ĐỒ ÁN CUỐI KÌ
THỰC HÀNH KĨ THUẬT LẬP TRÌNH

1. Danh sách thành viên

- Nguyễn Phạm Nhật Huy MSSV: 20120012
- Trần Ngọc Đô MSSV: 20120057

2. Giới thiệu

Chương trình `Search Engine` là chương trình quản lý dữ liệu thư viện nhằm mục đích tìm kiếm những văn bản thuộc thư viện liên quan đến từ khoá nhập vào một cách nhanh chóng.

Chương trình có một số chức năng chính sau:

- Xây dựng dữ liệu cho thư viện
- Truy vấn thông tin
- Cập nhật dữ liệu khi thêm văn bản vào thư viện hoặc xoá văn bản khỏi thư viện
- Lưu dữ liệu thư viện để tái sử dụng mà không cần xây dựng lại mỗi lần mở chương trình

3. Tổ chức dữ liệu

Các văn bản của thư viện được mã hoá theo định dạng `UTF-16 LE` nên phần xử lý của chương trình sử dụng một giá trị `uint16_t` để biểu diễn một ký tự Unicode.

1. Xây dựng cây tập tin

Dựa vào cấu trúc của thư viện, cây tập tin được tổ chức bằng struct Library. Việc xây dựng cây tập tin có thể thực hiện dễ dàng bằng cách câu lệnh có sẵn của <Windows.h>.

```
struct Library
{
    LinkedList folderlist;
    LinkedList* filelist;
    char** charfilelist;
    char** charfolderlist;
    int totalfolder;
    int totalfile;
};
```

Trong đó:

- `totalfolder` là số lượng thư mục chủ đề của thư viện
- `totalfile` là số lượng văn bản của thư viện
- `folderlist` là danh sách liên kết chứa các thư mục chủ đề
- `filelist` là mảng động dẫn đến các danh sách liên kết đại diện cho một thư mục chủ đề, chứa các văn bản thuộc thư mục đó
- `charfilelist` là mảng động chuyển đổi từ `folderlist`
- `charfolderlist` là mảng động chuyển đổi từ `filelist`

Danh sách liên kết (Linked List) được sử dụng vì dễ dàng thực hiện thao tác thêm/xoá phần tử.

Mảng động được sử dụng để truy cập phần tử nhanh chóng.

2. Thành phần rút trích

Các dữ liệu được lưu trữ như sau:

- stopword.txt chứa danh sách các stopword.
- asciiform.txt chứa ánh xạ từ Unicode sang Ascii.
- uppertolower.txt chứa ánh xạ từ chữ in hoa sang chữ thường.
- delimiter.txt chứa ánh xạ từ các ký tự không phải chữ cái hoặc số sang kí tự khoảng trắng.

Nguồn tham khảo:

- Stopword: https://github.com/stopwords/vietnamese-stopwords/blob/master/vietnamese-stopwords.txt
- Ánh xạ Unicode sang Ascii, ánh xạ chữ in hoa sang chữ thường: https://www.ibm.com/docs/en/i/7.2?topic=tables-unicode-lowercase-uppercase-conversion-mapping-table
- Các ký tự không phải chữ cái hoặc số: Tập dữ liệu

```
struct CharacterMap
{
    uint16_t key;
    uint16_t value;
};
struct Stopword
{
    uint16_t size;
    uint16_t* value;
};
struct ExtractComponents
{
    uint16_t nmlsize;
    CharacterMap* normalize;
    uint16_t asciisize;
    CharacterMap* asciimap;
    uint16_t delimsize;
    CharacterMap* delimiter;
    uint16_t stopwordsize;
    Stopword* stopword;
};
```

Trong đó:

- `nmlsize` là kích thước mảng động `normalize`
- `normalize` là mảng động tương ứng với uppertolower.txt
- `asciisize` là kích thước mảng động `asciimap`
- `asciimap` là mảng động tương ứng với asciiform.txt
- `delimsize` là kích thước mảng động `delimiter`
- `delimiter` là mảng động tương ứng với delimiter.txt
- `stopwordsize` là kích thước mảng động `stopword`
- `stopword` là mảng động tương ứng với stopword.txt

