INFO 6205: Program Structure & Algorithms.

Group 5:
1.Salman Shaikh (002811778)
2.Srinjana Nag (002478377)

Web Crawler Project Report

---

1. Introduction

This report presents the development and implementation of a dynamic web crawler designed to analyze websites based on user-provided URLs. The crawler uses Breadth-First Search (BFS) traversal and asynchronous processing to efficiently extract page titles and links. For demonstration, the crawler was applied to https://www.wikipedia.org/, successfully traversing over 50,000 pages and mapping more than 80,000 links.

The crawler stores data in Neo4j, enabling advanced analysis such as PageRank and link structure evaluation. It is built to handle diverse scenarios, including invalid links and missing titles, while ensuring scalability and reliability with robust error logging using Log4j. This report details the methodology, performance benchmarks, and findings, showcasing the crawler's versatility and effectiveness.

---

2. Project Objectives

1. **Develop a BFS-based Web Crawler**:
    - Crawl web pages starting from **https://www.wikipedia.org/**.
    - Traverse up to a specified maximum depth.

2. **Support Asynchronous Processing**:
    - Utilize Java's *CompletableFuture* for concurrent crawling and database writes.

3. **Store and Analyze the Web Graph**:
    - Store crawled pages and their links in a Neo4j graph database.
    - Perform PageRank analysis to determine the importance of pages.

4. **Benchmark Performance**:
    - Evaluate the crawler's performance in terms of pages crawled per second and database processing times.

---

3. Implementation Details

3.1 Architecture Overview

- The project consists of three main modules:

    - **Crawler**: Performs BFS traversal, extracts links, and interacts with the database.
    - **Graph Database Integration**: Neo4j is used to store nodes (pages) and relationships (links).
    - **Benchmarking Module**: Measures the performance of crawling and database operations.

3.2 Key Features

1. **Breadth-First Traversal**:
    - Queue-based traversal with visited pages tracked using a *HashSet*.
2. **Asynchronous Processing**:
    - Multithreaded crawling implemented using *CompletableFuture* for parallel processing.
3. **Graph Representation**:
    - Pages are represented as nodes with attributes like URL and title.
    - Links between pages are represented as LINKS_TO relationships.
4. **Data Integrity Validation**:
    - Validated link extraction and ensured proper database writes for accurate graph representation.
5. **Error Handling**:
    - Handled invalid URLs, SSL errors, and missing page titles.

