**Members:**

**1651040 - Nguyễn Mạch Thanh Vy**

**1651056 - Lưu Đức Tuấn**

**1651057 - Thái Mỹ Hương**

**1651076 - Bùi Quốc Huy**

**GROUP 04: PROJECT REPORT**

**PROBLEMS AND SOLUTIONS**

*INPUT*

We spent an amount of time to deal with names of files as efficient as possible. Finally we had found out that a function of visual studio could do this. Users could name files as they liked to but as long as they were located in a folder named “data”, we would able to load them into our application.

*QUERY*

At first, we built up one tree to store titles and another for contents but when we built a new tree to stock stop words, our application’s system had alerted memory leaking. As a result, we had merged titles and contents to one TRIE. In order to solve for “intitle” query, our group used an array of integer to reserve *length* of the array oftitle. After that, we checked for the last element of the array whether it was smaller than the length of the titles*.* If it was, then it must be in the title, if it was not, then it must not be.

At the beginning, our group planned to treat arrays in each query but that would take too much time and waste recursive properties of functions. That leaded us to process arrays from outside then use the array as query parameter.

In the first place, in order to proceed query “exact”, we came up with a plan to use z function but that was time wasting to run through all of the arrays. To prevent this, our group had switched our arrangement to use TRIE.

Query “and” and “or” was two independent queries at start but then we combined them to one. When users typed a keyword, the application would print “and” at the top, if it was not belonged to “and” then it must be in “or” (insured not to reprint two queries). We added an array to check whether a result had been output to create a connection between them. Each time we access in each query (“and” or “or”), we would pass this array in.

Firstly, “minus” could only process one keyword and one eliminated word and later on we had a solution, our application now can let users search for words and eliminate more.

*OUTPUT*

While solving for printing a sentence containing a keyword, we faced a problem of reprinting sentences because our group was using a matrix of n rows and m columns (n was the number of paragraphs, m was the length of the paragraph). This leaded our application to memory leaking so we had reused z function coded in query “exact” to process printed sentences.

In order to output a sentence including a keyword, our group had added an outputting array of m elements to our data structure (m was the amount of read data). This had lowered the process of inputting the array each time running sub application.

*DATA STRUCTURE*

At first, our plan of the maximum elements in each array was 5000 but later on, the memory of C++ did not allow that high amount of elements. As a result, we had to reduce our limit to 3000.

**DATA STRUCTURE**

* Our group will use TRIE for the main data structure.
* The reason we choose TRIE: TRIE is often used to store large number of strings. This is the most efficient data structure. TRIE is created based on each character. Each node contains one character. Thus, the complexity of TRIE when we try to search a key word is O(n).
* A TRIE will contain all the word in all given file. We will use an integer array to store the lengths of title in order to use query INTITLE. We also store all the contents’ length in order to optimize the speed of the system.
* We will separate one by one word in file and insert it into TRIE. We also have a linked list in each of node to determine the position of the word in what file and where it is in the paragraph. Because we realize the only one unique is the last character of the word (the first character can begin with many other character), we just need to store position of the last character in the node storing the last character. Besides, we also have the variable h to count the length of the word. In order to get the occurrence of one word, in the node storing the last character we have the variable occur to count how many times this word appear in the paragraph.
* Linked list of files in one node will contain a linked list of position of word.

For example: We have word “able” appearing at the position 1, 10, 20 in the first paragraph. The node storing “e” will have a linked list of files (struct order\_file) with order = 1 and list of words (struct order\_word) with val = 1 -> 10 -> 20 (-> means link to). If the second paragraph have this word, we will have list of files with order = 1 - > 2. In this second file, we will have list of words with similar structure.

By using this we use memories to increase the speed.

* In each node of TRIE, we have 40 links to other nodes which means 26 letters and 10 numbers (We create more than we need).
* Filename array will contain all the filename we input. We solve query FILETYPE by using the array to store the extension.
* To print the sentences containing all the key words, we use an array of strings named “output” to store the output sentence.

\*\*\* NOTE: We found the solution can read through all the files in the folder without using fixed name. Of course, we can get their name and extension.

