Divide And Conquer: Library Book Search

•••

Searching for a book in a physical library

By Benjamin Mai

Problem/Scenario

Information

Imagine that you want to search for a book throughout a library. There are many attributes associated with each book such as Title, Author, Genre, and Publisher. However, we can make this a much more simplified process by utilizing a 1D Binary Search for those searching based on genres, and a 2D Binary Search for searching based off book titles solely.

Algorithm Explanation:
Binary Search

Algorithm Used in Application - Binary Search

Step 1 → If first index is greater than the last, then return -1

Step 2 → Find the midpoint, which can be done by adding the first and last index, then dividing by 2

Step 3 → If mid is empty, then we can find the closest non-empty string

Step 4 → If left index is less than the first, and right index is greater than the last, then we return -1

Step 5 → If right index is less than or equal to the last and the length of this halve is not 0, then we can set mid by calculating it again

Step 6 → If the left index is greater than or equal to the first and the length of this part is not 0, we can calculate the mid again

Step 7 → If the string is found at the mid, then we return it

Step 8 → If the string is greater than the current mid, we can set the new lower bound and upper bound, then calculate the new mid

Step 9 → If the string is smaller than mid, we set a new upper bound and perhaps a lower bound, then calculate a new mid within this part

Step 10 → Repeat until search value is found!

Algorithm Used in Application - Binary2dSearch

Step 1 → Determine how to search the 2d matrix for a given book title

Step 2 → Set condition to end the loop if size of array is exceeded when iterating

Step 3 → Find the midpoint of matrix row

Step 4 → If the midpoint is found , return it

Step 5 → If the book trying to be found is in the first half relative to the midpoint, we would find the new midpoint in this section

Step 6 → If the book trying to be found is in the second half relative to the midpoint, we would find the new midpoint in this section

Step 7 → If the book isn't found at all, we return false (-1)

Time Complexity

Time Complexity of this algorithm:

O(log n)

Implementation of our Library Search System

How the Application Works

1. First off, our application reads a csv file within a shared file with the source code
2. Then, it creates a hashmap key that represents the genres, whereas the array of books are the associated values
3. We then categorize books based on genre
4. Next, sorting the book arrays in the file are done alphabetically
5. The searched book is compared to a string within the book array to look for it utilizing Binary Search as mentioned earlier

Usage: Searching Based on Option#1: By Genre

1. Users are prompted to input a Genre of their choice based on the listed ones on the user screen

2. Next, they will be asked to input a title within that genre

3. If it is found, the index along with other associated information such as Title, Author, Genre, and Publisher are displayed

4. If it isn’t found, the users will be notified that it is not in the system

Usage: Searching Based on Option#2 - By Books

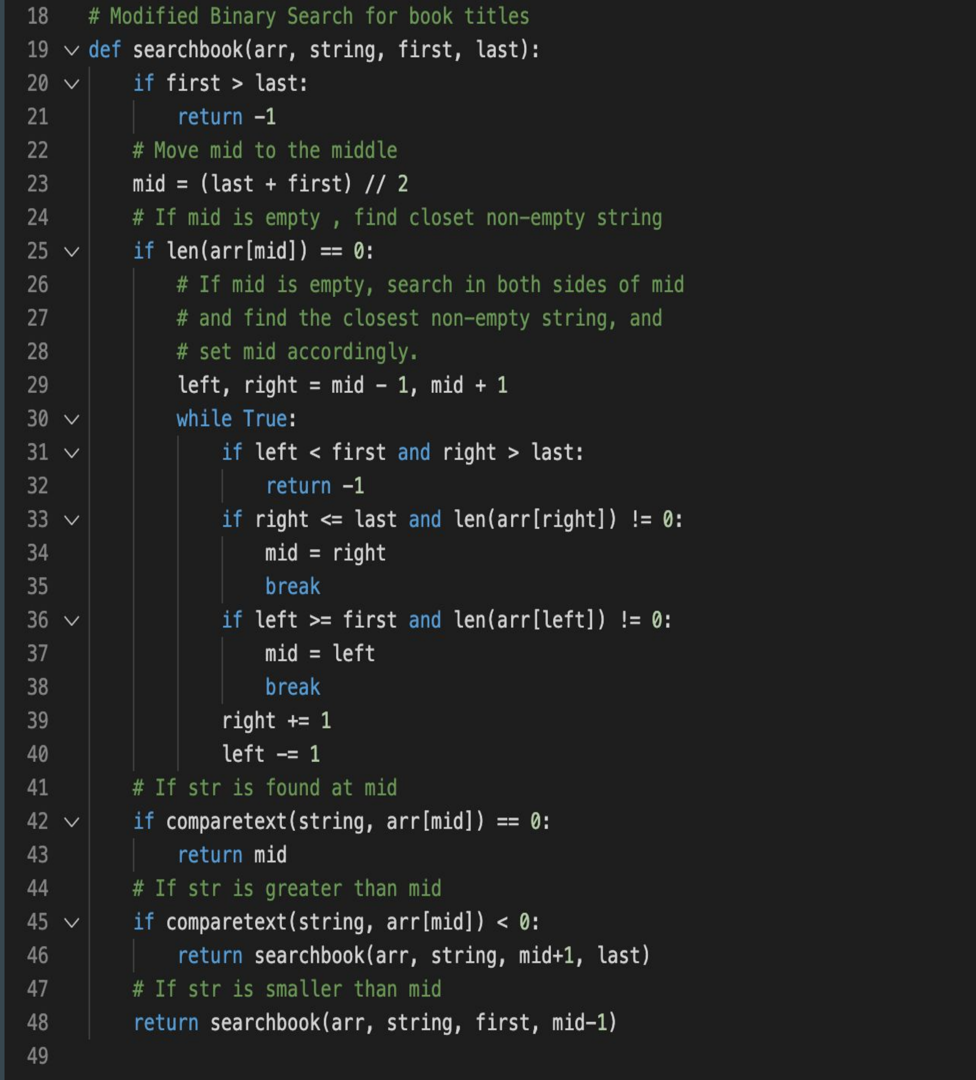
1. Users can be prompted to input a book title among the ones listed on the screen