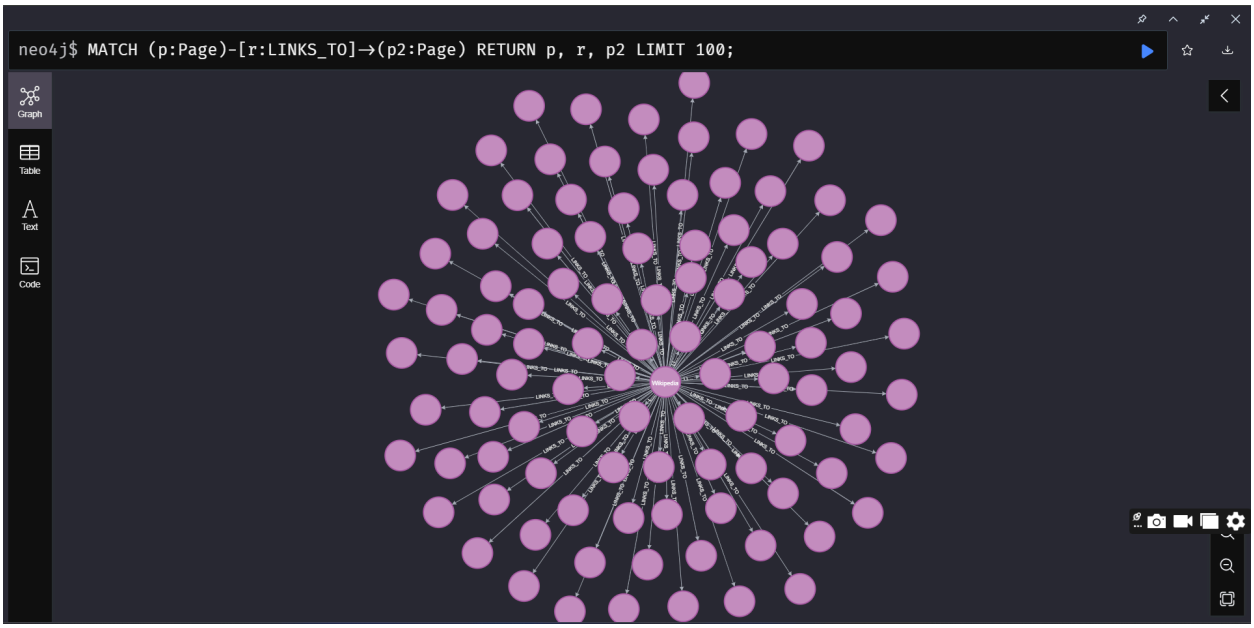
---

4. Results

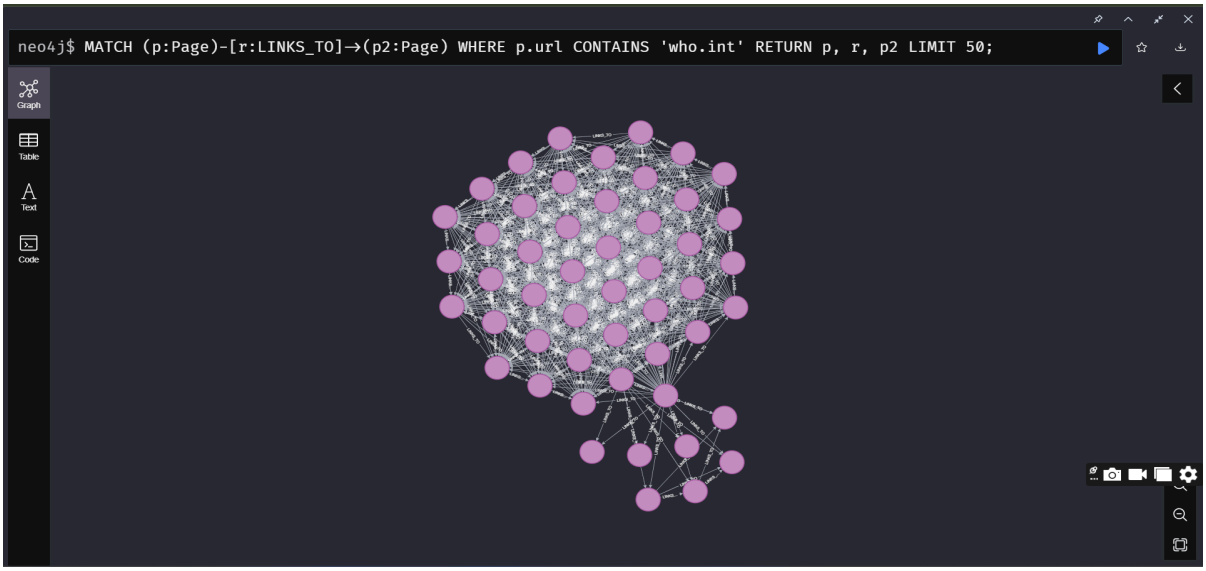
4.1 Graph Metrics

The crawler successfully analyzed the web structure of **https://www.wikipedia.org/** and generated the following metrics:

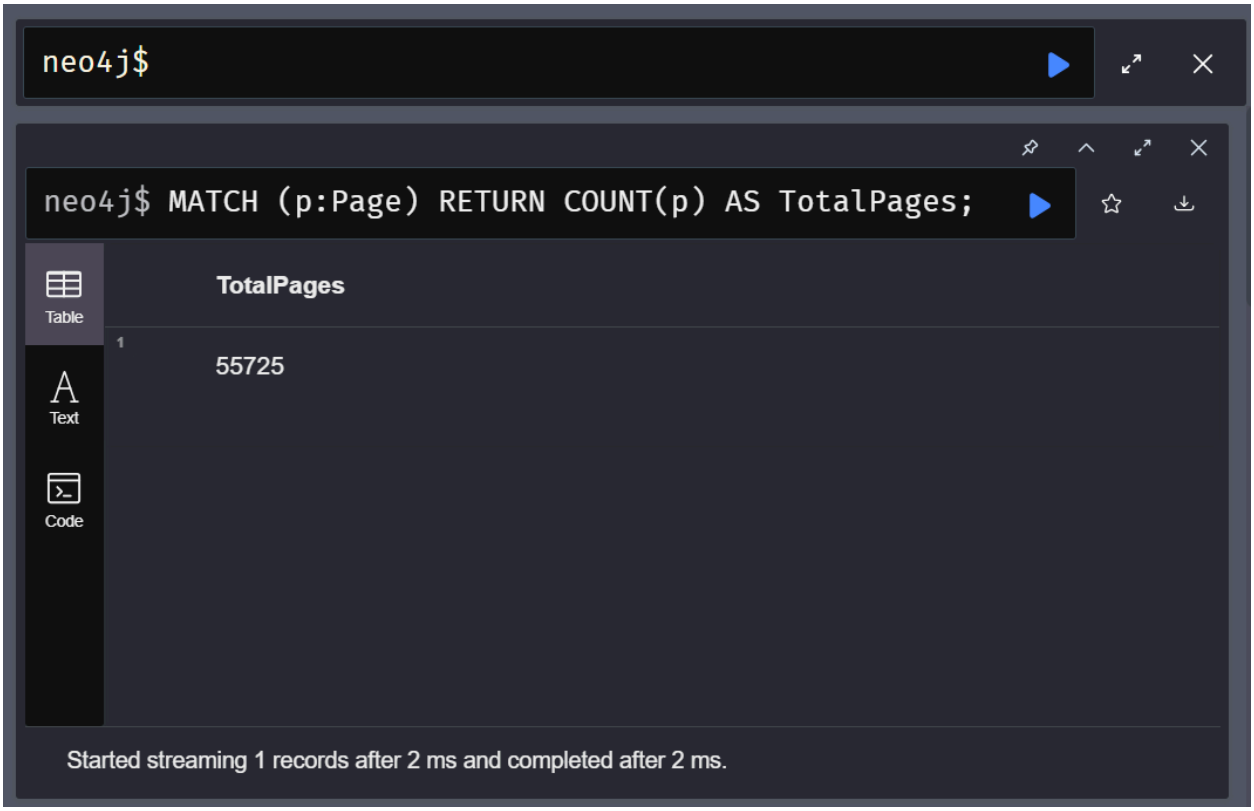
Graph Visualization of Page Relationships and Link Structure:



Graphical Visualization of Crawled Data:

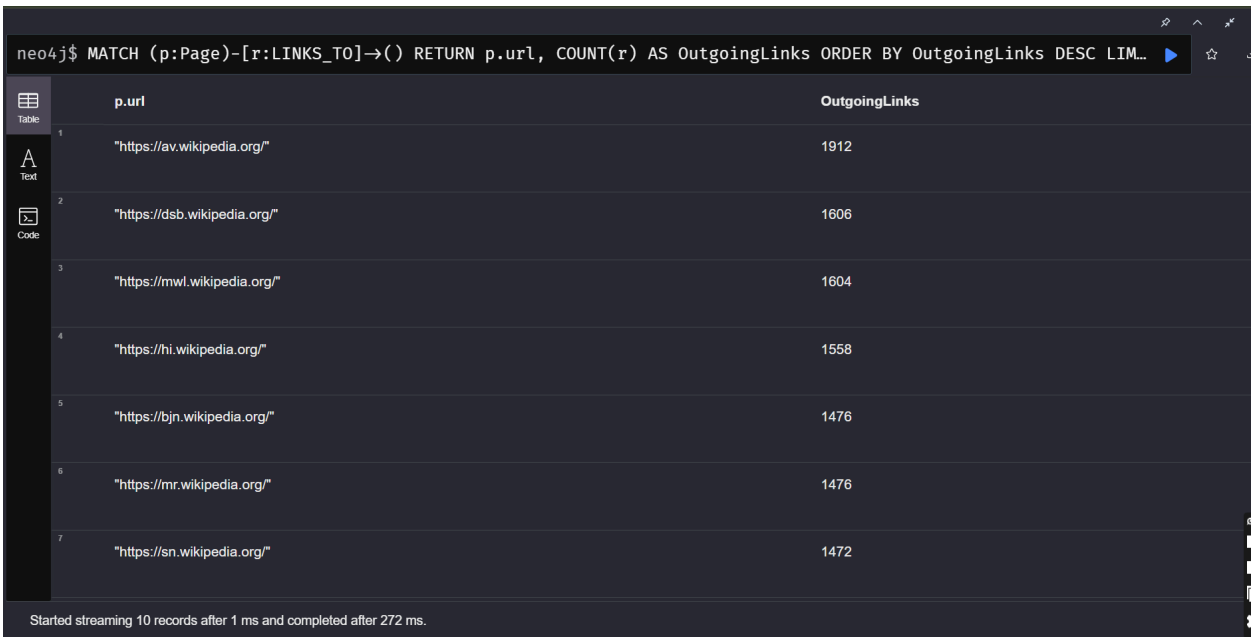


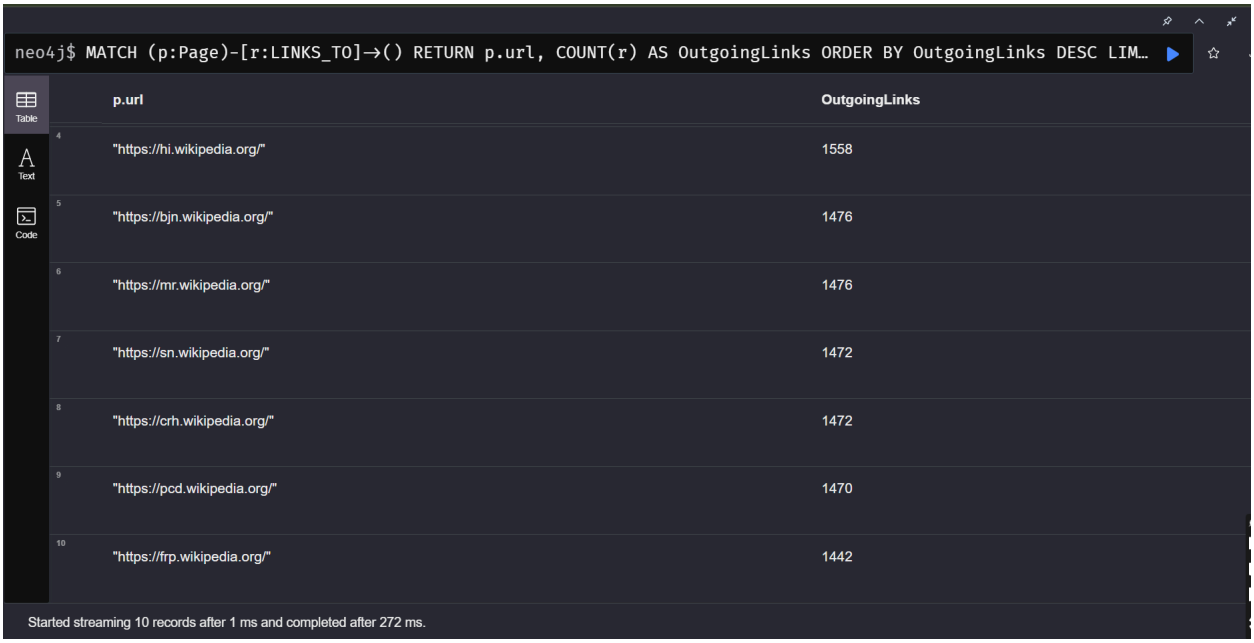
1. Total Pages Crawled: 55,725



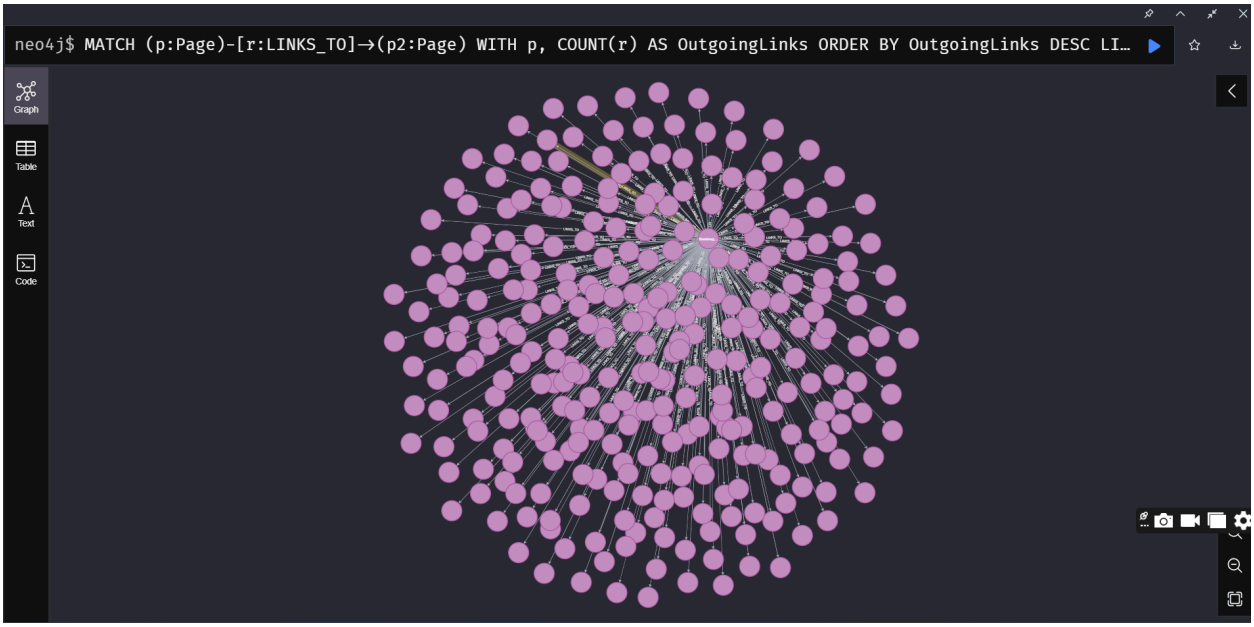
2. Total Links Extracted: 31,5022

3. Find Pages with the Most Outgoing Links





