Alexandria University,

Faculty of engineering,

Data structures.

B tree and search engine.

# Names:


- Mohamed Mohamed Abdlhakem (43).
- Mohamed Salah Osman (41).

# Problem statement:

It is 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. Unlike the red-black trees, B-tree nodes can store multiple keys and have many children. If an internal B-tree node x contains x.n keys, then x has x.n + 1 children. The keys in node x serve as dividing points separating the range of keys handled by x into x.n + 1 sub-ranges, each handled by one child of x.

## Simple Search Engine

You will be given a set of Wikipedia documents in the XML format and you are required to implement a simple search engine that given a search query of one or multiple words you should return the matched documents and order them based on the frequency of the query words in each wiki document, please check the requirements section for more details.

# Code design:

## Main data structures:

Hash map < string, integer > saves the words as keys and their frequencies as values for each document.

Hash map < string, Hash map < string, integer > > saves the id of the document as a key and the value is a hash map.

## Algorithms description:

- Insertion of b tree: insertion uses a downwards pass if it gets to any full node it splits it to maintain that it will not violate the b tree properties.
- Deletion: it works as follows it deletes the specified key and if it violates the property of b-tree, it fixes it in a recursive manner by rebalancing the tree by borrowing a spare key from immediate sibling or by merging with the parent key.

## Flow of algorithm of search engine:

1. In case of indexing the parser first reads the xml file and get the documents and calculate the occurrence of each word in each document and saves them in a hash table .then, the algorithm insert the word as a key passed to the b tree and the value is a hash table contains the id of each document contains this word and its frequency.
2. Indexing of a directory is done recursively on the folder till files is found and it starts indexing each file.
3. Searching is done as b tree ordinary search, once a word is found in the b tree, a hash map of ids of documents and the frequency of the word in each document is passed as a return value.
4. Searching by multiple words return the min number of occurrences of all these words in the b-tree.

## Time and space complexity:

1. B tree search: O (log n).
2. B tree delete: O (log n).
3. B tree insert: O (log n), where n is the number of nodes in the tree.
4. Search engine indexing of xml file.
    - Parsing: O (m * n), m number of documents and n is the average number of words in all documents in xml file.
    - Indexing: O (n * m * log k), m number of documents, n is the average number of words in all documents in xml file and k is the number of nodes in the b tree.