2. These books are already categorized based on their genre, and they are displayed in terms of a matrix, so that we can utilize 2D Binary Search

3. Once the book is searched, it will return information of the one they were looking for such as Index, Title, Genre, Author, Publisher, and how many are in stock

Python Code

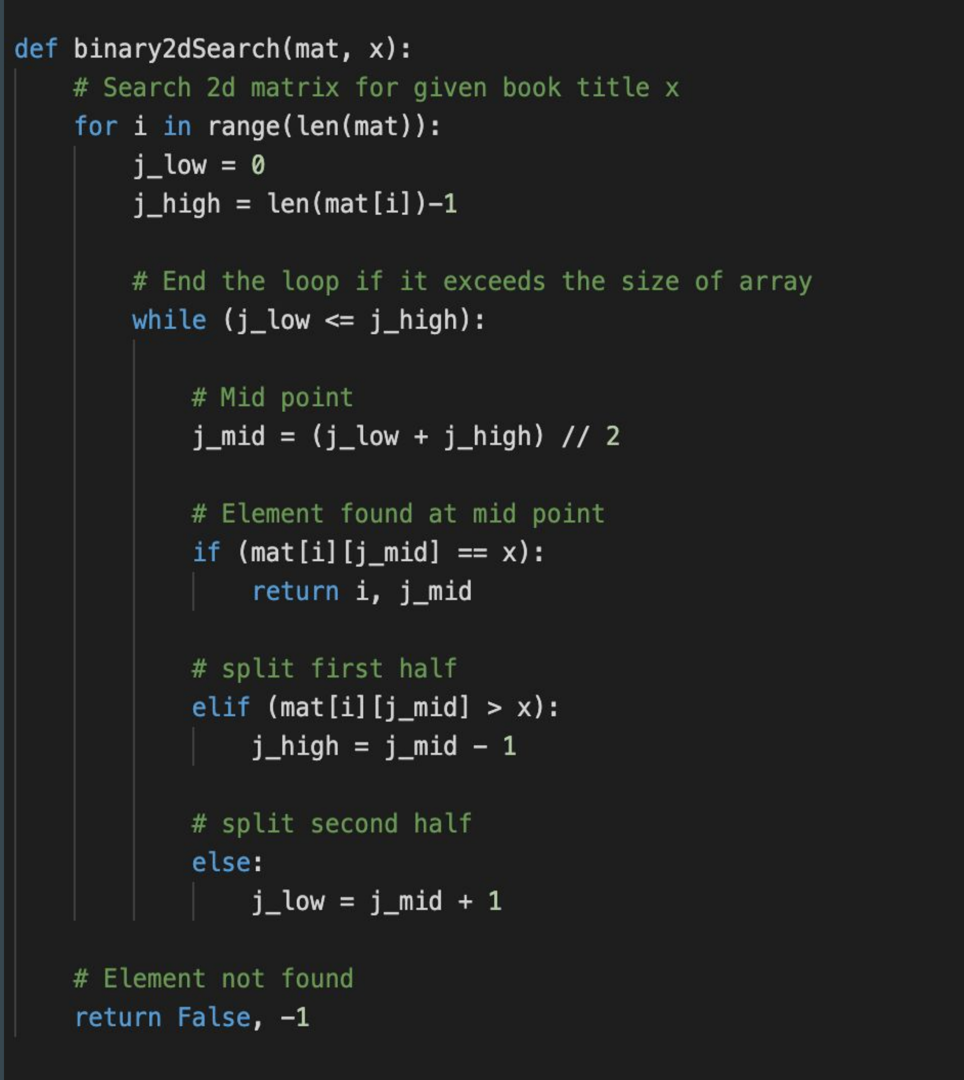
On the right is a more visual interpretation of Binary Search, which happens to utilize recursion. We left many comments to ensure you all understand what each chunk of code does. Basically, this is what helps us find books when considering genre first.



```python
# Modified Binary Search for book titles
def searchbook(arr, string, first, last):
    if first > last:
        return -1
    # Move mid to the middle
    mid = (last + first) // 2
    # If mid is empty , find closet non-empty string
    if len(arr[mid]) == 0:
        # If mid is empty, search in both sides of mid
        # and find the closest non-empty string, and
        # set mid accordingly.
        left, right = mid - 1, mid + 1
        while True:
            if left < first and right > last:
                return -1
            if right <= last and len(arr[right]) != 0:
                mid = right
                break
            if left >= first and len(arr[left]) != 0:
                mid = left
                break
            right += 1
            left -= 1
    # If str is found at mid
    if comparetext(string, arr[mid]) == 0:
        return mid
    # If str is greater than mid
    if comparetext(string, arr[mid]) < 0:
        return searchbook(arr, string, mid+1, last)
    # If str is smaller than mid
    return searchbook(arr, string, first, mid-1)