This visualization highlights the relative significance of pages by scaling their size based on the number of outgoing links they contain, offering a clear representation of their connectivity.

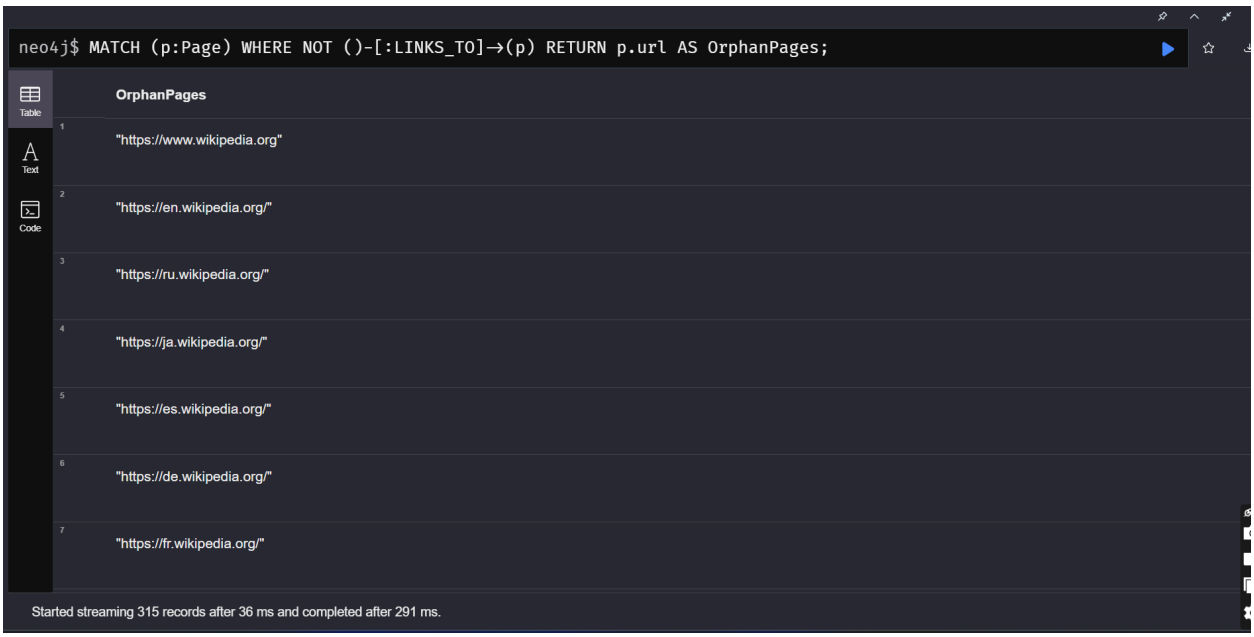


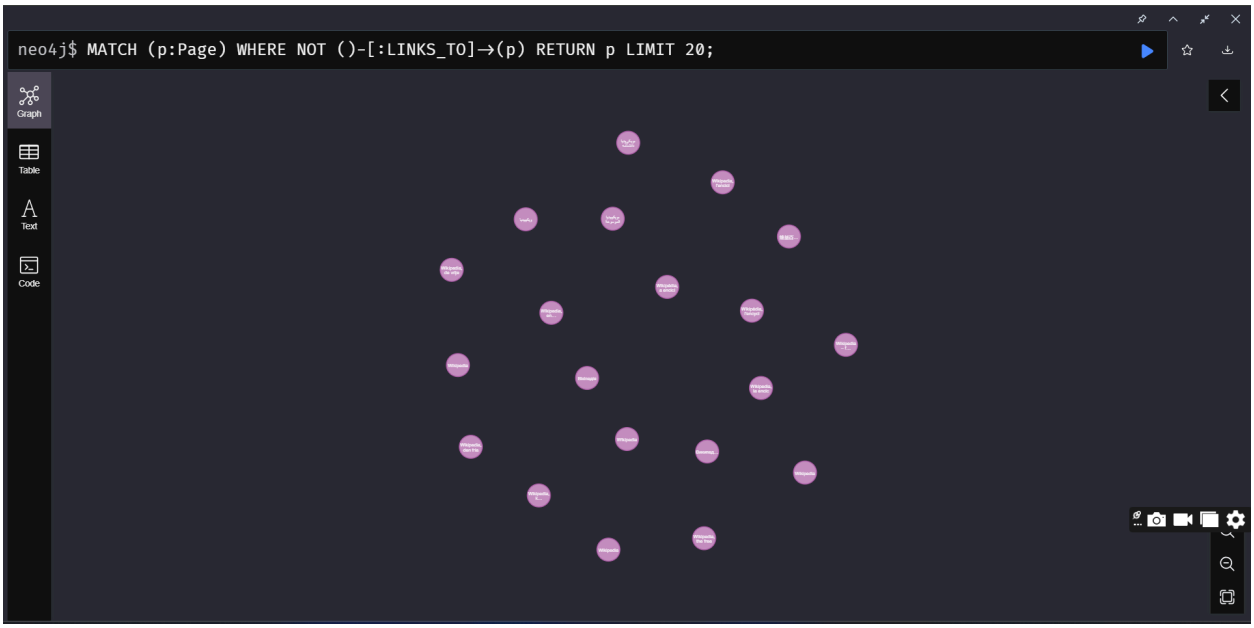
4. Find Pages with the Most Incoming Links


