Search Engine using hashing, sorting and linked list

Overview:

A search engine like google, yahoo, duckduckgo etc. is a program that searches for and identifies items in a database that correspond to keywords or characters specified by the user, used especially for finding particular sites. Following this, we have made a simple and basic program that uses hashing, linked list and sorting. Other details for explained below:

Concept of data structure:

Firstly, let's understand the concept of each data structure:

Hashing:

Hashing is a technique used to map large data sets of variable length to a fixed-size data set. It uses a hash function that converts the input data to a numerical index or key that is used to address or retrieve data. In the context of search engines, hashing can be used to index web pages based on their URLs or keywords.

Linked list:

A linked list is a data structure that consists of a sequence of nodes, where each node contains data and a reference (or pointer) to the next node in the sequence. Linked lists can be used to store and manipulate lists of web pages based on their relevance to a search query.

Sorting:

Sorting is the process of arranging items in a list or collection in a specific order, such as alphabetical or numerical order. Sorting can be used in search engines to rank web pages based on their relevance to a search query.

With the help of these three data structures, we'll make our project.

Implementation and building a search engine:

To build a search engine using hashing, linked list, and sorting in C++, you will first need to collect data about web pages using a web crawler. A web crawler is a program that automatically traverses the web and collects data about web pages. You can use a web crawler to collect data about web pages and store it in a database.

Once you have collected data about web pages, you can implement a hash table data structure to index web pages based on their URLs or keywords. A hash table is a data structure that maps a key to a value using a hash function. In the context of a search engine, you can use a hash function to map web page URLs or keywords to a unique index in the hash table. This allows you to quickly retrieve web pages that match a search query.

Next, you can implement a linked list data structure to store and manipulate lists of web pages based on their relevance to a search query. A linked list is a data structure that consists of a sequence of nodes, where each node contains data and a reference (or pointer) to the next node in the sequence. In the context of a search engine, you can use a linked list to store web pages that match a search query and to manipulate the order of the web pages based on their relevance score.

To rank web pages based on their relevance to a search query, you can implement a sorting algorithm. A sorting algorithm is a procedure that takes a list of items and arranges them in a specific order, such as alphabetical or numerical order. In the context of a search engine, you can use a sorting algorithm to rank web pages based on their relevance score. For example, you can use a sorting algorithm such as quicksort or mergesort to sort web pages based on their relevance score.

Finally, you can implement a search function that takes a search query as input and returns a list of web pages ranked by relevance to the search query. The search function

should use the hash table to find web pages that match the search query and the linked list and sorting algorithm to rank the web pages. The search function should also consider other factors such as user experience and performance, such as returning only the top N results or using pagination to display search results in a more user-friendly way.

Overall, building a search engine using hashing, linked list, and sorting in C++ involves combining different data structures and algorithms to create a system that can efficiently search and retrieve web pages based on user queries.

Code:

```
//First, we have created header files of iostream, string (because string is used instead of int and characters) unordered_map(for unsorted array) and list(to show everything in a sorted array).
#include <iostream>
#include <string>
#include <unordered_map>
#include <list>

using namespace std;
//Made a class Page to store the url and rank of the website.
class Page {
public:
    string url;
    int rank;
//used keyword *this* to point to variable
    Page(string url, int rank) {
        this->url = url;
        this->rank = rank;
```

