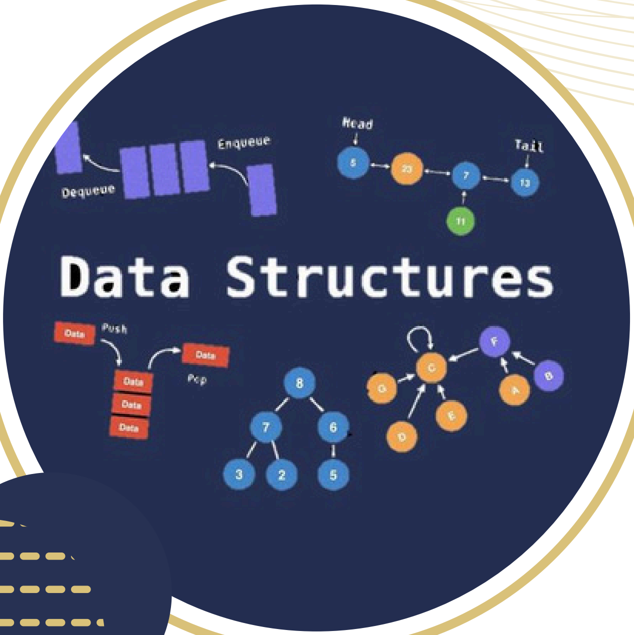
King Saud University
College of Computer and Information Sciences

CSC 212 PROJECT
SIMPLE SEARCH ENGINE



Prepared by :
Jana Alsuwailem
444200816
Walaa Mohammed
444200088
Section: 71140

Supervised by:
Dr. Amani Alajlan

I. EXPLANATION OF THE SEARCH ENGINE:

Objective:

The aim of this project is to develop a simple and efficient search engine capable of retrieving, processing, and ranking documents based on user queries. The search engine implements three retrieval methods:

1. Index Retrieval using a list of lists.
2. Inverted Index Retrieval using a list of lists.
3. Inverted Index Retrieval using a Binary Search Tree (BST).

Additionally, the system supports:

- Boolean retrieval (AND and OR operations).
- Ranked retrieval based on term frequency.

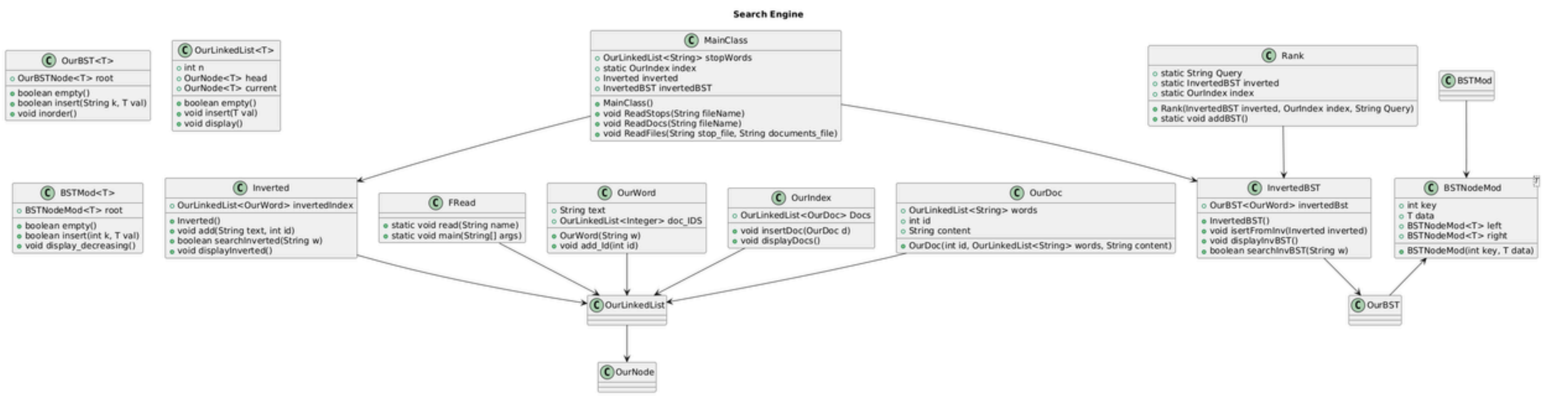
Workflow:

1. Data Input:
    - Documents and stop words are read from files.
    - Documents are tokenized and indexed.
2. Indexing:
    - Terms and document IDs are added to the respective indexing structures (list-based or BST-based).
3. Query Processing:
    - Users input a query, which is processed to retrieve matching documents.
4. Output:
    - The results are displayed, showing document IDs, ranked by relevance if applicable.

Features:

1. Search Methods:
2. Offers multiple retrieval methods to suit different performance needs.
3. Boolean and Ranked Retrieval:
4. Combines logical operators with ranking for flexible search capabilities.
5. Efficiency.
6. Preprocessing and inverted indexing reduce the time required to retrieve documents.
7. Scalability.
8. The use of BST enables the system to handle large datasets efficiently.

2. CLASS DIAGRAM:


Search Engine
OurBST<T>
OurBSTNode<T> root
boolean empty()
boolean insert(String k, T val)
void inorder()
OurLinkedList<T>
int n
OurNode<T> head
OurNode<T> current
boolean empty()
void insert(T val)
void display()
MainClass
OurLinkedList<String> stopWords
static OurIndex index
Inverted inverted
InvertedBST invertedBST
MainClass()
void ReadStops(String fileName)
void ReadDocs(String fileName)
void ReadFiles(String stop_file, String documents_file)
Rank
static String Query
static InvertedBST inverted
static OurIndex index
Rank(InvertedBST inverted, OurIndex index, String Query)
static void addBST()
BSTMod
BSTMod<T>
BSTNodeMod<T> root
boolean empty()
boolean insert(int k, T val)
void display_decreasing()
Inverted
OurLinkedList<OurWord> invertedIndex
Inverted()
void add(String text, int id)
boolean searchInverted(String w)
void displayInverted()
FRead
static void read(String name)
static void main(String[] args)
OurWord
String text
OurLinkedList<Integer> doc_IDS
OurWord(String w)
void add_Id(int id)
OurIndex
OurLinkedList<OurDoc> Docs
void insertDoc(OurDoc d)
void displayDocs()
OurDoc
OurLinkedList<String> words
int id
String content
OurDoc(int id, OurLinkedList<String> words, String content)
InvertedBST
OurBST<OurWord> invertedBst
InvertedBST()
void isertFromInv(Inverted inverted)
void displayInvBST()
boolean searchInvBST(String w)
BSTNodeMod
int key
T data
BSTNodeMod<T> left
BSTNodeMod<T> right
BSTNodeMod(int key, T data)
OurLinkedList
OurBST
OurNode

3. PERFORMANCE ANALYSIS:

I. Index Retrieval (Using List of List):

Method Name	Purpose	Time Complexity	Space Complexity
OurIndex.getDocByTerm (String t)	Retrieves document IDs containing a specific term by scanning through all documents.	O(n×m)	O(n×m)
OurIndex.displayDocs()	Displays all documents stored in the index, iterating through every document and its words.	O(n×m)	O(n×m)

Explanation:

- n: Number of documents.
- m: Average number of words per document.
- Time complexity is high because the method scans every document's word list to find matches.

2. Inverted Index Retrieval (Using List of List):

Method Name	Purpose	Time Complexity	Space Complexity
Inverted.add(String text, int id)	Adds a term to the inverted index if it doesn't exist, iterating through the list.	O(k+d)	O(k+d)
Inverted.searchInverted (String w)	Searches for a term in the inverted index list.	O(k)	O(k)
Inverted.displayInverted()	Displays all terms in the inverted index along with their document IDs.	O(k+d)	O(k+d)

Explanation:

- k: Number of unique terms.
- d: Total number of document IDs across all terms.
- Using a list structure, the time complexity for searching terms and retrieving documents depends on the length of the list and the associated document IDs.

3. Inverted Index Retrieval (Using BST):

Method Name	Purpose	Time Complexity	Space Complexity
InvertedBST.add(String text, int id)	Adds a term and document ID to the BST-based inverted index.	O(logk+d)	O(k+d)
InvertedBST.searchInvBST(String w)	Searches for a term in the BST-based inverted index.	O(logk)	O(k)
InvertedBST.displayInvBST()	Displays all terms in the BST and their associated document IDs in-order.	O(k+d)	O(k+d)

Explanation:

- The BST structure reduces the time complexity for searching terms to O(logk), making it faster than a list-based inverted index.

4. Ranking

Method Name	Purpose	Time Complexity	Space Complexity
Rank.addBST()	Ranks documents based on term frequency in the query, inserting scores into a BST.	O(d×m×logd)	O(d)
Rank.displayDocsByScore()	Displays ranked documents in descending order of their scores.	O(d)	O(d)
Rank.getDocByScore(OurDoc d, String Query)	Calculates the score for a document based on term frequency.	O(m)	O(1)

Explanation:

- d: Number of documents in the query results.
- mmm: Average number of words per document.
- Ranking involves calculating scores and inserting them into a BST for efficient sorting.

