Table of Contents

Objectives

The aim of this project is to develop a simple Crawler- based search engine that demonstrates the main features of a search engine (web crawling, indexing and ranking) and the interaction between them. Also, it is intended to enhance your Java programming skills.

Search Engine Modules

Web Crawler [20%]

The web crawler is a software agent that collects documents from the web. The crawler starts with a list of URL addresses (seed set). It downloads the documents identified by these URLs and extracts hyper-links from them. The extracted URLs are added to the list of URLs to be downloaded. Thus, web crawling is a recursive process.

Care should be taken when implementing web crawlers. At minimum, you have to take care of the following issues:

- The crawler must not visit the same PAGE more than once. Also, normalize URLs and check if they are referring to the same page.
- The crawler can only crawl documents of specific types (HTML is sufficient for the project).
- The crawler must maintain its state so that it can, if interrupted, be started again to crawl the documents on the list without revisiting documents that have been previously downloaded.
- Some web administrators choose to exclude some pages from the search such as their web pages check for Robot.txt.
- Provide a multithreaded crawler implementation where the user can control the number of threads before starting the crawler. Use synchronization appropriately if needed.
- Take Care of the choice of your seeds.
- Number of Crawled pages is 5000 pages (for the sake of the project). If you collect them during one crawl or many crawls, both are accepted.
- Use an appropriate data structure to determine the order of page visits.
- The crawler is independent program or process than the Indexer.

Indexer [30%]

The output of web crawling process is a set of downloaded HTML documents. To respond to user queries fast enough, the contents of these documents have to be indexed in a data structure that stores the words contained in each document and their importance (e.g., whether they are in the title, in a header or in plain text).This data structure has to satisfy the following properties:

- Persistence: The index has to be maintained in secondary storage. You can implement your own file structure or use a database.
- Fast Retrieval: The index must be optimized for responding to queries like:
    - The set of documents containing a specific word (or set of words)
- Incremental Update: It must be possible to update an existing index with a set of newly crawled HTML documents.
- When designing the Indexer, consider how you will store your result by looking ahead on Ranker and Searching.

Query Processor [10%]

This module receives search queries, performs necessary preprocessing and searches the index for relevant documents. Retrieve documents containing words that share the same stem with those in the search query. For example, the search query “travel” should match (with lower degree) the words “traveler”, “traveling” … etc.

Phrase Searching [5%]

Search engines will generally search for words as phrases when quotation marks are placed around the phrase.

Ranker [20%]

The ranker module sorts documents based on their popularity and relevance to the search query.

1. **Relevance**
   Relevance is a relation between the query words and the result page and could be calculated in several ways such as tf-idf of the query word in the result page or simply whether the query word appeared in the title, heading, or body. And then you aggregate the scores from all query words to produce the final page relevance score.

2. **Popularity**
   Popularity is a measure for the importance of any web page regardless the requested query. You can use pagerank algorithm (as explained in the lecture) or other ranking algorithms to calculate each page popularity.

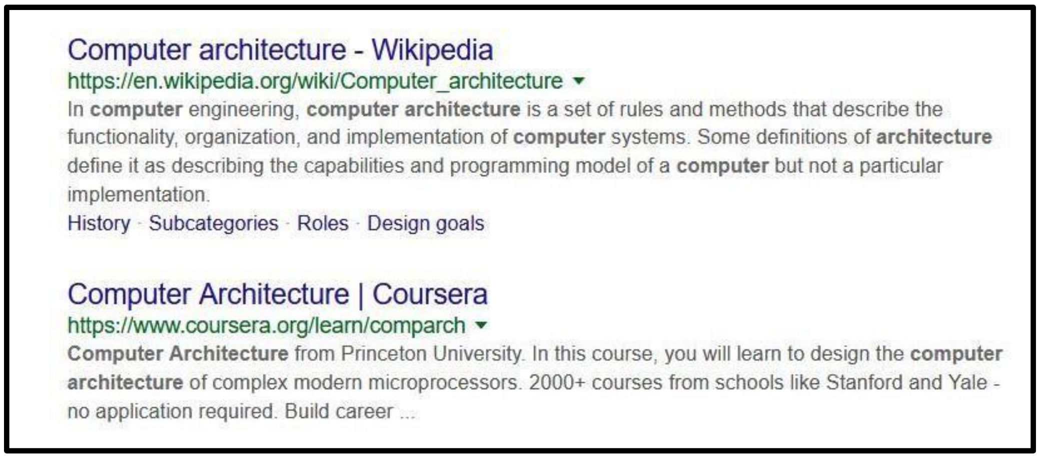
??

Grading criteria (20%): 5% for efficiency, 10% for correctness/understanding, 5% for implementation

Android/Web Interface [15%]

You have to implement a web interface or a mobile interface for your search engine. It is your choice.

- This interface receives user queries and displays the resulting pages returned by the engine
- The result appears with **snippets** of the text containing queries words. The output should look like google/bing's results page:



Computer architecture - Wikipedia
https://en.wikipedia.org/wiki/Computer_architecture ▾
In **computer** engineering, **computer architecture** is a set of rules and methods that describe the functionality, organization, and implementation of **computer** systems. Some definitions of **architecture** define it as describing the capabilities and programming model of a **computer** but not a particular implementation.
History · Subcategories · Roles · Design goals

Computer Architecture | Coursera
https://www.coursera.org/learn/comparch ▾
Computer Architecture from Princeton University. In this course, you will learn to design the **computer architecture** of complex modern microprocessors. 2000+ courses from schools like Stanford and Yale - no application required. Build career ...

Note: if there are errors just in showing results on the interface, show them on console or file to take a partial grade rather than zero.

- Pagination of results (i.e., if you got 200 results, they should appear on 20 pages, each page with 10 results)
- Add suggestion mechanism that stores queries submitted by all users. As the user types a new query, your web application should suggest interactively popular completions to that query.

If your web interface is having any problem showing results, you should make sure that the query retrieval is working correctly by displaying the results in a file/console to make sure that everything is working except for the interface.

Grading Criteria (15%): 5% for neatness, 5% for correctness/implementation, 5% for suggestion mechanism

Bonus

There are additional features that you can add to your project for bonus grades

- Voice Recognition Search. Use a voice query instead of a typed one.
- Efficient Innovation on indexing & ranking.

Implementation and Deliverables

Deadlines

Delivery	Deadline	Discussion
The Whole Project	Week 13	A graded discussion.

In all phases, A link to your github or gitless source project:

- Your code files
- A readme.txt, explaining how to run your code
- A members.txt containing the names and IDs of each student in the group and whether you are semester or credit
- A PDF file containing any algorithms you've used

Also provide a zipped folder containing the same material, Name the zipped folder:
Team_<team number>_<semester>.zip (replace semester with credit, replace your team number).

Teams

Work in groups of 3~4 maximum.

Implementation

- Implementation is done mainly in **Java**. This Java rule is for implementing the main modules like: crawler, indexer, etc., but for the frontend of the user interface, use the languages/libraries you want.
- It is your responsibility to select the best technique and tool that enhances the project performance

Libraries and Packages Regulations

You can use a library only if you followed **all** of the points below:

- You **can**'t use a library that do the whole module functionality or a high percentage of it (i.e., you can't use a library to do the whole crawler module), so using libraries must be for a small percentage of the module functionality. If you are not sure if a library usage is allowed or not, contact instructors for approval.
- It is allowed to use a library to parse HTML, normalize URLs, or to parse Robot.txt.
- You can use any database management system **BUT** you should be the one creating the scheme and adding the data. Don't use a database to do the whole indexing (i.e., don't give it the document and let it do the indexing for you.
- You are responsible for the library accuracy. (If it does a bad job, then it is your responsibility).
- You should understand how the library works.

Evaluation and Grading Criteria

- The project is graded as a whole and the discussion decides what is the grade of each student (there'sno piggybacking)
  - 50% of the project grade will be on requirement completeness.
  - 40% of the project grade will be on understanding how everything works in your system.
  - 10% of the project grade will be on code organization, neatness, using source control and naming convention.
- Any delay in delivery will be penalized by losing 10% of the grade for each late day
- Code must be original, and must not be copied or shared from any other source, except as provided by the class instructors.
- Note: A plagiarized project means ZERO in the project and deduct some grades from the other coursework. Please, do bit put yourself in such situation.

~~~~~ Good Luck ~~~~~
~~~~~