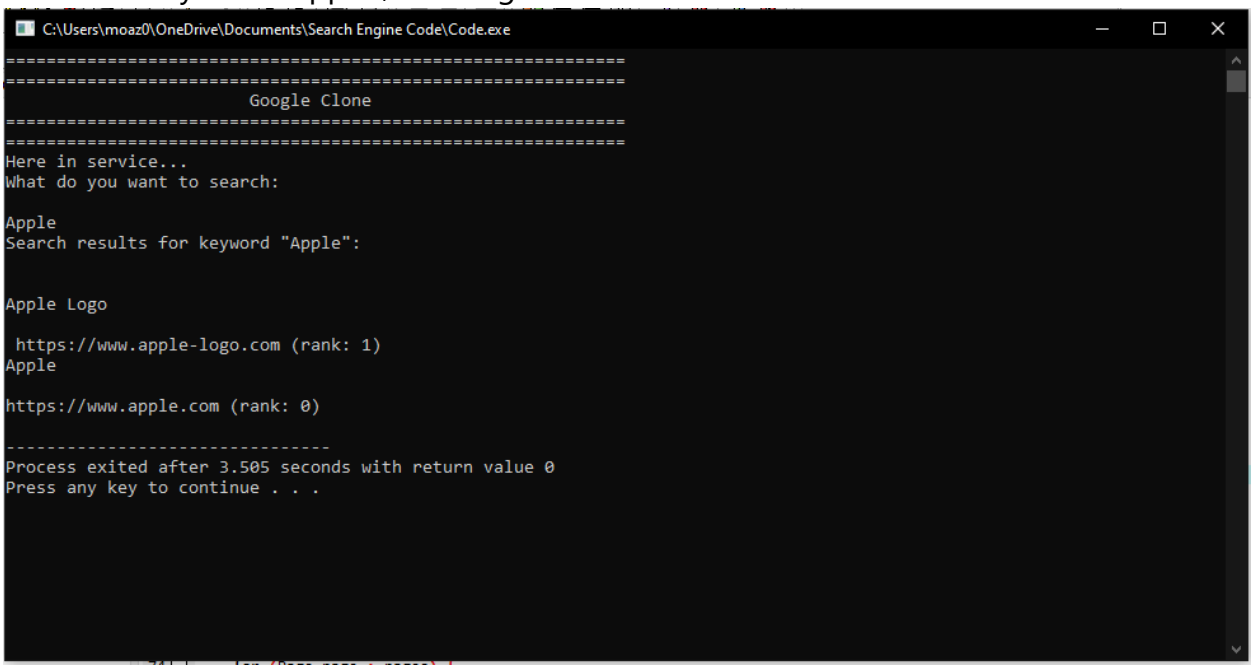
```
    }
};

class SearchEngine {
private:
    unordered_map<string, list<Page>> index;

public:
    void addPage(string keyword, Page page) {
        if (index.find(keyword) == index.end()) {
            list<Page> pages;
            pages.push_back(page);
            index[keyword] = pages;
        } else {
            index[keyword].push_back(page);
        }
    }

    list<Page> search(string keyword) {
        if (index.find(keyword) == index.end()) {
            return list<Page>();
        } else {
            return index[keyword];
        }
    }
```

```
    void sortPages(list<Page>& pages) {
        pages.sort([](Page a, Page b) {
            return a.rank > b.rank;
        });
    }
};

int main() {
    SearchEngine searchEngine;

cout<<"==========================================================
==========="<<endl;

cout<<"==========================================================
==========="<<endl;
        cout<<"\t\t\tGoogle Clone "<<endl;
        cout<<"==========================================================
==============="<<endl;
        cout<<"==========================================================
==============="<<endl;
    // Add some sample pages to the search engine

    searchEngine.addPage("Apple", Page("Apple\n\nhttps://www.apple.com",0));
    searchEngine.addPage("Apple", Page("Apple Logo\n\n https://www.apple-
logo.com",1));
    searchEngine.addPage("Iphone", Page("Iphone-
Apple\n\nhttps://www.apple.com/iphone", 2));
```

```
    // Search for pages containing the keywords that you have given in program
    string keyword;
    cout<<"Here in service...\nWhat do you want to search:\n\n";
    cin>>keyword;
    list<Page> pages = searchEngine.search(keyword);

    // Sort the pages by rank
    searchEngine.sortPages(pages);

    // Print the sorted pages
    cout << "Search results for keyword \"" << keyword << "\":\n\n" << endl;
    for (Page page : pages) {
        cout << page.url << " (rank: " << page.rank << ")" << endl;
    }

    return 0;
}
```

Output:

If I enter the keyword “Apple” in its output. Then the information which we have to stored on keyword “Apple”, will be given.



Now let’s search something else: