Algorithms Design
Homework Assignment

June, 2021

Student: Golea Cristian

Group: 1.2 A, First Year

Contents

1 Homework assignment

1.1 Problem statement

You must develop a software package for library management. The library has a set of books. Each book has a unique title and a set of one or more authors. An author can co-author one or more books. An author has a unique pair containing his or her given name and family name. It is important to design an efficient representation of the set of books of the library, avoiding redundant information. You choose to represent sets of books and sets of authors using singly linked lists, such that each author and each book are uniquely represented in the library management system. You must define algorithms for the following operations:
i) adding a book to the library;
ii) determining the list of books co-authored by a given author;
iii) determining the list of authors that co-authored a given book. At this step you must: describe the data structures of the library system, as well as the approach for the library operations and design algorithms in pseudo-code for the library operations. For each algorithm you must indicate and justify its computational complexity. Your implementation will minimally provide the following operations: to construct a book; to add a book to the library system; to determine the authors that co-authored a given book; to determine the books co-authored by a given author.

1.2 Algorithms

Declare node and pointers(book, autor)

```
1. struct book
2.     char *titlu
3.     int nr_authors
4.     int *author_list
5.     struct book *next
6. struct author
7.     char *nume
8.     char *prenume
9.     int nr_books
10.    int *books_list
11.    struct author *next
```

Create two empty linked list (one for books and one for authors)

```
1. head = NULL
2. head1 = NULL;
```

Add a book to the list of books(head, head1)

```
1. We allocate memory for a new book
2. read titlu
3. Insert the new title in the book node
```

```
4. read $numberofauthors$
5. Insert the number of authors in the book node
6. We allocate memory for the $author\_list$ for each book
7. for $j$=1; j ¡= $nr\_ath$; $j$++
8.   read $prenume1$ and $nume1$
9.   $ok$=0
10.  We declare a iterator for author_list: $*iterator = head1$
11.    while $iterator->next$ != NULL and $ok$!=1
12.    if we found the author in the list
13.      ok=1
14.    $iterator = iterator->next$
15.    if (ok==0)
16.      We add the new author to the list by calling the add_author function
17.      We add in the vector of books from this author the book he wrote
18.    else
19.      $iterator->nr\_books$++
20.      We add in the vector of books from this author the book he wrote
21. $new\_book->next = head->next$ // we link the book to the list.
22. $head->next = new\_book$ // we link the book to the list.
```

Add an author to the list of authors(head, nume1, prenume1)

```
1. We allocate memory for the new author
2. Insert the first and last name in the new author node
3. We allocate memory for the $book\_list$ for each author
4. $new\_author->nr\_books = 1$
5. $new\_author->next = head->next$ // we link the new author to the list
6. $head->next = new\_author$ // we link the new author to the list
```

Printing the list of books co-authored by a given author (head, head1)

```
1. We find the author in the list of authors
2. print $books\_list$
```

Printing the list of authors of a book(head, head1)

```
1. We find the book in the list of books
2. print $author\_list$
```

Time Complexity

Linked lists have most of their benefit when it comes to the insertion and deletion of nodes in the list. Unlike the dynamic array, insertion and deletion at any part of the list takes constant time. However, unlike dynamic arrays, accessing the data in these nodes takes linear time because of the need to search through the entire list via pointers. It's also important to note that there is no way of optimizing search in linked lists. In the array, we could at least keep the array sorted. However, since we don't know how long the linked list is, there is no way of performing a binary search: Indexing - O(n), Insertion - O(1), Search - O(n). Deletion - O(1);

Space Complexity
Linked lists hold two main pieces of information (the value and pointer) per node. This means that the amount of data stored increases linearly with the number of nodes in the list. Therefore, the space complexity of the linked list is linear: Space - O(n).

1.3 Experimental data

```
gen-random-string(*s, len)
1. alphabet[]="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
2. for i=1; i ¡= len; j++
3.     $s[i] = alphabet[rand()\%(sizeof(alphabet) - 1)]$
4.     s[len] = 0
```

```
gen-random-number(lower, upper)
1. num=(rand() % (upper - lower +1)) + lower
2. return num
```

```
data-generator(nr_of_books)
1.print("How many books do you want to generate? ")
2.read(nr_of_books)
3.allocate memory for s
4.for i=1 i <= nr_of_books i++
5.    gen_random_string(s,gen_random_num(2,10))
6.    nr = gen_random_num(2,5)
7.    print(s, nr)
8.    for j = 1 j <= nr j++)
9.        gen_random_string(s,gen_random_num(2,10))
10.         print (s )
11.         gen_random_string(s,gen_random_num(2,10))
12.         print (s )
13.    print newline
```

Description of generating algorithm

The generating data algorithm focus on generating books. You need to enter the number of books you want to generate. It generate a random title, for each title it generate a random number of authors and for each author a first and last name.

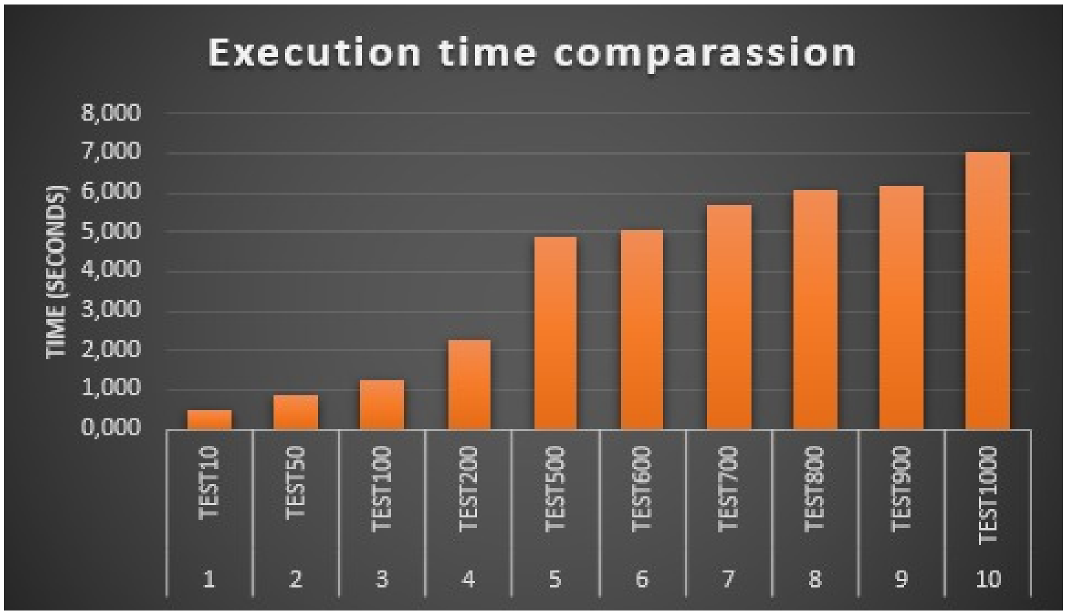
After running both algorithms we get a list of books and a list of authors. Now, we can see the books written by an given author and the authors of a book.

Modules for C Code:

- **main.c** - It is the main module and it interconnect all the other modules.
- **linked_list.c** - Here are all the function of the lists.
- **menu.c** - In this module is the menu were we can acces all the function from linked_lists.
- **linked_list.h** - It is a header type module. It is the header of linked_list.c.
- **menu.h** - It is a header type module. It is the header of menu.c .

1.4 Results & Conclusions

In this section it is presented a comparison of time execution on the C code. After running the algorithm it is generated the library of books. The execution time increase when the number of books increases.



Time Execution Graph

Conclusion and personal thoughts

After lots and lots of work and research I finally finished the project. The biggest achievement I've got is that I can write pseudocode, write code, test the code, and write a report about it. I don't do it perfectly, but I have a starting point. In this semester, I discovered how many things I have to learn in this huge domain.

The most challenging part was implementing the algorithms in Python, because I had no experience with this language before this project. In fact, I could not deliver a code in Python as good as the one in C (but it works). I could not learn Python as well as I know C or C++. For example, I could not find the structures from C in Python

Another challenging thing was the testing part, I had so many possible cases and different type of input and I got lost, but finally with a bit of help form the teachers and from other teammates I handled it.

From this project I learned to be more responsible and to put the work in and results will come. I got to use new technologies and to find new sources of information for learning new skills.

For the future, I want to get better using C, but firstly I would like to be better in writing code in general, to develop smart algorithms and after that translating them into a language or another.

I think that weekly assignments were a very good idea and help me a lot to acknowledge a lot of information.

1.5 References

References

[1] Algorithm Design Course held by Professor PhD. Eng. Costin Bădică

[2] Thomas H. Cormen and Charles E. Leiserson and Ronald L. Rivest and Clifford Stein, *Introduction to Algorithms*. MIT Press, 3rd Edition, 2009.

[3] LaTeX project site: http://latex-project.org/

[4] LaTeX documentation site: https://www.overleaf.com/learn

[5] YouTube : https://www.youtube.com/

[6] Quora : https://www.quora.com/

[7] Briliant : https://brilliant.org/

[8] GeekForGeeks : https://www.geeksforgeeks.org/

[9] IncludeHelp :https://www.includehelp.com/c-programming-questions/what-is-makefile.aspx

[10] StackOverflow: https://stackoverflow.com/