3. Rút trích dữ liệu

Các văn bản sau khi được tải lên bộ nhớ sẽ được rút trích nội dung chính bằng thuật toán tìm kiếm nhị phân. Sau đó dữ liệu này sẽ được lưu trong metadata.txt và thông tin của các văn bản cũng được lưu trong phần tử FileRank tương ứng.

```
struct FileRank
{
    int folderindex;
    int fileindex;
    int metadatapos;
    int size;
    int score;
};
```

Trong đó:

- `folderindex` là số thứ tự thư mục của văn bản trong mảng động `charfolderlist`
- `fileindex` là số thứ tự của văn bản trong mảng động `charfilelist`
- `metadatapos` là vị trí lưu dữ liệu văn bản trong metadata.txt

- `size` là kích thước lưu trữ
- `score` là điểm số để đánh giá độ liên quan

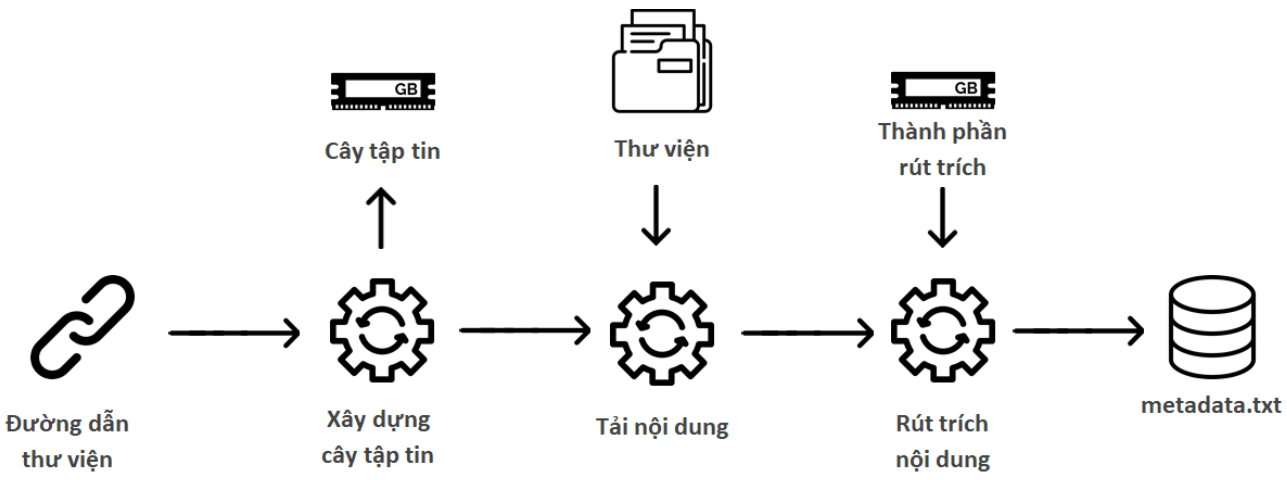
Thứ tự thực hiện rút trích:

- Chuyển văn bản về dạng chữ thường
- Chuyển các ký tự không phải chữ cái hoặc số về ký tự khoảng trắng
- Loại bỏ stopword
- Xoá khoảng trắng

Pseudo-code:

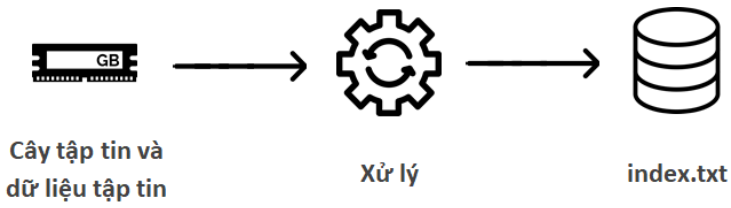
```
binarySearch(left, right, x)

while (left <= right)
     mid = (left + right) / 2
     if (x = value at mid )
          return mid
     else if (x > value at mid)
           left = mid + 1
      else
           right = mid - 1
```

