/JDBC_Basic.html. [Accessed: Dec. 6, 2024].

[5] Stack Overflow, "Java Web Crawler Libraries," [Online]. Available: https://stackoverflow.com/questions/11282503/java-web-crawler-libraries. [Accessed: Dec. 6, 2024].

[6] Oracle, "Class Pattern," [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html. [Accessed: Dec. 6, 2024].