

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO TỔNG KẾT ĐỒ ÁN
KỸ THUẬT LẬP TRÌNH
ĐỀ TÀI: Máy tìm kiếm - Search Engine

Giảng viên hướng dẫn: TS Võ Hoài Việt

Lớp: 20CTT1TN

Thành viên thực hiện:

- 20120131 – Nguyễn Văn Lộc
- 20120536 – Võ Trọng Nghĩa

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 6-7 NĂM 2021

Mục lục

1	Lời nói đầu	4
2	Tổ chức chương trình	6
2.1	Các files header	6
2.2	Các files source	6
3	Các tập tin siêu dữ liệu (metadata)	7
3.1	TF và IDF	7
3.1.1	TF	7
3.1.2	IDF	8
3.2	Tạo các tập tin siêu dữ liệu	10
3.2.1	Yêu cầu cần tạo các tập tin siêu dữ liệu	10
3.2.2	Tạo các tập tin siêu dữ liệu	10
4	Xử lí dữ liệu tìm kiếm đầu vào	14
5	Quá trình hoạt động của chương trình	16
5.1	Lần chạy đầu tiên	16
5.2	Load dữ liệu vào RAM	16
5.3	Menu của chương trình	17
5.3.1	Thêm một thư mục vào dataset folder	17
5.3.2	Đổi sang một dataset folder khác	18
5.3.3	Tìm kiếm	18
5.3.4	Thông tin về người phát triển của chương trình	20
5.3.5	Thoát khỏi chương trình	21
6	Tài nguyên chương trình tiêu tốn	22
6.1	Cấu hình máy dùng để chạy thử chương trình	22
6.2	Thời gian chạy thử chương trình	22
6.3	Bộ nhớ và CPU tiêu tốn	24
7	Nhận xét về chương trình và dự án	25
7.1	Về thời gian chạy của từng quá trình	25
7.2	Về bộ nhớ	25
7.3	Về cấu trúc chương trình	25
7.4	Về quản lý dự án	25

Danh sách hình vẽ

1	Công cụ tìm kiếm Google	4
2	Công cụ tìm kiếm của Facebook	4
3	Kết quả khi gõ một từ khóa trên Google	4
4	Tổ chức các files mã nguồn trong chương trình	6
5	Các hàm xử lý cấu trúc TF	8
6	Các hàm xử lý cấu trúc IDF	10
7	Các hàm hỗ trợ tạo tập tin siêu dữ liệu trong <i>Utility.h</i>	11
8	Các hàm hỗ trợ tạo tập tin siêu dữ liệu trong <i>FileProgression.h</i>	11
9	Một ví dụ về tập tin <i>FolderList.txt</i>	12
10	Một ví dụ về các thư mục con trong thư mục <i>metadata</i>	12
11	Một ví dụ về các tập tin văn bản và tập tin IDF trong thư mục <i>metadata</i>	13
12	Một ví dụ về thư mục con của <i>metadata</i>	13
13	Các "punctuations" trong C++	14
14	Nhập tên của dataset folder ở lần chạy đầu tiên	16
15	Chương trình đang tạo các tập tin siêu dữ liệu cho lần chạy đầu tiên	16
16	Thông báo tạo thành công các tập tin siêu dữ liệu	16
17	Chương trình đang tiến hành load dữ liệu từ database	17
18	Menu của chương trình	17
19	Ví dụ thêm thư mục vào dataset folder	18
20	Thay đổi dataset folder	18
21	Nhập từ khóa tìm kiếm	18
22	Kết quả tìm kiếm	19
23	Mở tập tin kết quả với Notepad	19
24	Tính năng <i>next page</i>	20
25	Tính năng <i>previous page</i>	20
26	About us	21
27	Thoát khỏi chương trình	21
28	Cấu hình máy chạy thử chương trình.	22
29	Kích thước dataset folder	23
30	Thời gian tạo các tập tin siêu dữ liệu	23
31	Thời gian load dữ liệu vào RAM	23
32	Tài nguyên máy tính sử dụng khi tạo tập tin siêu dữ liệu	24
33	Tài nguyên máy tính sử dụng khi load dữ liệu vào RAM	24
34	Quản lý dự án bằng Github	25

Danh sách bảng

1	Bảng phân công thành viên	3
---	-------------------------------------	---

Bảng phân công thành viên

Thành viên	Công việc
Nguyễn Văn Lộc	<ul style="list-style-type: none"> Viết báo cáo tổng kết đồ án, Làm slide thuyết trình giới thiệu dự án, Viết hướng dẫn sử dụng, Test các chức năng của chương trình và thống kê thời gian chạy ở mỗi giai đoạn, Thiết kế UI, UX chính cho chương trình, Phụ trách chính phần nhập xuất metadata của chương trình, Nghiên cứu tài liệu về cách hoạt động của máy tìm kiếm, về công thức $TF*IDF$.
Võ Trọng Nghĩa	<ul style="list-style-type: none"> Quay demo cho chương trình, Test các chức năng của chương trình, sửa đổi và cấu trúc lại các đoạn code, thống kê bộ nhớ và lượng CPU tiêu tốn trong quá trình chạy, Nghiên cứu về các thuật toán sắp xếp phù hợp và sử dụng đến đa luồng để tối ưu thời gian chạy, Viết cấu trúc cho kết quả tìm được và các thành phần phụ quanh nó để hỗ trợ về thông tin, Viết thư viện hỗ trợ "<i>Utility.h</i>" đơn giản hóa quá trình giao tiếp lệnh giữa mã chương trình và hệ thống máy tính; xử lý các loại mảng động khác nhau, Viết các hàm chuẩn hóa và tìm tài liệu về cách đọc dạng file UTF16-LE, Phác họa chương trình, xây dựng cấu trúc TF, IDF, Đôn đốc dự án, Nghiên cứu tài liệu về cách hoạt động của máy tìm kiếm, về công thức $TF*IDF$.

Bảng 1: Bảng phân công thành viên

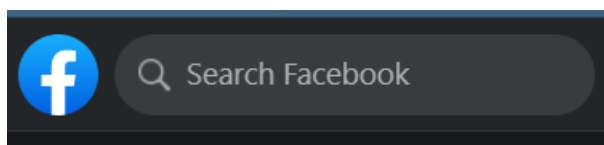
1 Lời nói đầu

Hiện nay, Internet đang trên đà phát triển nhanh chóng, đi kèm với đó là những tiện ích mà nó mang lại. Có thể kể đến các mạng xã hội, đã giúp cho việc giao tiếp giữa những con người cách xa nhau hàng ngàn kilometers không còn trở nên khó khăn. Hay trong thời điểm đại dịch COVID-19 đang hoành hành, nhu cầu học tập, làm việc trực tuyến trở nên tăng vọt, những ứng dụng giúp ích cho việc này như Google Meet, Zoom Cloud Meetings hay Microsoft Teams lên ngôi.

Ngoài ra, cùng với nhu cầu học tập, tham khảo, nghiên cứu cũng như giải trí với nguồn tài nguyên khổng lồ trên Internet, không thể không kể đến những **công cụ tìm kiếm** (máy tìm kiếm – search engine). Một ví dụ đơn giản, gần gũi với hầu hết người dùng Internet là Google, search engine được sáng lập bởi Larry Page và Sergey Brin; hay công cụ tìm kiếm của Facebook, được sử dụng khi ta có nhu cầu tìm một người bạn, một fanpage hay một hội nhóm nào đó.

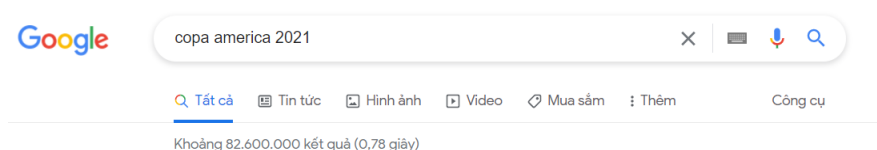


Hình 1: Công cụ tìm kiếm Google



Hình 2: Công cụ tìm kiếm của Facebook

Khi ta gõ một từ khóa đơn giản vào Google, ta có thể thu được hàng triệu kết quả chỉ trong một khoảng thời gian rất ngắn, cho thấy tính ưu việt của các thuật toán và cấu trúc dữ liệu mà Google sử dụng.
















Hình 3: Kết quả khi gõ một từ khóa trên Google

Thấy được sự tiện ích của các search engines đó, dưới sự hướng dẫn của TS Võ Hoài Việt, nhóm chúng em xin phép nghiên cứu và trình bày một công cụ tìm kiếm đơn giản, giúp rút trích đơn giản nội dung chính của văn bản tiếng Việt, tìm kiếm những văn bản có nội dung tương ứng với từ khóa do người dùng nhập vào, xếp hạng kết quả tìm kiếm theo mức độ liên quan đến từ khóa từ cao đến thấp.

Chúng em xin cảm ơn Thầy Việt vì những chỉ dẫn của Thầy trong suốt quá trình thực hiện đồ án này.

2 Tổ chức chương trình

Chương trình được viết bằng ngôn ngữ C/C++, được tổ chức thành các file header (.h) và file source (.cpp) như sau:

 FileProgression.h	12/07/2021 4:39 PM	C/C++ Header	1 KB
 IDF.h	12/07/2021 9:22 PM	C/C++ Header	1 KB
 Normalizer.h	12/07/2021 6:46 PM	C/C++ Header	1 KB
 TF.h	12/07/2021 4:39 PM	C/C++ Header	1 KB
 UI.h	10/07/2021 8:11 PM	C/C++ Header	1 KB
 Utility.h	12/07/2021 4:39 PM	C/C++ Header	2 KB
 FileProgression.cpp	12/07/2021 6:46 PM	C++ Source	11 KB
 IDF.cpp	12/07/2021 6:05 PM	C++ Source	3 KB
 Normalizer.cpp	12/07/2021 6:46 PM	C++ Source	3 KB
 SearchEngine.cpp	12/07/2021 6:19 PM	C++ Source	1 KB
 TF.cpp	12/07/2021 4:39 PM	C++ Source	3 KB
 UI.cpp	12/07/2021 5:06 PM	C++ Source	4 KB
 Utility.cpp	12/07/2021 4:39 PM	C++ Source	6 KB

Hình 4: Tổ chức các files mã nguồn trong chương trình

2.1 Các files header

- File header *FileProgression.h* chứa các hàm tham gia vào việc xử lý các files dữ liệu đầu vào.
- Các files header *IDF.h*, *TF.h* chứa các hàm tham gia vào việc tạo tập tin siêu dữ liệu (sẽ được nhắc đến trong phần sau).
- File header *Normalizer.h* giúp cho việc đơn giản hóa các kí tự tiếng Việt.
- File header *UI.h* chứa những hàm tạo nên giao diện người dùng trên console.
- File header *Utility.h* chứa các hàm hỗ trợ khác.

2.2 Các files source

- Các file source có tên tương ứng với các files header thì chứa mô tả về các hàm trong file header đó.
- File *SearchEngine.cpp* chứa hàm `main()`, hàm chính của chương trình.

3 Các tập tin siêu dữ liệu (metadata)

3.1 TF và IDF

TF – IDF là một kĩ thuật được sử dụng trong khai phá và truy hồi thông tin trong văn bản. Trọng số này được dùng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng càng cao, phụ thuộc vào số lần xuất hiện của một từ trong một văn bản và số văn bản mà từ đó xuất hiện trong tập dữ liệu. Ngoài ra, TF và IDF còn được ứng dụng để loại bỏ những stopwords (những từ lặp lại nhiều nhưng không có ý nghĩa) trong các bài toán tóm tắt và phân loại văn bản. [1]

3.1.1 TF

TF là viết tắt của từ tiếng Anh *Term Frequency*, có nghĩa là tần suất xuất hiện của từ trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, giá trị của TF thường được tính bằng thương số của số lần xuất hiện và số lượng từ trong văn bản. [1]

Trong phạm vi chương trình này, giá trị của TF (TF value) được tính như sau:

$$tf(t, d) = 0.5 + 0.5 \times \frac{f(t, d)}{\max\{f(w, d) : w \in d\}}, \quad (1)$$

trong đó

- $tf(t, d)$ là giá trị TF của từ t trong văn bản d ,
- $f(t, d)$ là số lần xuất hiện của từ t trong văn bản d ,
- $\max\{f(w, d) : w \in d\}$ là số lần xuất hiện lớn nhất của một từ trong văn bản d .

Trong chương trình, TF được biểu diễn bằng các cấu trúc như sau:

```
struct TF {
    string word;
    int count;
};

struct TF_list {
    int size;
    int capacity;
    int maxCount;
    TF* arrNorm;
};
```

Cấu trúc TF được dùng để biểu diễn cho **một từ**, với 2 trường là **word** và **count**.

- Trường **word** có kiểu **string**, là từ mà ta đang xét,
- Trường **count** có kiểu **int**, là số lần xuất hiện của từ đó trong văn bản.

Cấu trúc **TF_list** được dùng để biểu diễn cho **một văn bản**, với 4 trường là **size**, **capacity**, **maxCount**, **arrNorm**.

- Trường **size** có kiểu **int**, là số lượng TF có trong văn bản,

- Trường `capacity` có kiểu `int`, là số lượng TF tối đa có thể chứa, được khởi tạo và thay đổi phù hợp cho việc cấp phát động `arrNorm`,
- Trường `maxCount` có kiểu `int`, là số lượng tối đa của **một từ** trong văn bản đó,
- Trường `arrNorm` là một mảng động để chứa các TF trong văn bản.

Các hàm thao tác cho việc xử lý cấu trúc TF được viết trong file header `TF.h` như sau:

```
double getTFValue(TF_list, int);

void tfListInit(TF_list&);
void addTF(TF_list&, const TF&);
void loadTFList(const string&, TF_list&);
void saveTFList(const string&, TF_list&);
void freeTFList(TF_list&);

void tfListInput(TF_list&, string*, int);
```

Hình 5: Các hàm xử lý cấu trúc TF

- Hàm `double getTFValue(const TF_list List, const int i)` được dùng để tính giá trị TF của phần tử thứ `i` trong `list` theo công thức (1),
- Hàm `void TFListInit(TF_list& list)` khởi tạo một `TF_list` mới,
- Hàm `void addTF(TF_list &List, const TF& Data)` có tác dụng thêm TF Data vào `list`,
- Hàm `void loadTFList(const string& Filename, TF_list& List)` được sử dụng để load các cấu trúc TF từ tập tin có tên `fileName` vào `List`,
- Hàm `void saveTFList(const string& Filename, const TF_list List)` dùng để lưu các giá trị từ `List` vào tập tin có tên là `fileName`,
- Hàm `void FreeTFList(TF_list &List)` có tác dụng trả các vùng nhớ được cấp phát động sau khi sử dụng, tránh tình trạng memory leak,
- Hàm `void tfListInput(TF_list& List, string* Data, const int N)` có tác dụng tạo một `TF_list` có kích thước là `N` từ một mảng các `string` đã được sắp xếp tăng dần là `Data`.

3.1.2 IDF

IDF là viết tắt của từ tiếng Anh *Inverse Document Frequency*, tạm dịch là nghịch đảo tần suất của văn bản, giúp đánh giá tầm quan trọng của một từ. Khi ta tính toán TF, tất cả các từ được coi là có độ quan trọng như nhau. Nhưng một số stopwords như "is", "of", "that" trong tiếng Anh hay "à", "là", "đó", ... trong tiếng Việt thường xuất hiện rất nhiều lần nhưng độ quan trọng không cao. Như thế, chúng ta cần giảm độ quan trọng của những từ này xuống. [1] Trong phạm vi chương trình này, giá trị của IDF (IDF value) được tính như sau:

$$idf(t, D) = \log \frac{n_D}{n_d : d \in D \wedge t \in d}, \quad (2)$$

trong đó

- $idf(t, D)$ là giá trị IDF của từ t trong thư mục D ,
- n_D là tổng số văn bản trong thư mục D ,
- $n_d : d \in D \wedge t \in d$ là số văn bản nằm trong thư mục D và chứa từ t .

Logarithm cơ số 10 trong công thức không làm thay đổi tính tăng giảm của giá trị IDF của từ mà chỉ thu hẹp phạm vi giá trị của IDF. Việc sử dụng IDF nhằm giúp giá trị TF-IDF của một từ nhỏ hơn, vì chương trình sử dụng công thức tính giá trị TF-IDF của một từ trong một văn bản như sau:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D), \quad (3)$$

trong đó

- $tf(t, d)$ là giá trị TF của từ t trong văn bản d ,
- $idf(t, D)$ là giá trị IDF của từ t trong thư mục D .

Từ đó, ta có thể thấy rằng những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều ở một văn bản nhưng xuất hiện ít hoặc không xuất hiện ở các văn bản khác. Việc này giúp ta lọc ra những từ phổ biến và giữ lại những từ có giá trị cao, tạo thành tập các từ khóa của văn bản. [1]

Tương tự như TF, trong chương trình, IDF được biểu diễn bằng các cấu trúc như sau:

```
struct IDF {
    string word;
    int value;
};

struct IDF_list {
    int size;
    int capacity;
    int numFile;
    IDF* arrNorm;
};
```

Cấu trúc IDF được dùng để biểu diễn **một từ**, với 2 trường là **word** và **value**.

- Trường **word** có kiểu **string**, là từ mà ta đang xét,
- Trường **value** có kiểu **int**, là số lượng văn bản trong thư mục chứa từ đó.

Cấu trúc **IDF_list** được dùng để biểu diễn cho **một thư mục**, với 4 trường là **size**, **capacity**, **numFile** và **arrNorm**.

- Trường **size** có kiểu **int**, là số lượng IDF có trong thư mục,
- Trường **capacity** có kiểu **int**, là số lượng IDF tối đa có thể chứa, được khởi tạo và thay đổi phù hợp cho việc cấp phát động **arrNorm**,
- Trường **numFile** có kiểu **int**, là số lượng tập văn bản có trong thư mục,
- Trường **arrNorm** là một mảng động để chứa các **IDF** trong thư mục.

Các hàm thao tác cho việc xử lý cấu trúc IDF được viết trong file header *IDF.h* như sau:

```
double getIDFValue(IDF_list, int);

void idfListInit(IDF_list&);
void addIDF(IDF_list&, IDF);
void loadIDFList(const string&, IDF_list&);
void saveIDFList(const string&, IDF_list);
void freeIDFList(IDF_list&);

void idfListInput(IDF_list&, int, string*, int);
```

Hình 6: Các hàm xử lý cấu trúc IDF

- Hàm `double getIDFValue(const IDF_list List, const int i)` được dùng để tính giá trị IDF của phần tử thứ *i* trong *list* theo công thức (2),
- Hàm `void idfListInit(IDF_list &List)` khởi tạo một *IDF_list* mới,
- Hàm `void addIDF(IDF_list &List, IDF Data)` có tác dụng thêm *IDF Data* vào *list*,
- Hàm `void loadIDFList(const string& Filename, IDF_list &List)` được sử dụng để load các cấu trúc IDF từ tập tin có tên *fileName* vào *List*,
- Hàm `void saveIDFList(const string& Filename, const IDF_list List)` dùng để ghi các giá trị từ *List* vào tập tin có tên là *fileName*,
- Hàm `void freeIDFList(IDF_list &List)` có tác dụng trả các vùng nhớ được cấp phát động sau khi sử dụng, tránh tình trạng *memory leak*,
- Hàm `void idfListInput(IDF_list& List, const int NumFile, string* Data, const int N)` có tác dụng tạo một *IDF_list* cho một thư mục có *NumFile* tập tin từ một mảng động các *string* đã được sắp xếp tăng dần là *Data*.

3.2 Tạo các tập tin siêu dữ liệu

3.2.1 Yêu cầu cần tạo các tập tin siêu dữ liệu

Nếu tìm kiếm những từ khóa đầu vào một cách bình thường, tuần tự bằng cách duyệt từng văn bản thì sẽ tốn rất nhiều thời gian cho mỗi lần tìm kiếm, dẫn đến thời gian phản hồi kết quả lâu.

Do đó, ta cần phải rút trích các nội dung chính của từng văn bản, của từng thư mục để tạo ra các tập tin siêu dữ liệu rồi tìm kiếm trên các tập tin ấy.

3.2.2 Tạo các tập tin siêu dữ liệu

Hai header files *FileProgression.h* và *Utility.h* chứa các hàm giúp cho việc tạo và sử dụng các tập tin siêu dữ liệu.

```
void makeFolderWrapper(const string&);
void getFolderWrapper(const string&, const string&);
void getFileWrapper(const string&, const string&);
void deleteFileWrapper(const string&);
string extractPath(string);
```

Hình 7: Các hàm hỗ trợ tạo tập tin siêu dữ liệu trong *Utility.h*

```
bool isFirstTime();
void updateMetadata(string Path);
void createMetadata(const string& folderDataset);
void prepareFile(const string& FolderPath);
void loadToRAM();
void freeRAM();
void searchSentence(const string& Sentence);
```

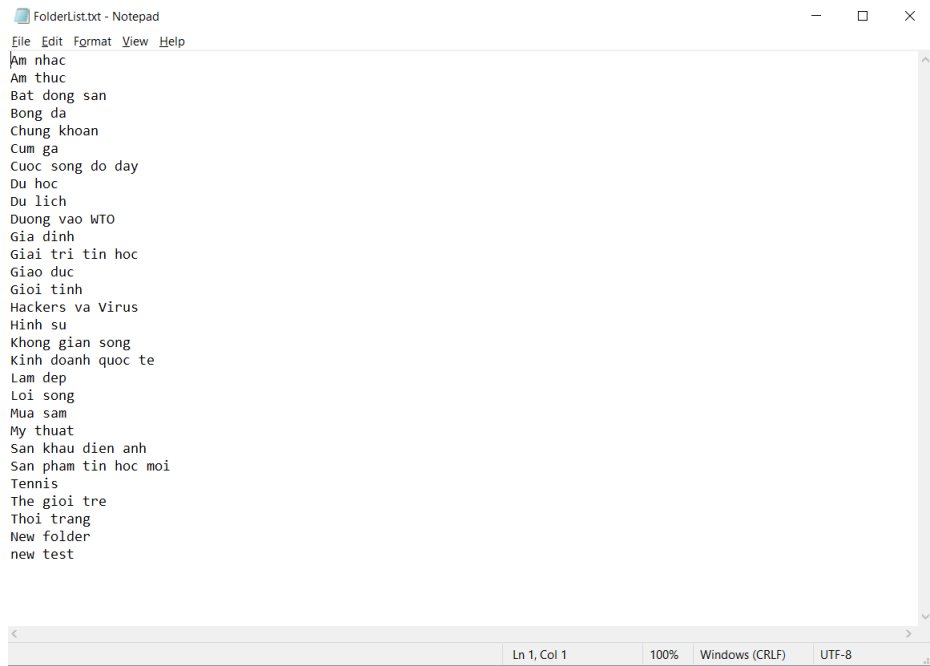
Hình 8: Các hàm hỗ trợ tạo tập tin siêu dữ liệu trong *FileProgression.h*

Quy trình tạo các tập tin siêu dữ liệu như sau:

- Trước tiên, ta dùng lệnh `dir` của *Windows* để lấy tên của các thư mục con trong thư mục chứa dữ liệu cần tìm (dataset folder), lưu vào tập tin *FolderList.txt*,
- Sau đó, ta tiếp tục dùng lệnh `dir` để lấy tên của từng tập tin trong các thư mục con, tên các tập tin của thư mục A được lưu vào tập tin *A.txt*,
- Tạo tập tin *index.txt* chứa đường dẫn tương đối từ chương trình đến từng tập tin văn bản,
- Ta chuyển các kí tự tiếng Việt từ dữ liệu đầu vào thành các kí tự Latin đơn giản bằng hàm `std::string VEconvert(const std::wstring& Source)` trong header *Normalize.h*,
- Tiếp theo đó, đối với từng tập tin văn bản trong các thư mục con, ta tạo các tập tin TF có tên tương ứng với tên của tập tin văn bản, ví dụ tập tin văn bản *A.txt* thì tập tin TF tương ứng sẽ có tên là *A.txt.tf*,
- Sau khi tạo xong các tập tin TF ứng với từng thư mục con, ta tạo tập tin IDF tương ứng với thư mục con đó, thư mục A tương ứng với tập tin IDF là *A.idf*,
- Sau đó, ta lặp lại quá trình này cho các thư mục con khác của dataset folder.

Sau khi tạo xong các tập tin siêu dữ liệu, ta sẽ thu được các kết quả như sau:

- Tập tin *FolderList.txt* lưu trữ nên các thư mục con trong dataset folder, cùng với tên của dataset folder ở dòng cuối cùng.

Hình 9: Một ví dụ về tập tin *FolderList.txt*

- Thư mục metadata chứa các tập tin siêu dữ liệu sẽ có các thư mục cùng tên với các thư mục con trong dataset folder, các tập tin văn bản chứa tên của từng tập tin trong các thư mục con, và các tập tin IDF tương ứng với các thư mục.

Am nhac	12/07/2021 4:46 PM	File folder
Am thuc	12/07/2021 4:46 PM	File folder
Bat dong san	12/07/2021 4:46 PM	File folder
Bong da	12/07/2021 4:47 PM	File folder
Chung khoan	12/07/2021 4:49 PM	File folder
Cum ga	12/07/2021 4:49 PM	File folder
Cuoc song do day	12/07/2021 4:49 PM	File folder
Du hoc	12/07/2021 4:49 PM	File folder
Du lich	12/07/2021 4:50 PM	File folder
Duong vao WTO	12/07/2021 4:50 PM	File folder
Gia dinh	12/07/2021 4:50 PM	File folder
Giai tri tin hoc	12/07/2021 4:50 PM	File folder
Giao duc	12/07/2021 4:51 PM	File folder
Gioi tinh	12/07/2021 4:51 PM	File folder
Hackers va Virus	12/07/2021 4:51 PM	File folder
Hinh su	12/07/2021 4:51 PM	File folder
Khong gian song	12/07/2021 4:51 PM	File folder
Kinh doanh quoc te	12/07/2021 4:52 PM	File folder
Lam dep	12/07/2021 4:52 PM	File folder
Loi song	12/07/2021 4:52 PM	File folder
Mua sam	12/07/2021 4:52 PM	File folder
My thuat	12/07/2021 4:52 PM	File folder
New folder	12/07/2021 6:41 PM	File folder
San khau dien anh	12/07/2021 4:53 PM	File folder
San pham tin hoc moi	12/07/2021 4:53 PM	File folder
Tennis	12/07/2021 4:53 PM	File folder
The gioi tre	12/07/2021 4:54 PM	File folder

Hình 10: Một ví dụ về các thư mục con trong thư mục *metadata*

Am nhac.idf	12/07/2021 6:23 PM	IDF File	59 KB
Am nhac.txt	12/07/2021 6:22 PM	Text Document	17 KB
Am thuc.idf	12/07/2021 6:23 PM	IDF File	20 KB
Am thuc.txt	12/07/2021 6:22 PM	Text Document	9 KB
Bat dong san.idf	12/07/2021 6:23 PM	IDF File	15 KB
Bat dong san.txt	12/07/2021 6:22 PM	Text Document	6 KB
Bong da.idf	12/07/2021 6:25 PM	IDF File	125 KB
Bong da.txt	12/07/2021 6:22 PM	Text Document	31 KB
Chung khoan.idf	12/07/2021 6:25 PM	IDF File	14 KB
Chung khoan.txt	12/07/2021 6:22 PM	Text Document	7 KB
Cum ga.idf	12/07/2021 6:26 PM	IDF File	23 KB
Cum ga.txt	12/07/2021 6:22 PM	Text Document	8 KB
Cuoc song do day.idf	12/07/2021 6:26 PM	IDF File	44 KB
Cuoc song do day.txt	12/07/2021 6:22 PM	Text Document	9 KB
Du hoc.idf	12/07/2021 6:26 PM	IDF File	34 KB
Du hoc.txt	12/07/2021 6:22 PM	Text Document	8 KB
Du lich.idf	12/07/2021 6:26 PM	IDF File	45 KB
Du lich.txt	12/07/2021 6:22 PM	Text Document	12 KB
Duong vao WTO.idf	12/07/2021 6:26 PM	IDF File	16 KB
Duong vao WTO.txt	12/07/2021 6:22 PM	Text Document	4 KB
Gia dinh.idf	12/07/2021 6:27 PM	IDF File	20 KB
Gia dinh.txt	12/07/2021 6:22 PM	Text Document	6 KB
Giai tri tin hoc.idf	12/07/2021 6:27 PM	IDF File	64 KB
Giai tri tin hoc.txt	12/07/2021 6:22 PM	Text Document	16 KB
Giao duc.idf	12/07/2021 6:27 PM	IDF File	32 KB
Giao duc.txt	12/07/2021 6:22 PM	Text Document	15 KB
Gini tinh.idf	12/07/2021 6:27 PM	IDF File	26 KB

Hình 11: Một ví dụ về các tập tin văn bản và tập tin IDF trong thư mục *metadata*

- Trong từng thư mục con của thư mục *metadata* sẽ chứa các tập tin TF tương ứng với các tập tin văn bản trong dataset folder.

AN_NLD_T_ (759).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (761).txt.tf	12/07/2021 6:22 PM	TF File	1 KB
AN_TN_T_ (762).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (763).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (764).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (765).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (766).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (767).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (768).txt.tf	12/07/2021 6:22 PM	TF File	1 KB
AN_TN_T_ (769).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (770).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (771).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (772).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (773).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (774).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (775).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (776).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (777).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (778).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (779).txt.tf	12/07/2021 6:22 PM	TF File	1 KB
AN_TN_T_ (780).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (781).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (782).txt.tf	12/07/2021 6:22 PM	TF File	3 KB
AN_TN_T_ (783).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (784).txt.tf	12/07/2021 6:22 PM	TF File	2 KB
AN_TN_T_ (785).txt.tf	12/07/2021 6:22 PM	TF File	4 KB
AN_TN_T_ (786).txt.tf	12/07/2021 6:22 PM	TF File	3 KB

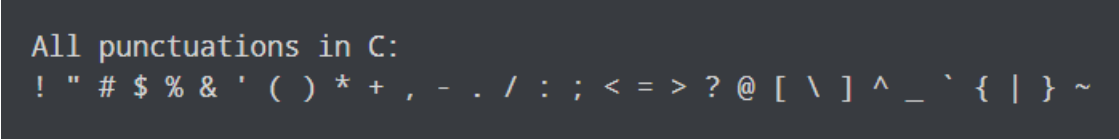
Hình 12: Một ví dụ về thư mục con của *metadata*

Lưu ý: Trong quá trình xử lý việc tạo/chỉnh sửa các tập tin siêu dữ liệu, chương trình sẽ tạo ra các tập tin và các thư mục như trên. Ta **KHÔNG** nên xóa hay có bất kì thao tác chỉnh sửa nào đối với các tập tin và thư mục này vì có thể dẫn đến việc chương trình cho ra kết quả sai hoặc gây ra lỗi trong việc vận hành của chương trình.

4 Xử lý dữ liệu tìm kiếm đầu vào

Chương trình chạy trên console nên chỉ hỗ trợ tìm kiếm các từ không dấu. Chức năng tìm kiếm với các từ tiếng Việt có dấu chưa được hỗ trợ.

Sau khi người dùng nhập dữ liệu đầu vào, các từ sẽ được tách ra thành các tokens bằng `stringstream` trong thư viện `sstream`. Sau đó, các tokens này được bỏ bớt các "punctuations" bằng hàm `ispunct` của thư viện `cctype`. Danh sách các punctuations được liệt kê trong hình 13 sau. [3]



```
All punctuations in C:
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

Hình 13: Các "punctuations" trong C++

Các tokens này được hàm `void searchSentence(const string& Sentence)` trong `FileProgression.h` xử lý, duyệt qua các `IDF_list` của các thư mục, tìm kiếm xem có token đó hay không, nếu có thì ta tìm tiếp trong các `TF_list` của các tập tin văn bản trong thư mục đó, nếu không thì ta bỏ qua thư mục đó, cứ như vậy đến hết các thư mục.

Do việc tìm kiếm tuần tự có thể tốn thời gian rất lâu, dựa vào việc khi tạo các tập tin siêu dữ liệu thì các từ đã được sắp xếp tăng dần, các thao tác tìm kiếm trong chương trình đều sử dụng thuật toán tìm kiếm nhị phân để có kết quả nhanh chóng.

Sau khi tìm được tất cả các tập tin chứa ít nhất một token, chương trình sẽ trả ra kết quả là tất cả các tập tin đó, với độ ưu tiên từ cao đến thấp, đi kèm là các tùy chọn cho người dùng (sẽ được nhắc đến trong phần sau).

Mức độ ưu tiên của các tập tin được tính như sau:

- Tập tin nào có số tokens trùng khớp nhiều hơn được ưu tiên trước.
- Nếu 2 tập tin có số tokens trùng khớp như nhau, tập tin có giá trị $tfidf(t, d, D)$ cao hơn được ưu tiên trước.

Các cấu trúc mà chương trình sử dụng trong quá trình này:

- Cấu trúc `FileData` dùng để quản lý các tập tin, gồm có 4 trường: `posFolder`, `posFile`, `value` và `intersectionCount`.

```
struct FileData {
    int posFolder;
    int posFile;
    double value;
    int intersectionCount;
};
```

- Trường `posFolder` có kiểu `int`, trường `posFile` có kiểu `int` tương ứng với vị trí của thư mục con trong danh sách thư mục và vị trí của tập tin trong danh sách tập tin của thư mục con đó. Danh sách thư mục được quản lý bởi một mảng động `string* folder_list`, phần tử thứ `i` của `folder_list` là thư mục thứ `i` trong danh sách thư mục con; danh sách tập tin trong thư mục con được quản lý bởi một mảng động hai chiều `string** file_list`, phần tử `file_list[i][j]` là tập tin thứ `j` của thư mục con thứ `i` trong danh sách,

- Trường `value` có kiểu `double` là giá trị $tfidf(t, d, D)$ của token đang tìm trong tập tin đó,
- Trường `intersectionCount` có kiểu `int` là số lượng tokens mà ta đang tìm trong tập tin.

- Cấu trúc `FolderData` dùng để quản lí các thư mục, gồm có 2 trường: `idfL` và `tfLArr`.

```
struct FolderData {  
    IDF_list idfL;  
    TF_list* tfLArr;  
};
```

- Trường `idfL` là `IDF_list` tương ứng với thư mục,
- Trường `tfLArr` là một mảng động các `TF_list` tương ứng với thư mục này.

- Cấu trúc `ResponseData` là dữ liệu về kết quả phản hồi cho người dùng, gồm 3 trường: `file`, `size` và `cap`.

```
struct ResponseData {  
    FileData* file;  
    int size;  
    int cap;  
};
```

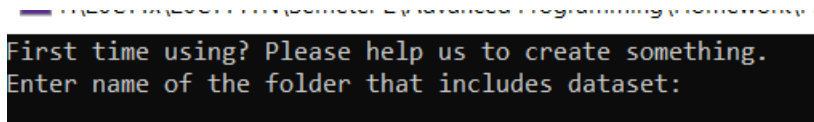
- Trường `file` là một mảng động các `fileData`, chứa danh sách các tập tin có chứa các tokens đang tìm,
- Trường `size` là số lượng các phần tử của `file`,
- Trường `cap` là số lượng tối đa các phần tử mà `file` có thể chứa, được khởi tạo và thay đổi cho phù hợp.

- Ngoài ra còn sử dụng một vài cấu trúc phụ khác.

5 Quá trình hoạt động của chương trình

5.1 Lần chạy đầu tiên

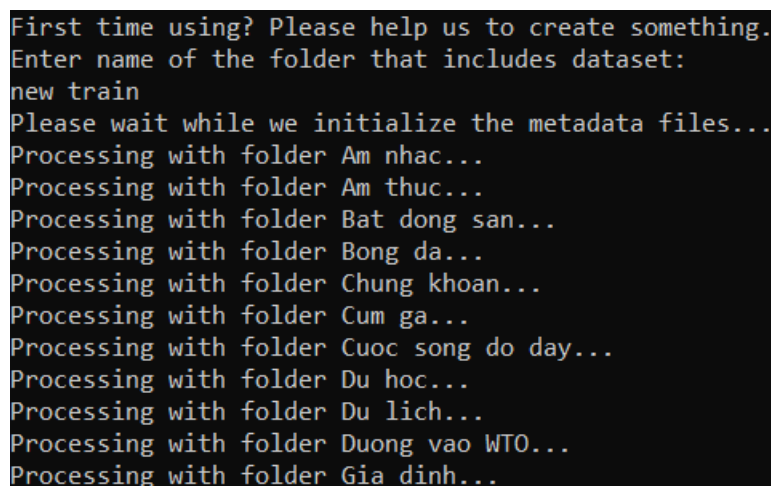
Khi khởi động, chương trình sẽ kiểm tra xem đây có phải là lần chạy đầu tiên hay không (do lần đầu tiên chưa có các tập tin siêu dữ liệu). Nếu có, chương trình sẽ yêu cầu người dùng nhập tên của dataset folder như hình dưới. Ở đây, dataset folder được nhập vào có tên là *new train*.



```
First time using? Please help us to create something.  
Enter name of the folder that includes dataset:
```

Hình 14: Nhập tên của dataset folder ở lần chạy đầu tiên

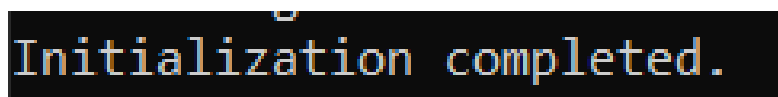
Sau khi nhập tên của dataset folder xong, chương trình sẽ tiến hành tạo các tập tin siêu dữ liệu theo quy trình bên trên. Quá trình này có thể tốn một vài phút, tùy vào số lượng tập tin văn bản trong thư mục dataset.



```
First time using? Please help us to create something.  
Enter name of the folder that includes dataset:  
new train  
Please wait while we initialize the metadata files...  
Processing with folder Am nhac...  
Processing with folder Am thuc...  
Processing with folder Bat dong san...  
Processing with folder Bong da...  
Processing with folder Chung khoan...  
Processing with folder Cum ga...  
Processing with folder Cuoc song do day...  
Processing with folder Du hoc...  
Processing with folder Du lich...  
Processing with folder Duong vao WTO...  
Processing with folder Gia dinh...
```

Hình 15: Chương trình đang tạo các tập tin siêu dữ liệu cho lần chạy đầu tiên

Lưu ý: các tập tin văn bản trong thư mục dataset cần phải ở dạng UTF-16, nếu không chương trình sẽ bị những lỗi không mong muốn.