neo4j$ MATCH ()-[r:LINKS_TO]→(p:Page) RETURN p.url, COUNT(r) AS IncomingLinks ORDER BY IncomingLinks DESC LIM…
Table
Text
Code
p.url
IncomingLinks
"https://meta.wikimedia.org/wiki/Main_Page" 616
"https://wikisource.org/wiki/Main_Page" 616
"https://commons.wikimedia.org/wiki/Main_Page" 616
"https://foundation.wikimedia.org/wiki/Home" 616
"https://outreach.wikimedia.org/wiki/Main_Page" 616
"https://www.mediawiki.org/wiki/MediaWiki" 616
"https://www.wikidata.org/wiki/Wikidata:Main_Page" 616
Started streaming 10 records after 39 ms and completed after 799 ms.

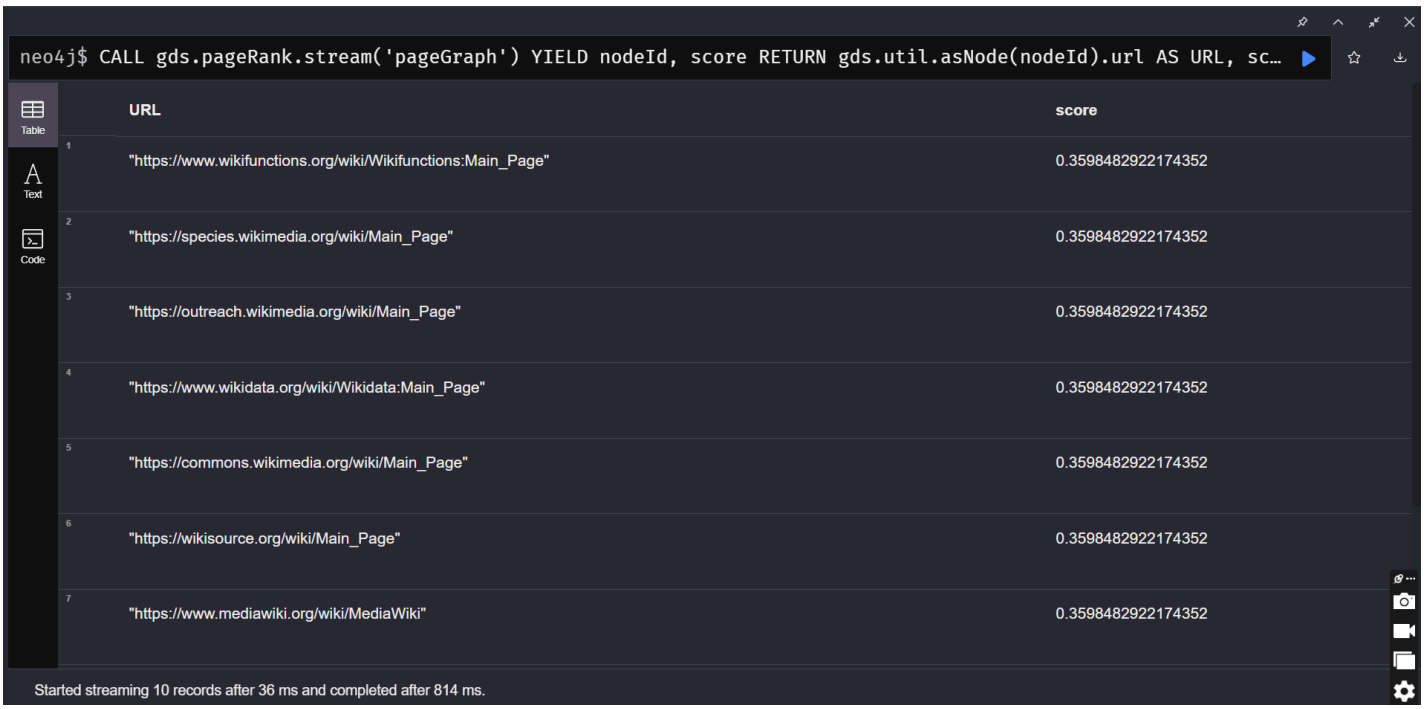

neo4j$ MATCH (p1:Page)-[r:LINKS_TO]→(p:Page) WITH p, COUNT(r) AS IncomingLinks ORDER BY IncomingLinks DESC LI…
Graph
Table
Text
Code

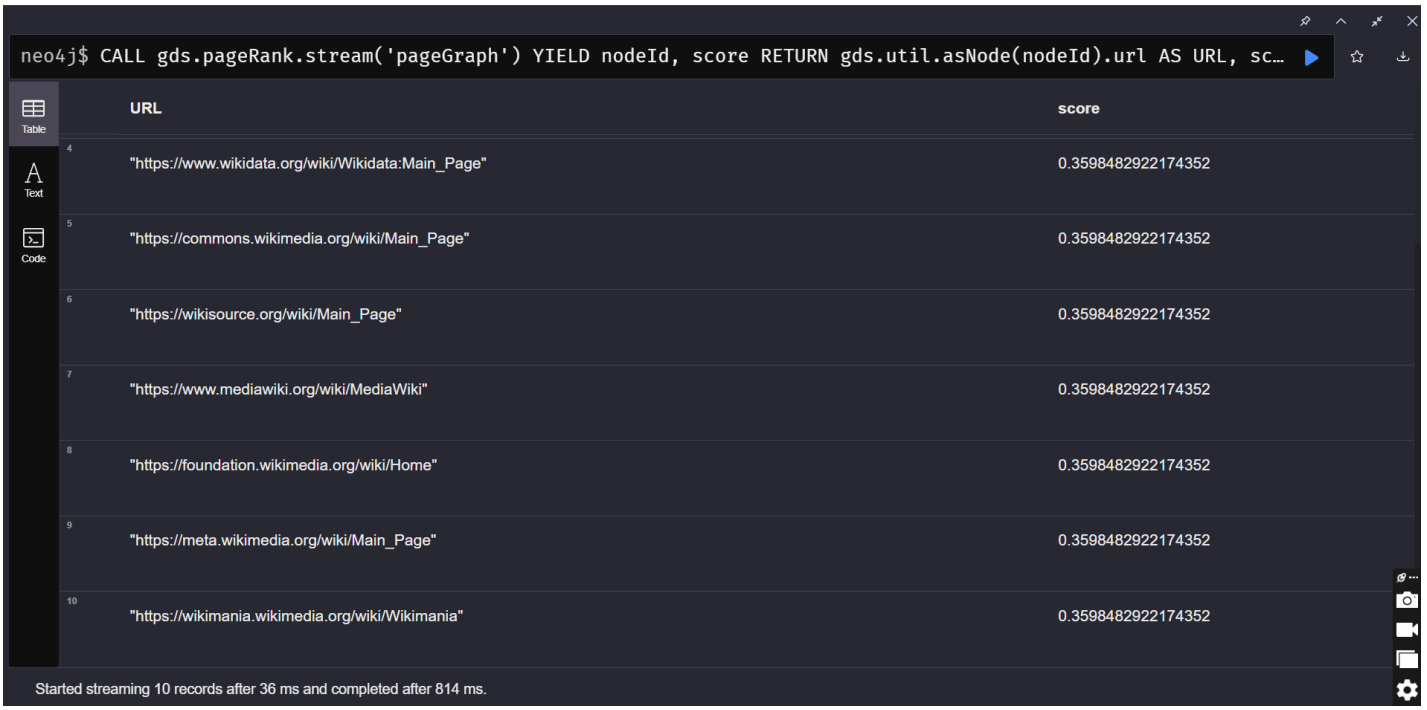
5. Identify Orphan Pages (No Incoming Links)