4. Cách dữ liệu thư viên mới được xây dựng

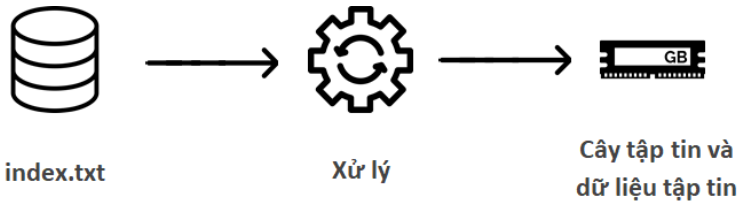


5. Lưu trữ dữ liệu

- Lưu dữ liệu thư viện



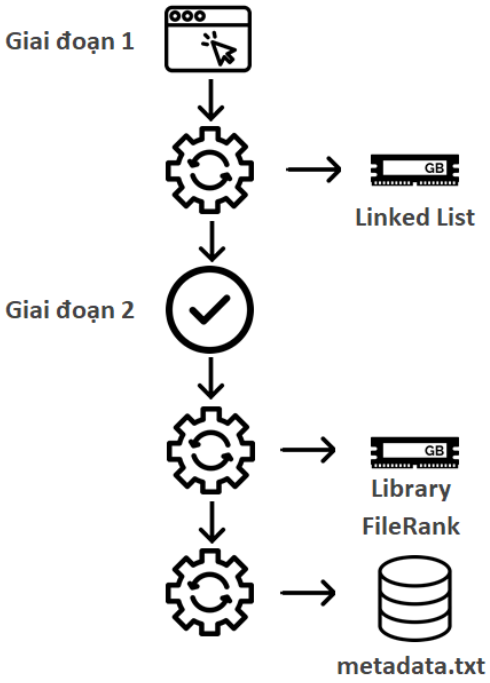
- Tải dữ liệu thư viện



6. Thêm/xoá văn bản

Được chia làm hai giai đoạn:

- Giai đoạn 1: Khi một thao tác được chương trình xử lý, folderlist và filelist sẽ được cập nhật vì thao tác này với LinkedList rất nhanh chóng.
- Giai đoạn 2: Sau khi quá trình cập nhật thư viện kết thúc, chương trình sẽ xây dựng lại các phần còn lại của struct Library và mảng động FileRank từ folderlist và filelist đã được cập nhật. Dữ liệu văn bản mới được thêm vào cũng được cập nhật.



4. Truy vấn thông tin

1. Xử lý đầu vào

Từ khoá sẽ được xử lý như cách rút trích văn bản. Sau đó, chương trình sẽ ghép từ khoá này với nội dung của từng văn bản trong `metadata.txt` để làm đầu vào cho quá trình tính toán.

Giả sử:

- `uint16_t* S` là từ khoá đã được xử lý.
- `uint16_t* T` là nội dung chính của văn bản được lưu trong `metadata.txt`.

Đầu vào cho quá trình tính toán là mảng động `M` được ghép theo công thức sau: `S + (uint16_t)65535 + T`. Trong đó giá trị 65535 có ý nghĩa là ký tự Unicode không thể xuất hiện văn bản hay từ khoá nhập vào từ người dùng. Giá trị này được chọn từ nhận xét rằng giá trị 65279 được dùng để đánh dấu tập tin UTF16 – LE là giá trị lớn nhất có thể có trong văn bản.

Ngoài ra, chương trình chuyển mảng động `M` có được sang dạng Ascii và tính toán lại một lần nữa để xử lý trường hợp từ Tiếng Việt không dấu.

2. Tính toán độ liên quan

Thuật toán được sử dụng là thuật toán so khớp chuỗi Knuth - Morris - Pratt (KMP) với thời gian chạy tuyến tính.

Ý tưởng thuật toán KMP là khi tìm kiếm sự xuất hiện của một "từ" W trong một "xâu văn bản" S, bản thân "từ" W cho ta đầy đủ thông tin để xác định vị trí bắt đầu của ký tự so sánh tiếp theo, do đó bỏ qua quá trình kiểm tra lại các ký tự đã so sánh trước đó.