5. Query Processing:

Index-Based Query Methods:

Method Name	Purpose	Time Complexity	Space Complexity
QIndex.And(String Q)	Processes AND queries using the index by finding common documents for all terms.	O(n×m)	O(n)
QIndex.Or(String Q)	Processes OR queries using the index by merging document lists for all terms.	O(n×m)	O(n)

Inverted Index Query Methods:

Method Name	Purpose	Time Complexity	Space Complexity
QInverted.And(String Q)	Processes AND queries using the inverted index by intersecting document ID lists.	O(k+d)	O(d)
QInverted.Or(String Q)	Processes OR queries using the inverted index by merging document ID lists.	O(k+d)	O(d)

BST-Based Query Methods:

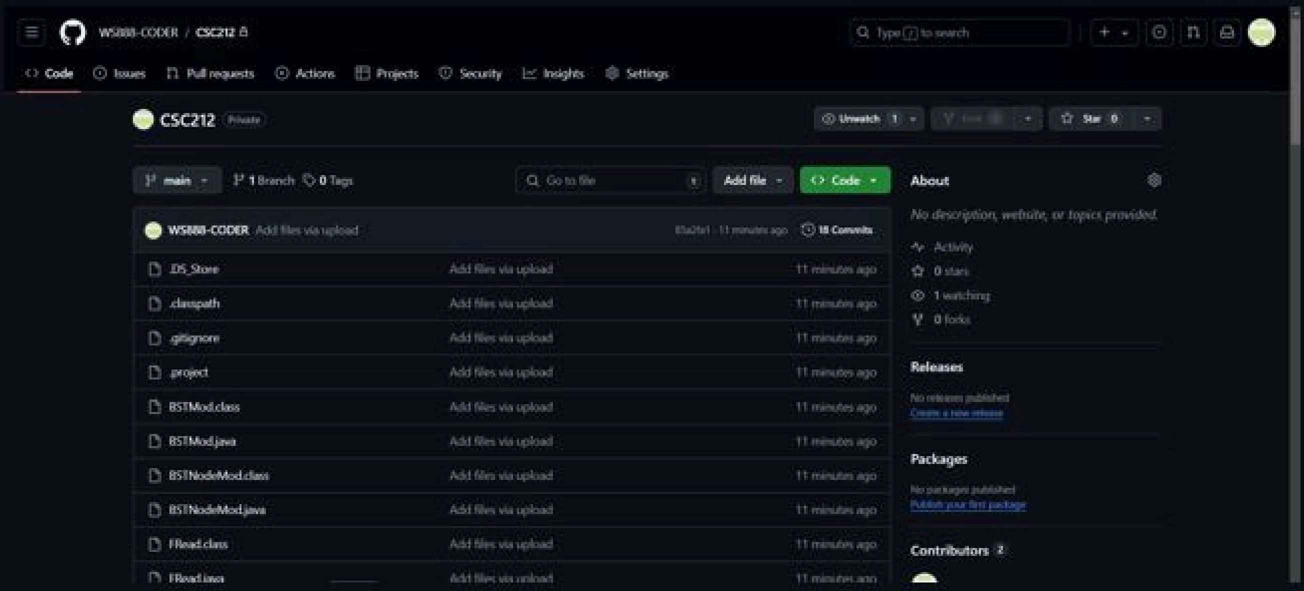
Method Name	Purpose	Time Complexity	Space Complexity
QBST.And(String Q)	Processes AND queries using the BST by intersecting document ID lists.	O(logk+d)	O(d)
QBST.Or(String Q)	Processes OR queries using the BST by merging document ID lists.	O(logk+d)	O(d)

Comparison Table:

Method Name	Time Complexity	Space Complexity	Notes
Index Retrieval	O(n×m)	O(n×m)	Inefficient for large datasets due to full document scan for every query.
Inverted Index (List of List)	O(k+d)	O(k+d)	Improves efficiency by mapping terms to document IDs.
Inverted Index (BST)	O(logk+d)	O(k+d)	Most efficient retrieval method due to logarithmic search.
Query Processing (Index)	O(n×m)	O(n)	Handles Boolean queries but is slow for large datasets.
Query Processing (Inverted Index)	O(k+d)	O(d)	Faster for Boolean queries due to direct access to term document lists.
Query Processing (BST)	O(logk+d)	O(d)	Optimized query performance with faster term lookups.
Ranking	O(d×m×logd)	O(d)	Adds ranking overhead but provides sorted query results.

The Inverted Index Retrieval (Using BST) is the most efficient method for large datasets due to its logarithmic search time and optimized query performance. While the List-based Inverted Index also offers significant improvements over the basic index retrieval, it is slower than the BST approach for large vocabularies or datasets.
For ranking, the overhead is justified as it enhances the search engine’s usability by providing sorted and relevant results. However, for small datasets or less frequent queries, the basic index retrieval may still suffice due to its simplicity.

4. GETHUB REPOSITRY LINK:



CSC212 Private

1 Branch 0 Tags

WS888-CODER Add files via upload · 11 minutes ago · 18 Commits

.DS_Store	Add files via upload	11 minutes ago
.classpath	Add files via upload	11 minutes ago
.gitignore	Add files via upload	11 minutes ago
.project	Add files via upload	11 minutes ago
BSTMod.class	Add files via upload	11 minutes ago
BSTMod.java	Add files via upload	11 minutes ago
BSTNodeMod.class	Add files via upload	11 minutes ago
BSTNodeMod.java	Add files via upload	11 minutes ago
FRead.class	Add files via upload	11 minutes ago
FRead.java	Add files via upload	11 minutes ago

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 2

https://github.com/WS888-CODER/CSC212.git