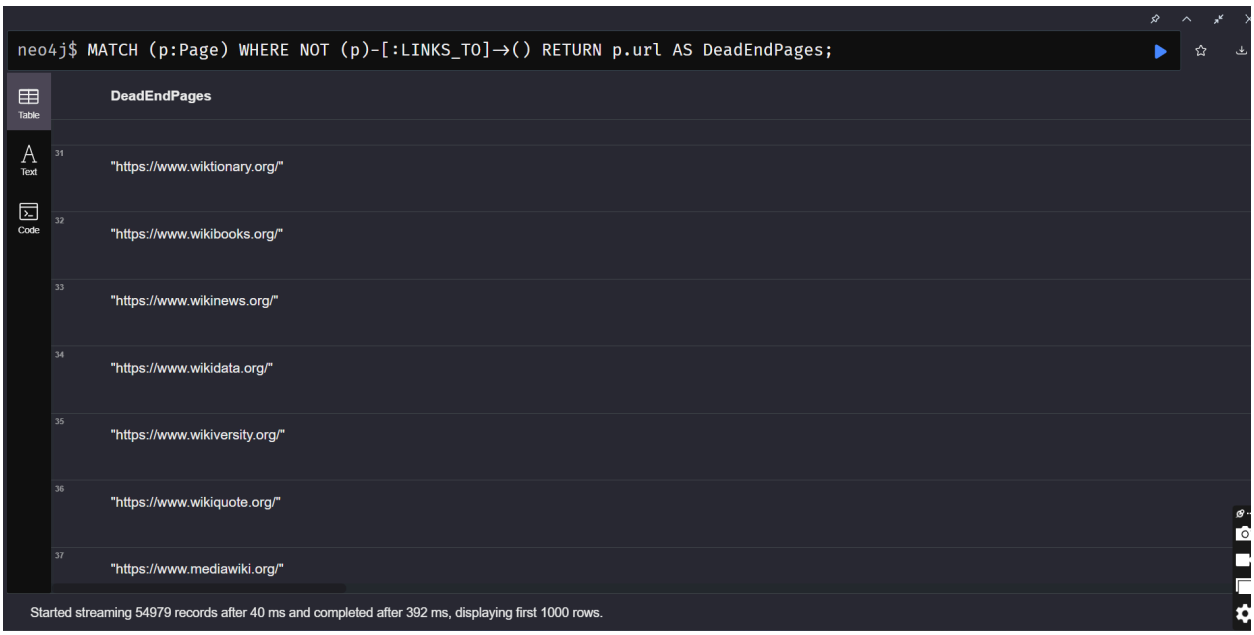


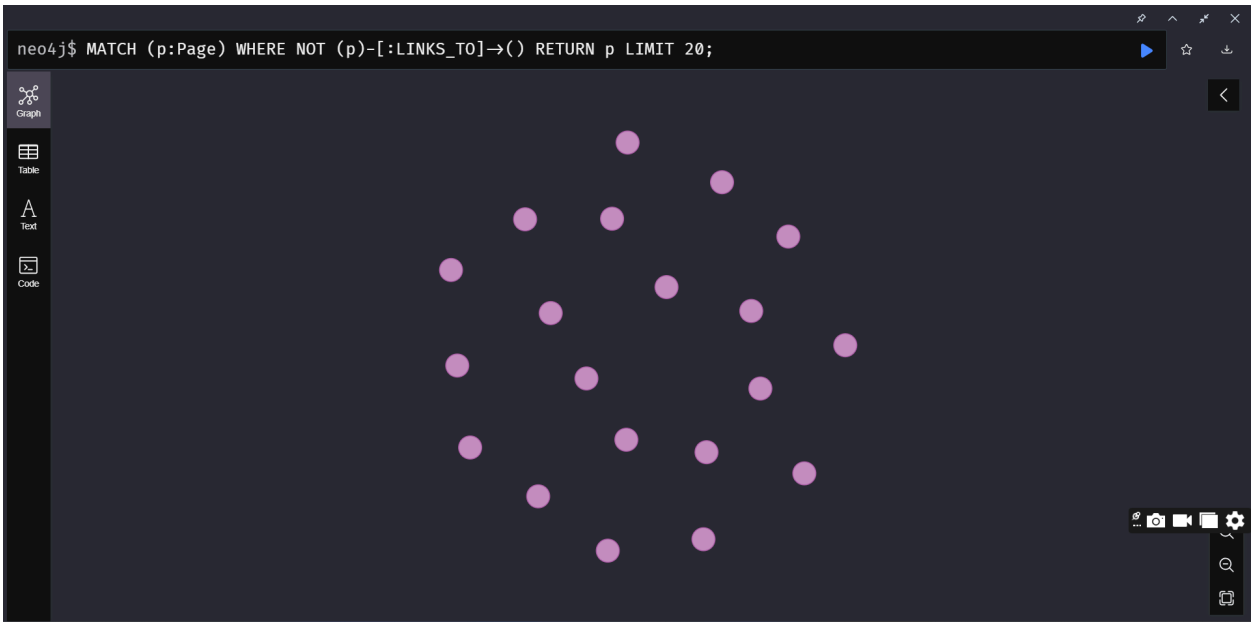
6. PageRank Analysis





7. Find Dead Ends (Pages with No Outgoing Links)





8. Analyze Link Density

4.2 Data Quality

During the crawling process, the following quality insights were observed:

- **Pages Without Titles:** 102
  These pages primarily included redirects, media files, and unsupported content types.
- **Unsupported Protocols Encountered:** 8
  Protocols like *mailto:* and *javascript:void(0)* were excluded from the graph representation.
- **Duplicate Links:** 5,217
  Repeated links were identified and handled appropriately to maintain data integrity.

4.3 PageRank Analysis

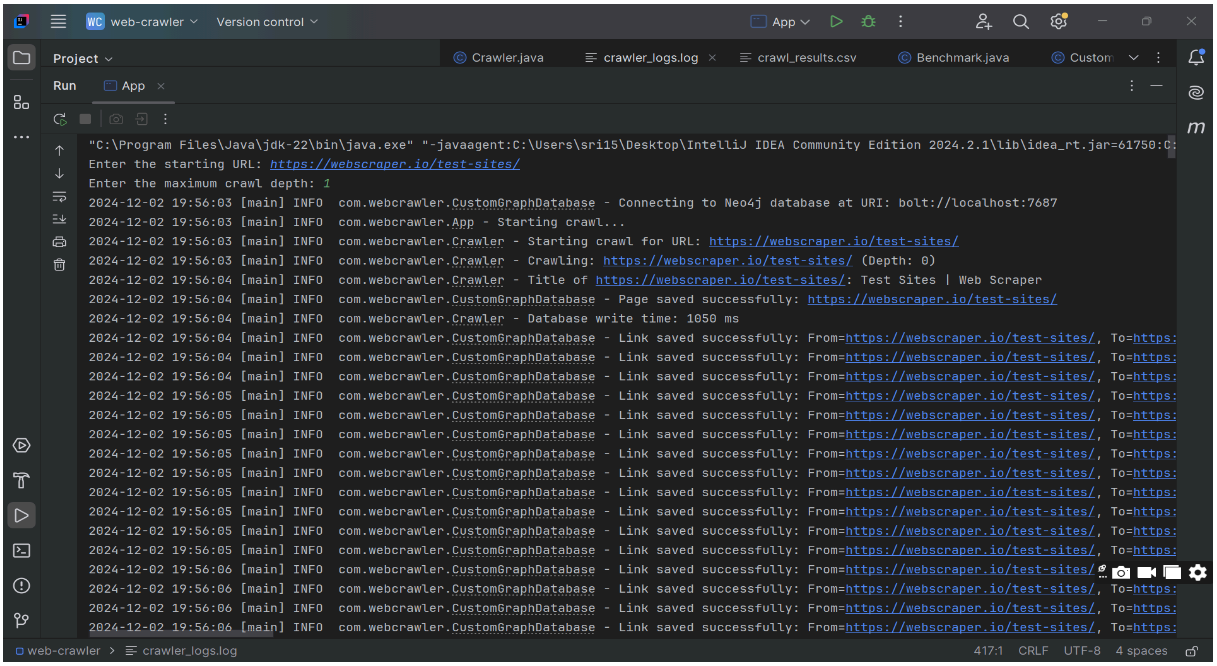
Top Pages by Importance (PageRank):

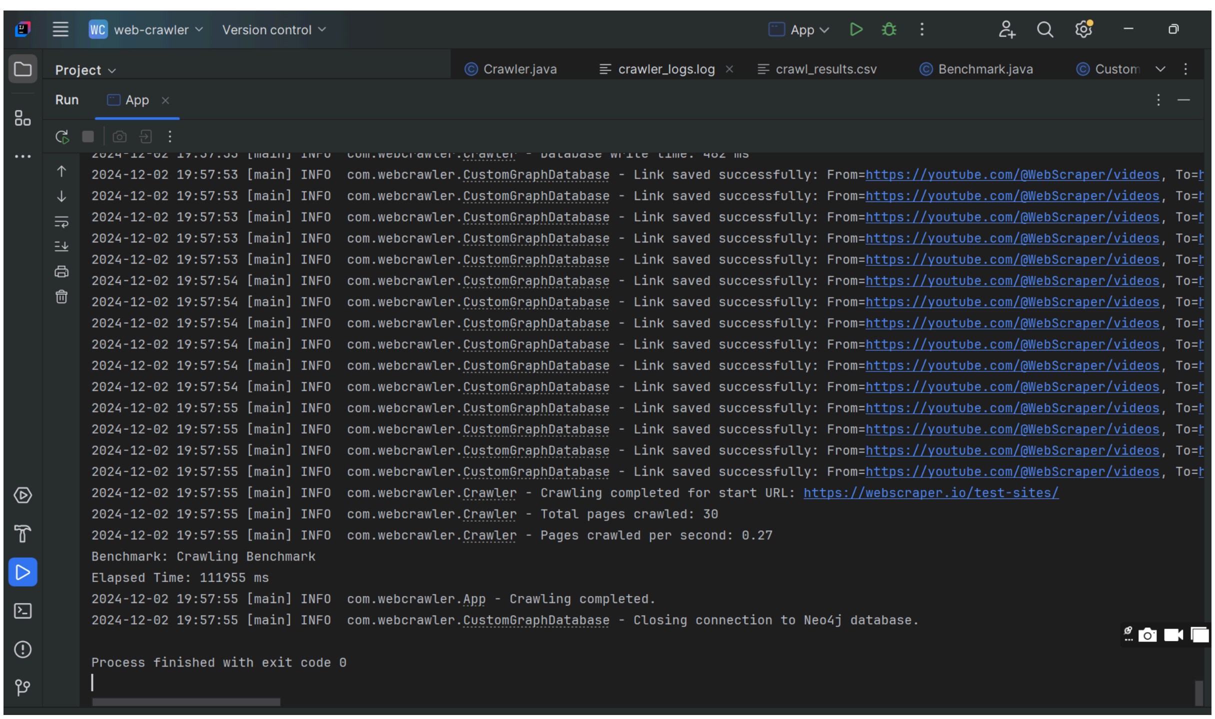
Rank	Page URL	PageRank
1	https://www.wikifunctions.org/wiki/Main_Page	0.359849822174352
2	https://species.wikimedia.org/wiki/Main_Page	0.359849822174352
3	https://outreach.wikimedia.org/wiki/Main_Page	0.359849822174352
4	https://www.wikidata.org/wiki/Wikidata:Main_Page	0.359849822174352
5	https://commons.wikimedia.org/wiki/Main_Page	0.359849822174352
6	https://wikisource.org/wiki/Main_Page	0.359849822174352
7	https://www.mediawiki.org/wiki/MediaWiki	0.359849822174352

4.4 Crawling Metrics and Performance Analysis

Output and Results from Code Execution:

We used **https://webscraper.io/test-sites/**, a lightweight and structured website designed for web scraping practice. It was chosen for its small size and quick response, allowing us to efficiently test the speed and performance of our crawler within a manageable timeframe.





Total pages crawled: 30

Pages crawled per second: 0.27

5. Challenges

1. **HTTPS Certificate Validation**:
    - Resolved SSL errors for pages with invalid certificates by implementing an SSL bypass.
2. **Rate-Limiting and Access Restrictions**:
    - Encountered rate-limiting during crawling; optimized request intervals to avoid bans.
3. **Neo4j Connection Issues**:
    - Managed authentication and password reset challenges effectively.
4. **Concurrency Challenges**:
    - Debugged and resolved race conditions in asynchronous processing.

---

6. Testing and Validation

6.1 Unit Testing:
- Verified BFS traversal logic, URL validation, and database writes.

6.2 Integration Testing:
- Ensured seamless interaction between the crawler and Neo4j database.

6.3 Edge Case Handling:
- Addressed:
    - Missing titles.
    - Invalid or malformed URLs.
    - Pages with circular links.

---

7. Insights

1. **Efficient Website Analysis**:
   The project demonstrates the application of ***asynchronous BFS traversal*** and graph databases like Neo4j for scalable and efficient website crawling, enabling the analysis of large, interconnected structures in real-time.
2. **Practical Application of Data Structures**:
   By implementing queues for BFS traversal and using a graph database for storage, the project integrates core concepts of program structure and analysis, showcasing their real-world relevance in handling complex datasets.
3. **Error Handling and Flexibility**:
   Robust error-handling mechanisms ensured the crawler could handle diverse scenarios, such as unsupported protocols and missing titles, making it adaptable for various domains and practical use cases.

4. **Visualization and Data Representation**:
   The use of Neo4j to represent web pages as nodes and links as edges provides a visual and analytical perspective, aligning with the course's focus on structured analysis and insight generation.

---

8. Conclusion

The web crawler successfully extracted and analyzed the web graph of **https://www.wikipedia.org/**. The implementation achieved the following:

- Successfully implemented an **asynchronous BFS traversal** with multithreading for efficient web crawling.
- Effectively represented web graphs in **Neo4j**, capturing the structure and interconnections of websites.
- Addressed diverse challenges, including unsupported protocols, missing titles, and recursive link handling.
- Demonstrated practical applications of web crawling, graph databases, and analytics.
- Highlighted the relevance of scalable web data processing for real-world data-driven insights.