Pseudo-code:

```
KMP(P, T):
q = 0
i = 0
while (i < n) // P[1, … , q] = T[i – q + 1, … , i]
     if (P[q + 1] = T[i + 1])
          q = q + 1
          i = i + 1
          if (q = m)
               output “i – q is a match”
               q = π(q) //slide the pattern to the right
     else // a mismatch occurred
          if (q = 0)
               i = i + 1
          else
               q = π(q)
```

Ngoài ra, chương trình còn tối ưu bộ nhớ cho mảng π từ nhận xét rằng giá trị lớn nhất của mảng này chỉ bằng độ dài từ khoá.

3. Sắp xếp thứ hạng

Thuật toán sắp xếp được sử dụng là thuật toán sắp xếp trộn Merge Sort với thời gian xử lý là `N.log(N)` và bộ nhớ `N`.

Merge Sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành hai nửa. Tiếp tục chia mỗi nửa này thành hai nửa bé hơn cho tới khi mỗi nửa con chỉ có một phần tử. Sau đó gộp lần lượt các nửa đó thành mảng đã sắp xếp.

Pseudo-code:

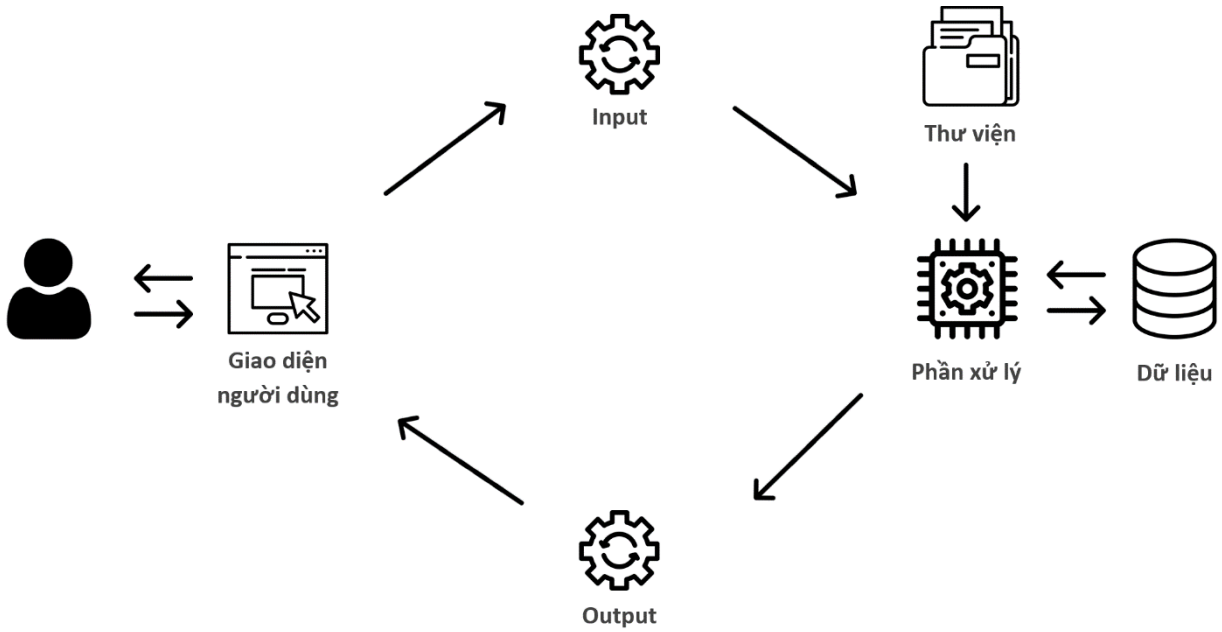
```
MergeSort(arr, left, right):
    if left > right
        return
    mid = (left + right) / 2
    mergeSort(arr, left, mid)
    mergeSort(arr, mid + 1, right)
    merge(arr, left, mid, right)
```

5. Giao diện người dùng

Giao diện người dùng được viết bằng `C++/CLI` trên nền `.NET Framework 4.6.1`.

Giao diện người dùng có chức năng:

- Thể hiện trực quan các chức năng của chương trình
- Chuyển đổi giữa các kiểu dữ liệu của ngôn ngữ lập trình `C++/CLI` và kiểu dữ liệu của ngôn ngữ lập trình `C` để tương tác với phần xử lý



Cách chương trình hoạt động