```
Initialization completed.
```

Hình 16: Thông báo tạo thành công các tập tin siêu dữ liệu

5.2 Load dữ liệu vào RAM

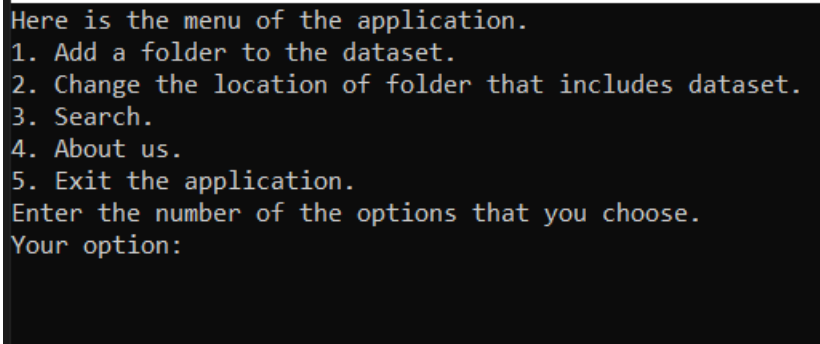
Sau khi tạo xong các tập tin siêu dữ liệu ở lần gọi đầu tiên, hoặc khi ta khởi động chương trình các lần tiếp theo, chương trình sẽ tiến hành load dữ liệu từ database về các tập tin siêu dữ liệu vào RAM.



Hình 17: Chương trình đang tiến hành load dữ liệu từ database

5.3 Menu của chương trình

Sau khi dữ liệu được đưa vào RAM, chương trình sẽ hiển thị một menu bao gồm các tính năng mà chương trình hỗ trợ.



Hình 18: Menu của chương trình

Ta có thể thấy các tính năng được liệt kê của chương trình bao gồm:

- Thêm một thư mục vào dataset folder,
- Đổi sang một dataset folder khác,
- Tìm kiếm
- Thông tin về người phát triển của chương trình,
- Thoát khỏi chương trình

5.3.1 Thêm một thư mục vào dataset folder

Khi ta chọn tùy chọn đầu tiên này, chương trình sẽ yêu cầu ta nhập đường dẫn đến thư mục cần thêm. Đường dẫn được nhập ở đây phải là **đường dẫn tuyệt đối** hoặc **tương đối với thư mục chứa chương trình**, ví dụ *F:\Folder Needs Adding*. Các tập tin trong thư mục này cũng phải có định dạng **UTF-16**, nếu không sẽ dẫn đến những lỗi không mong muốn cho chương trình.

Sau đó, thư mục này sẽ được thêm vào dataset folder, chương trình sẽ tạo các tập tin siêu dữ liệu tương ứng với thư mục này tương tự như với các thư mục khác. Tên của thư mục này được đưa vào **cuối** tập tin *FolderList.txt*. Tiếp theo đó, chương trình sẽ tự động cập nhật database rồi load vào RAM.

Sau khi thêm thư mục thành công, chương trình sẽ báo như hình 19. Ta có thể vào thư mục *metadata* để kiểm tra thành quả.

Lưu ý: chương trình chưa hỗ trợ việc thêm trực tiếp một tập tin văn bản vào

dataset folder, do đó, ta cần phải đóng gói các tập tin văn bản vào một thư mục trước khi thêm.

```
Your option: 1
Enter path to the folder you want to add: F:\Folder Needs Adding
Reloading database!, please be patient
Press any key to continue . . .
```

Hình 19: Ví dụ thêm thư mục vào dataset folder

5.3.2 Đổi sang một dataset folder khác

Đối với tùy chọn thứ 2 này, chương trình sẽ yêu cầu ta nhập tên của thư mục dataset folder mới.

Thư mục dataset mới cần phải nằm cùng thư mục với chương trình.

Sau khi nhận được tên của dataset folder mới, chương trình sẽ tiến hành giải phóng các tập tin siêu dữ liệu cũ, rồi lặp lại quy trình tạo các tập tin siêu dữ liệu đối với thư mục mới. Thời gian tiêu tốn của quá trình này phụ thuộc vào kích thước của thư mục dataset folder.

Sau khi thành công, chương trình sẽ thông báo như hình 20.

```
Your option: 2
Enter name of the folder that includes dataset:
another dataset
Please wait while we initialize the metadata files...
Processing with folder Am thuc...
Initialization completed.
Press any key to continue . . .
```

Hình 20: Thay đổi dataset folder

5.3.3 Tìm kiếm

Đây là phần mấu chốt, là mục tiêu của đồ án này. Việc tìm kiếm chỉ có thể thực hiện sau khi các tập tin siêu dữ liệu được tạo và dữ liệu được load vào RAM. Khi ta chọn tùy chọn này, chương trình sẽ yêu cầu ta nhập từ khóa (keyword) để tìm kiếm như hình 21.

```
Your option: 3
Enter the keywords to search:
chung ket copa america
```

Hình 21: Nhập từ khóa tìm kiếm

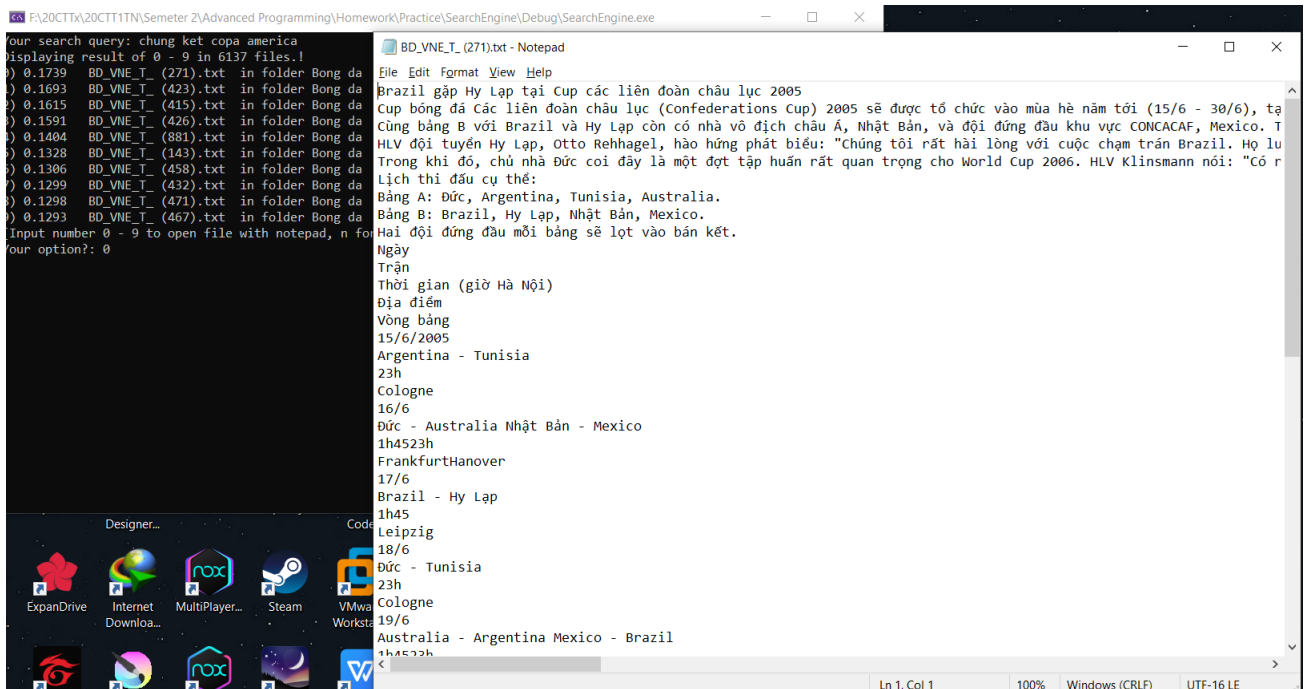
Sau đó, chương trình sẽ tiến hành xử lý và trả kết quả như hình 22 sau.

```
Your search query: chung ket copa america
Displaying result of 0 - 9 in 6137 files.
0) 0.1739 BD_VNE_T_ (271).txt in folder Bong da
1) 0.1693 BD_VNE_T_ (423).txt in folder Bong da
2) 0.1615 BD_VNE_T_ (415).txt in folder Bong da
3) 0.1591 BD_VNE_T_ (426).txt in folder Bong da
4) 0.1404 BD_VNE_T_ (881).txt in folder Bong da
5) 0.1328 BD_VNE_T_ (143).txt in folder Bong da
6) 0.1306 BD_VNE_T_ (458).txt in folder Bong da
7) 0.1299 BD_VNE_T_ (432).txt in folder Bong da
8) 0.1298 BD_VNE_T_ (471).txt in folder Bong da
9) 0.1293 BD_VNE_T_ (467).txt in folder Bong da
[Input number 0 - 9 to open file with notepad, n for next page, p for previous page and b to back to menu]
Your option?:
```

Hình 22: Kết quả tìm kiếm

Từ đó, ta có thể thấy các tính năng hỗ trợ người dùng trên giao diện kết quả như sau:

- Cho biết tổng số tập tin chứa ít nhất một **từ đơn** trong từ khóa cần tìm, cho biết ta đang xem các tập tin có thứ tự bao nhiêu.
- Cho biết thư mục chứa tập tin đó, cũng như giá trị $tfidf(t, d, D)$ của từ trong văn bản đó làm căn cứ để xếp hạng độ ưu tiên.
- Tính năng giúp mở tập tin bất kì trong danh sách kết quả với Notepad như hình 23.
Lưu ý: sau khi mở và xem một tập tin với Notepad, để chương trình có thể tiếp tục hoạt động, ta cần đóng tập tin đó lại.



Hình 23: Mở tập tin kết quả với Notepad

- Tính năng *next page* (hình 24) và *previous page* (hình 25) giúp ta đi đến trang trước hoặc trang sau của trang kết quả tìm kiếm hiện tại.

```
Your search query: chung ket copa america
Displaying result of 30 - 39 in 6137 files.!
0) 0.3527  SPTHM_VNE_T_ (123).txt      in folder San pham tin hoc moi
1) 0.3527  SPTHM_TT_T_ (541).txt      in folder San pham tin hoc moi
2) 0.3527  SPTHM_TT_T_ (487).txt      in folder San pham tin hoc moi
3) 0.3527  SPTHM_TT_T_ (506).txt      in folder San pham tin hoc moi
4) 0.3527  SPTHM_TT_T_ (518).txt      in folder San pham tin hoc moi
5) 0.3517  GTTH_TT_T_ (325).txt      in folder Giai tri tin hoc
6) 0.3471  DH_TN_T_ (252).txt      in folder Du hoc
7) 0.3415  DL_TT_T_ (387).txt      in folder Du lich
8) 0.3366  HS_VNE_T_ (113).txt      in folder Hinh su
9) 0.3311  MS_VNE_T_ (3).txt      in folder Mua sam
[Input number 0 - 9 to open file with notepad, n for next page, p for previous page and b to back to menu]
Your option?: _
```

Hình 24: Tính năng *next page*

```
Your search query: chung ket copa america
Displaying result of 20 - 29 in 6137 files.!
0) 0.1629  CSDD_TT_T_ (393).txt      in folder Cuoc song do day
1) 0.1484  BD_VNE_T_ (12).txt      in folder Bong da
2) 0.4863  SPTHM_TT_T_ (503).txt      in folder San pham tin hoc moi
3) 0.3880  GTTH_TT_T_ (410).txt      in folder Giai tri tin hoc
4) 0.3831  MS_VNE_T_ (18).txt      in folder Mua sam
5) 0.3809  SPTHM_TT_T_ (586).txt      in folder San pham tin hoc moi
6) 0.3703  SPTHM_VNE_T_ (295).txt      in folder San pham tin hoc moi
7) 0.3637  AT_VNE_T_ (24).txt      in folder Am thuc
8) 0.3554  AT_VNE_T_ (36).txt      in folder Am thuc
9) 0.3527  SPTHM_VNE_T_ (32).txt      in folder San pham tin hoc moi
[Input number 0 - 9 to open file with notepad, n for next page, p for previous page and b to back to menu]
Your option?:
```

Hình 25: Tính năng *previous page*

- Tính năng *back* giúp ta quay về menu chính của chương trình.

5.3.4 Thông tin về người phát triển của chương trình

Khi ta chọn tùy chọn này, màn hình sẽ hiển thị thông tin về các tác giả của chương trình như hình 26.

```
Your option: 4
PROJECT SEARCH ENGINE
Member 1.
Full name: Nguyen Van Loc
Date of birth: 18/08/2002
Student ID: 20120131
Member 2.
Full name: Vo Trong Nghia
Date of birth: 04/04/2002
Student ID: 20120536
Press any key to continue . . .
```

Hình 26: About us

5.3.5 Thoát khỏi chương trình

Đây là tùy chọn giúp ta thoát khỏi chương trình.

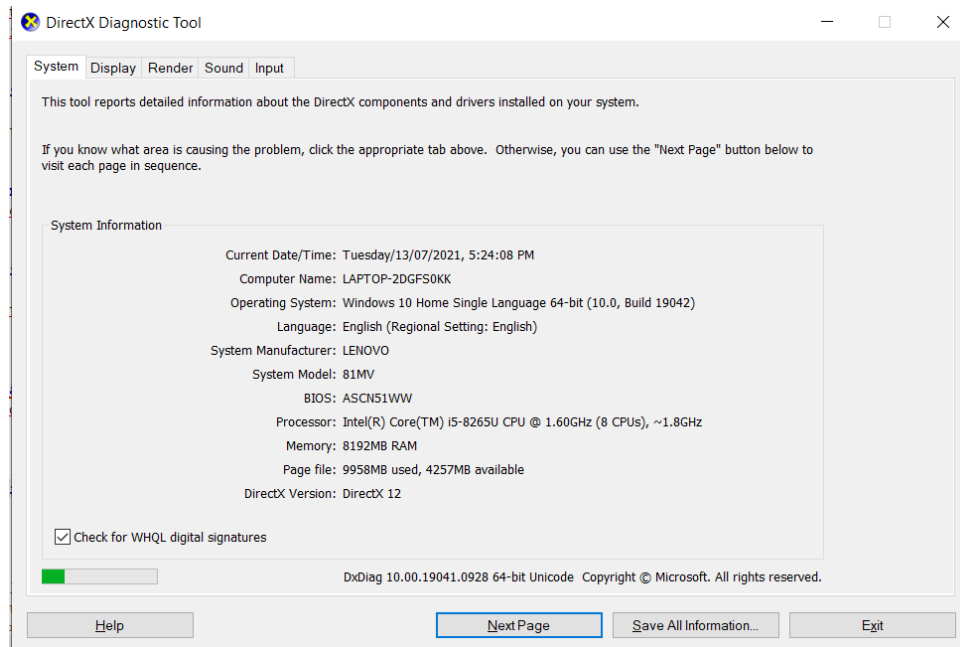
```
Your option: 5
Thank you very much! See you again!
F:\20CTTx\20CTT1TN\Semeter 2\Advanced Programming\Homework\Practice\SearchEngine\Debug\SearchEngine.exe (process 15020)
exited with code 0.
Press any key to close this window . . .
```

Hình 27: Thoát khỏi chương trình

6 Tài nguyên chương trình tiêu tốn

6.1 Cấu hình máy dùng để chạy thử chương trình

Chương trình được chạy thử trên máy có cấu hình như trong hình 28.

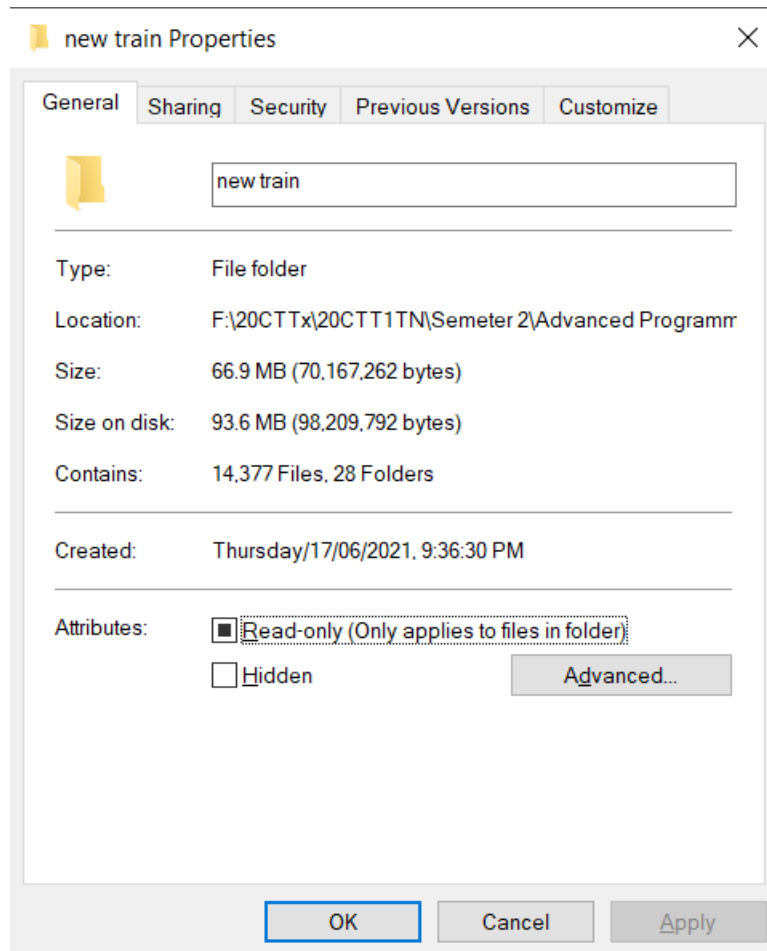


Hình 28: Cấu hình máy chạy thử chương trình.

6.2 Thời gian chạy thử chương trình

Ở đây ta sẽ chú trọng vào thời gian xử lý việc tạo tập tin siêu dữ liệu và thời gian xử lý việc load database vào RAM.

Thư mục dataset folder có 14377 tập tin văn bản với tổng kích thước là khoảng 66.9 MB như hình 29.



Hình 29: Kích thước dataset folder

Với tập dữ liệu trên, các thông số thời gian đo được là:

- Thời gian tạo các tập tin siêu dữ liệu là khoảng 647.29 giây.

Creating metadata files takes 663.645 seconds

Hình 30: Thời gian tạo các tập tin siêu dữ liệu

- Thời gian load database lên RAM khoảng 12.704 giây.

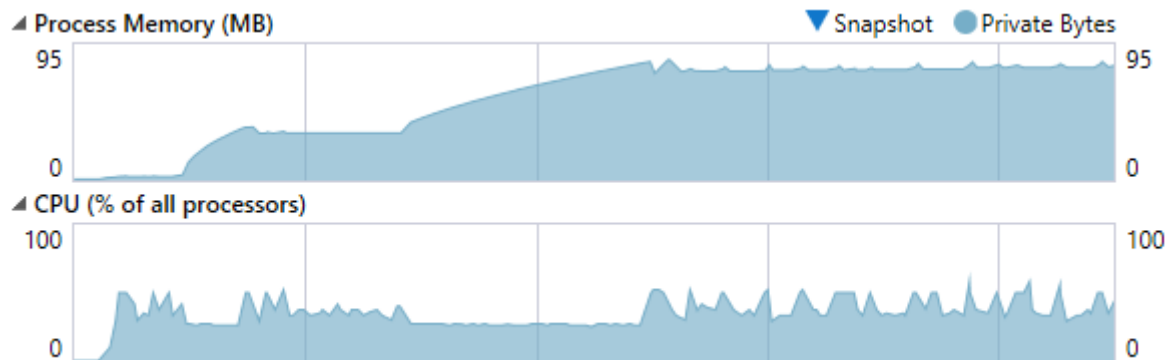
Loading to RAM takes 20.595 seconds

Hình 31: Thời gian load dữ liệu vào RAM

Thời gian tìm kiếm các từ khóa phụ thuộc vào độ dài và số lượng **từ đơn** của từ khóa, dao động khoảng 1 đến 2 giây.

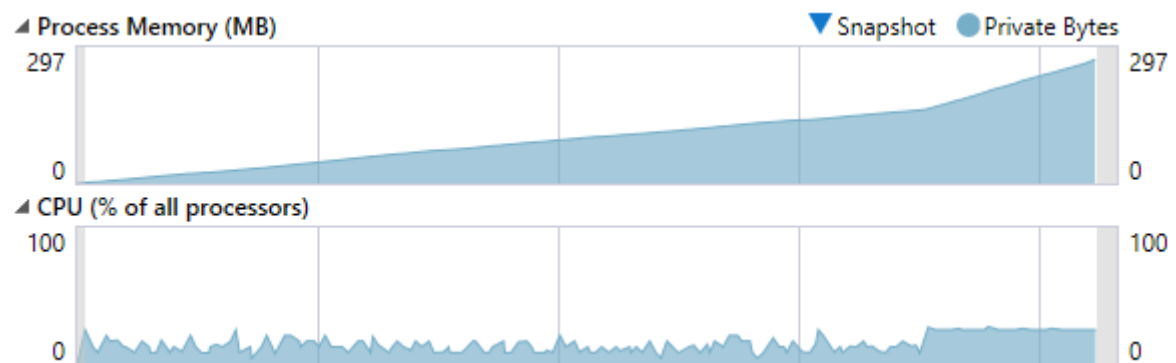
6.3 Bộ nhớ và CPU tiêu tốn

- Trong quá trình tạo các tập tin siêu dữ liệu:



Hình 32: Tài nguyên máy tính sử dụng khi tạo tập tin siêu dữ liệu

- Trong quá trình load dữ liệu từ database vào RAM:



Hình 33: Tài nguyên máy tính sử dụng khi load dữ liệu vào RAM

7 Nhận xét về chương trình và dự án

7.1 Về thời gian chạy của từng quá trình

- Hầu hết quá trình tạo/cập nhật các tập tin siêu dữ liệu là sắp xếp trên chuỗi bằng thuật toán merge sort. Merge sort đã được nhóm cải tiến sử dụng đa luồng, giúp sử dụng CPU hiệu quả hơn và giảm được gần một nửa thời gian chạy so với bình thường.
- Trong quá trình load dữ liệu từ database vào RAM, chương trình sử dụng một khoảng thời gian bình thường, do không có các quá trình nào tiêu tốn quá nhiều thời gian.
- Trong quá trình tìm kiếm và trả kết quả cho người dùng, do dữ liệu trong các tập tin TF và IDF đã được sắp xếp tăng dần nên thuật toán tìm kiếm nhị phân giúp xác định nhanh chóng *tất cả* các kết quả cần tìm.

7.2 Về bộ nhớ

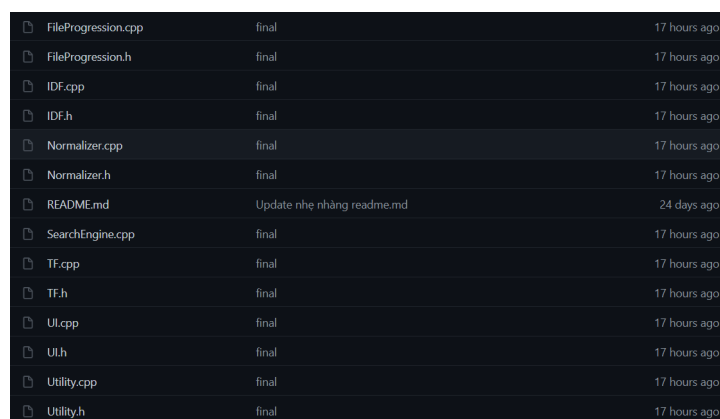
- Lớp `std::string` trong thư viện chuẩn của C++ được sử dụng xuyên suốt cả chương trình, đảm bảo an toàn về mặt bộ nhớ và dữ liệu được lưu trong heap.
- Một vài mảng động các `string` được vận dụng khi cần thiết, giảm thiểu việc allocate và de-allocate liên tục, gây ảnh hưởng đến performance của chương trình.
- Mỗi hàm sau khi kết thúc sẽ thực hiện trả vùng nhớ, đảm bảo tối ưu vùng nhớ heap.
- Không gây ra tình trạng memory leak trong chương trình.

7.3 Về cấu trúc chương trình

- Các files được chia hợp lý, thể hiện được các mục đích nhất định và không quá phụ thuộc vào nhau.
- Chương trình khi chạy luôn đảm bảo tham số đầu vào đúng mục đích.

7.4 Về quản lý dự án

Dự án được cooperated bởi hai thành viên trên nền tảng Github, đảm bảo tiến độ hoàn thành và quản lý các hàm, hỗ trợ sửa lỗi về sau.



FileProgression.cpp	final	17 hours ago
FileProgression.h	final	17 hours ago
IDF.cpp	final	17 hours ago
IDF.h	final	17 hours ago
Normalizer.cpp	final	17 hours ago
Normalizer.h	final	17 hours ago
README.md	Update nhẹ nhàng readme.md	24 days ago
SearchEngine.cpp	final	17 hours ago
TF.cpp	final	17 hours ago
TF.h	final	17 hours ago
Ul.cpp	final	17 hours ago
Ul.h	final	17 hours ago
Utility.cpp	final	17 hours ago
Utility.h	final	17 hours ago

Hình 34: Quản lý dự án bằng Github

Tài liệu

- [1] <https://nguyenvanhieu.vn/tf-idf-la-gi/>
- [2] <https://vi.wikipedia.org/wiki/Tf%E2%80%93idf>
- [3] <https://www.programiz.com/c-programming/library-function/ctype.h/ispunct>
- [4] <https://www.geeksforgeeks.org/>
- [5] <https://stackoverflow.com/>
- [6] <https://www.cplusplus.com/>