**DESCRIPTION**

* User enters a keyword and the program runs each time only a query, not at the same time.
* If the program sees that user enters “#”, it will understand the input as for a “hashtag query”. Hashtag will be searched as an “or query”, it would only need a “#” to satisfy the condition.
* Because we have not done the “price in range query”, the program only has one price query. If the program finds “$”, it would automatically understand it is looking at “price query”. Price will be searched as an “and query” because it would requires an object’s name plus its price compulsively in the content of the article. If there is price but no object’s name in the article then it would still be considered as unsatisfying.
* If the keyword does not belong to those special queries above, the program will check whether it has “-” or not and another condition has to be satisfied is that right after the character “-” has to be a word or number. Only then, the program will perform the “minus query”. In the process of splitting each word, the program has already examined words that have the character “-” before them, which will need to be eliminated in this query. “Minus query” will return results with all the words the in the keyword and leave out words that need to be eliminated.
* “Exact query” is the query that would be checked next. The keyword only needs character “"” to be classified in the “exact query”. In the making of the “exact query”, our group needs to process string of keywords one more time by grouping characters within the “"” to be a word and those outside the “"” would still be spit normally. “Exact query” will return result when the passage contains all the keywords, which means as the same time as “and query”.
* The program will search to file type when the keyword contains the word “filetype”. For example, filetype: txt inp int. We will use an extra sub-array to store file extension.
* “Intitle query” only searches keywords inside the tittle. The length of tittles is already known so as long as the keyword has the last position smaller than the tittle’s length then it belongs to the tittle. And it only needs one word in words of the keyword to return the result.
* If it does not belong to the special cases, the program will use “and query” and then “or query”.
* Finally the program will print out the number of results has been found.

**ALGORITHMS**

* In order to classify the types of queries, we use the available function of string is string find.
* When we input keywords, we will split the keywords into separated words. Firstly, we will add a dot and a space in the end of the keywords. We scan through each character of the keyword to split the keywords into individual words. In this case our group does not use the available find string function, since if we use it, the separating character between 2 words could be one or many punctuation marks for example. Thus in order to avoid this case, we try to scan each character to split all the keywords.
* Search algorithm: After splitting all the keywords, we scan through each character of all the keywords to access to TRIE to check if TRIE contain that word or not. This is the main method in our program to search a keyword.
* Z Function: the algorithm to find a substring in a string. This method is used to optimize the speed in finding substrings. We use this in store output sentences. Firstly, we will find the sentence contain the keyword, after that we will check it if it is a substring of the already output sentence or not. If not, we will add this string into an output result.

**EXTRA:**

* Eliminating all the stop words: we use TRIE to store a list of stop words in stopwords.txt file. If we see any input keywords in this list, we will skip it. However, in TRIE, we have already stored all the stop words. If user wants search for stop words, you just need add “+” before that words. A stop word file names “stopwords.txt”, each line in this file contains 1 stop word.

For example: “a”, “the” are stop words. If you want to search for them, you just need to input: +a +the.

* History of words: we try to store 10 most searched words. At first, we think of using queue to store the data. But then, we find out that we can’t print all the data inside the queue without deleting all of them. Finally, we come up with using vector so as to store the searched words. Each running time, we will output the list of searched words to user in order to see the old words.
* Reading file: in this project, we try to use the new method to read all files in a folder. If we have already learned the method to open from files but we need that file names. However, we found the method to read all the files without knowing their names. Visual Studio, our IDE, helped us to do this. This is our library and a small code about this.

Example:

#include<filesystem>

#include<experimental/filesystem>

using namespace std::experimental::filesystem; //Read all file

for (auto& p : directory\_iterator("data")) //Data files put in folder "data"

{

fi.open(p.path(), ios::in); //p.path() : file name

...

}

**OPTIMIZING ISSUES**

We learned more about Z function which is a finding substring in the string. Initially, our program ran very slowly to output all the results, because we checked a keyword be a substring in the parent string by using the algorithm that checks all the characters in that keyword to make sure the parent string contains all of them. Then we applied the Z function that uses dynamic programming to find a substring in one string. We added the keyword before the parent string. In addition, between the keyword and parent string, we also added a special character between them. Then we applied Z function to handle. Finally, by using this, our program runs faster.

**EXAMPLE**

Use group 4 data only (56 files)

// The running time is based on the processor of the computer

|  |  |  |
| --- | --- | --- |
| **Query** | **Input** | **Output** |
| Exact | "Boston University" | QUERY EXACT  Running time: 0.004  All the keywords inputted appeared in article:  Group04\_25.txt: He retired from teaching at Boston University in 2007 and spent more of his time in St.  Group04\_34.txt: The researchers, led by Matthew Pase of the Boston University School of Medicine and colleagues, studied more than 4,000 people for their report, published in the journal Stroke.  About 2 result(s)  ENTER TO CONTINUE … |
| Hashtag | #Pride30 | QUERY HASHTAG  Running time: 0.002  All the keywords inputted appeared in article:  Group04\_22.txt: Bruce Cohen was nominated for NBC Out's #Pride30 list by singer Melissa Etheridge, who said Cohen is an "amazing person" and credited him with inspiring her to come out publicly.  About 1 result(s)  ENTER TO CONTINUE … |
| Price | Uber $70 | QUERY PRICE  Running time: 0.012  All the keywords inputted appeared in article:  Group04\_42.txt: Not only are senior leaders going to be held accountable — both in performance review and pay — but Holder's report also calls on Uber to "eliminate those values which have been identified as redundant or as having been used to justify poor behavior.". Former attorney general Eric Holder's recommendations into how Uber can reform its workplace were released on Tuesday and shed light on what it's like to work at a company that went from a few core team members in 2009, to the $70 billion giant it is today..  About 1 result(s)  ENTER TO CONTINUE … |
| Filetype | filetype txt | QUERY FILETYPE  Running time: 0  All the keywords inputted appeared in article:  Group04\_01.txt  Group04\_02.txt  Group04\_03.txt  Group04\_04.txt  Group04\_05.txt  Group04\_06.txt  Group04\_07.txt  Group04\_08.txt  Group04\_09.txt  Group04\_10.txt  Group04\_11.txt  Group04\_12.txt  Group04\_13.txt  Group04\_14.txt  Group04\_15.txt  Group04\_16.txt  Group04\_17.txt  Group04\_18.txt  Group04\_19.txt  Group04\_20.txt  Group04\_21.txt  Group04\_22.txt  Group04\_23.txt  Group04\_24.txt  Group04\_25.txt  Group04\_26.txt  Group04\_27.txt  Group04\_28.txt  Group04\_29.txt  Group04\_30.txt  Group04\_31.txt  Group04\_32.txt  Group04\_33.txt  Group04\_34.txt  Group04\_35.txt  Group04\_36.txt  Group04\_37.txt  Group04\_38.txt  Group04\_39.txt  Group04\_40.txt  Group04\_41.txt  Group04\_42.txt  Group04\_43.txt  Group04\_44.txt  Group04\_45.txt  Group04\_46.txt  Group04\_47.txt  Group04\_48.txt  Group04\_49.txt  Group04\_50.txt  Group04\_51.txt  Group04\_52.txt  Group04\_53.txt  Group04\_54.txt  Group04\_55.txt  Group04\_56.txt  About 56 result(s)  ENTER TO CONTINUE … |
| Intitle | Intitle Obama Cuba Policy | QUERY INTITLE  Running time: 0.007  All the keywords inputted appeared in article:  Group04\_03.txt: Trump Will Tighten, but not Nix, Cuba Travel .  Group04\_05.txt: Trump Seeks Limited Changes to Obama Cuba Policy.  Group04\_54.txt: After Ferguson: Is ‘Hashtag Activism’ Spurring Policy Changes?.  About 3 result(s)  ENTER TO CONTINUE … |
| Minus | apple -pay | QUERY MINUS  Running time: 0.012  The keywords inputted appeared in article:  Group04\_27.txt: “I had the same lunch every day and it was a protein shake with an apple and peanut butter and it’s like, ‘Oh that’s totally a healthy lunch — this is a great lunch.' But when I weighed peanut butter for the first time I was taking like three servings and I thought it was only one,” Easter told NBC News Better..  Group04\_39.txt: She has worked at Apple for nearly 14 years and is their Senior Manager of accessibility policy and initiatives..  Group04\_40.txt: If your car has Apple's connected car platform, CarPlay, or a standard Bluetooth system, you should get an alert when you exit your car, plotting in Apple Maps where you parked.  About 3 result(s)  ENTER TO CONTINUE … |
| And + or (include stopword) | fans +whom | QUERY AND + OR  Running time: 0.034  Group04\_23.txt: The show's themes of family, friendship, and individuality have also resonated with fans, many of whom express their appreciation with cast members via social media.  Group04\_37.txt: The researchers saw similar results for people age 75 and older, for whom the ACC/AHA guidelines generally recommend a slightly lower dose of statins..  Group04\_50.txt: Manufacturers and fans of these products claim they are as safe as caffeine, but there is little evidence to support that claim..  Group04\_51.txt: When "Game of Thrones" fans, upset over the "Red Wedding," took to Twitter using the #GoT hashtag, they averaged 6,000 tweets per minute, according to Trendrr.tv.  Group04\_53.txt: He said there is no intention to prevent fans and customers from using the hashtags on sites like Facebook, Twitter and Instagram..  About 5 result(s)  ENTER TO CONTINUE … |

**CONCLUSION**

After weeks of trying and failing, we have finished our project successfully. During that time, we tried ways to build up this project, including Z function and TRIE tree. Some of us have not met these materials yet but they catch up fast and in time for our coding process.

Changing our plans made our team suffered a hard time to rebuild our work. But we also realized that keeping our old source code was quite important. We have reused our Z function source code to improve our running time. We also learnt that we needed to review our own source code that had been coded for months. Because we would never know they could be useful in the future and save us times to recode them.

We also knew the way to manage multi-functional application. It was hard to understand source code of other teammates and make sure that our code did not need more variables to pass them in. If we did not, our project would need to build more variables.

Furthermore, the most important thing after this project is using as much resource which we already had, as possible (mentioned in the description). That would save us a lot of time compare to code a new function to process each query.

In conclusion, this project provides us new knowledge, new ways to build up an application. It also sums up all reparation for the final exam.