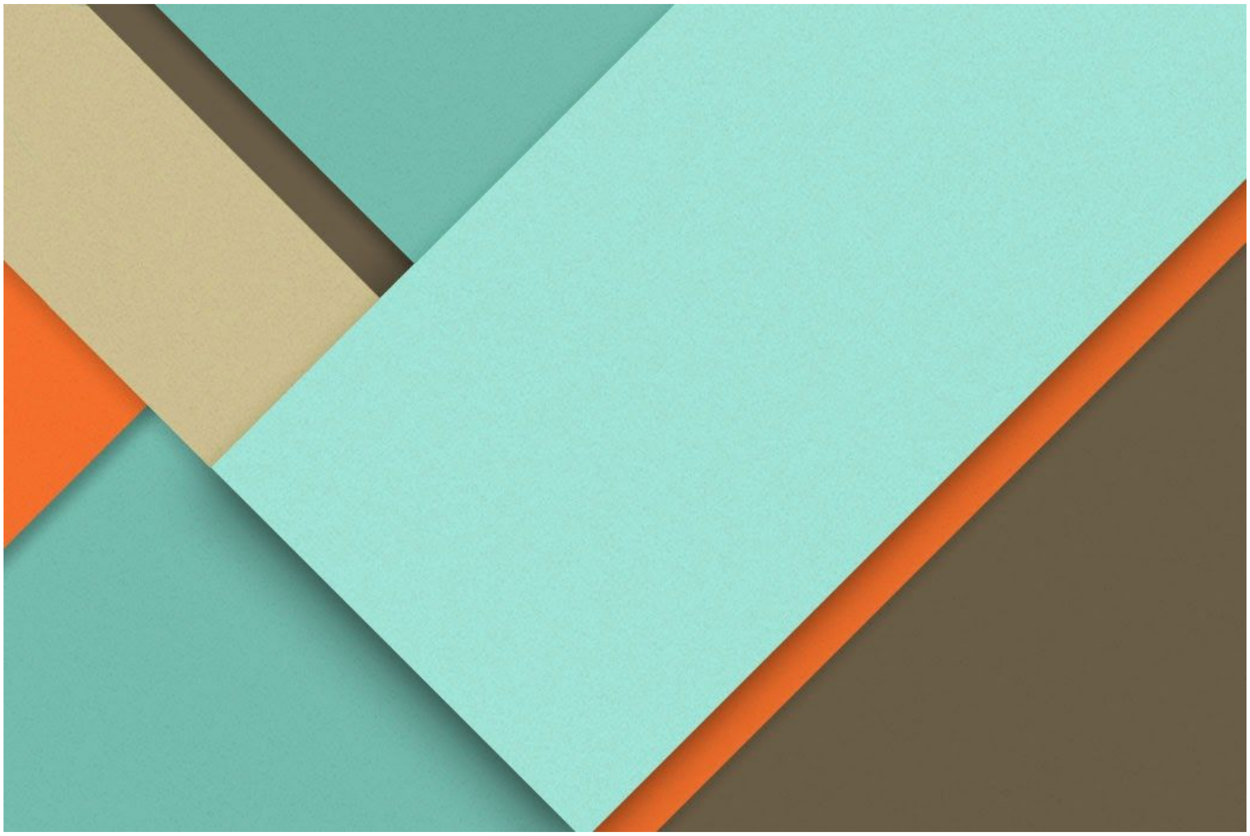

Data Structures 2 - Lab 3

B-Tree and Indexing

Afnan Mousa	15
Enas Morsy	20
Sara Mohammad	31
Shimaa kamal	34

Overview

In this assignment, It’s required to implement a B-tree and a simple search engine application that utilizes the B-Tree for data indexing.

B-Tree

B-trees are balanced search trees designed to work well on disks or other direct access secondary storage devices. Its nodes can store multiple keys and have many children.

Search Engine

Given a search query of one or multiple words, should return the matched documents and order them based on the frequency of the query words in each document.

Search engine code design:

- The main concept in the search engine application is that there is only one B-tree storing all required data, this tree nodes consist of:
  - **Keys:** each word in every document represents a key in this tree without any repeated keys.
  - **Values:** list of “ISearchResult” interface; for every key, word, there is a list having ids of all documents containing this word and its rank in every id.
- When a new document path is inserted using the “indexWebPage “ function our engine takes this document and extracts all words from it, then it maps every word to its rank, which is a number showing how many times this word has been repeated in that document, and saves them as the form of “ISearchResult”.
- By looping through this map:
  - if a specific word is already a key in B-tree it adds this “ISearchResult” to the key value which is a list.
  - If a specific word isn’t a key in any node it creates a new node, makes this word its key and pushes ”ISearchResult” to be the first element in its value list.
- The engine uses this B-tree to traverse through the documents’ words making any modification supported by ISearchEngine functions.

Time and space complexity

B-Tree:

- **getMinimumDegree()**
  Time: O(1)

- **search()**
  Time: O(log n)
  Space: Just searching, no need for space.

- **insert()**
  Time: O(log n)
  Space: O(n)

- **delete()**
  Time: O(log n)
  Space: O(n)

Search Engine

- **indexWebPage(**filePath**)**

  Time: O(m*n*log(n)), where m is no. of documents and n is no. of keys to be indexed and log n is for insertion.

  Space: O(d* n), where d is no. of documents and n is no. of words.

- **indexDirectory(**directoryPath**)**

  Time: O(f*m*n*log(n)), where f is no. of files in the directory, m is no of documents and n is no. of keys to be indexed.

  Space: O(f*d* n)

- **deleteWebPage(**filePath**)**

  Time: O(m*n*log(n)), where m is no. of documents and n is no. of keys to be deleted from the B-Tree.

  Space: O(d* n), where d is no. of documents and n is no. of words.

- **searchByWordWithRanking(**word**)**

  Same as searching in BTree
  Time: O(log n)
  Space: O(n)

- **searchByMultipleWordWithRanking(**sentence**)**

  Time: O(m log n), where m is no. of words in the sentence.
  Space: O(n+m)