```

Python Code

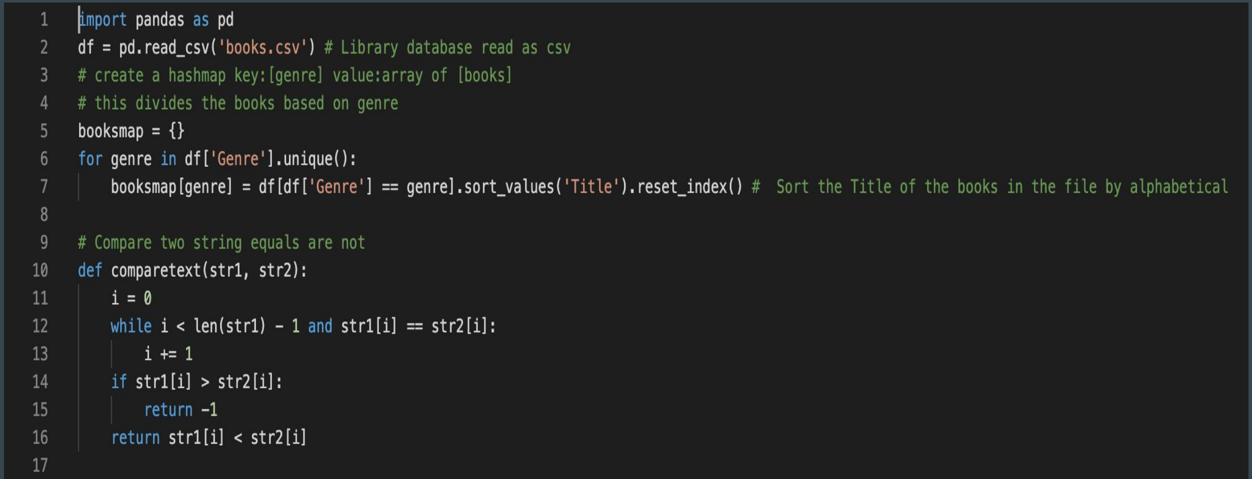
On the right is the code for the 2D Binary Search Algorithm that is used in our program to look for books when users only want to search based on book titles specifically. Together, this and the previous 1D Binary Search algorithm make this application utilize both algorithms for different purposes in our application, which we explained earlier.



```python
def binary2dSearch(mat, x):
    # Search 2d matrix for given book title x
    for i in range(len(mat)):
        j_low = 0
        j_high = len(mat[i])-1

        # End the loop if it exceeds the size of array
        while (j_low <= j_high):

            # Mid point
            j_mid = (j_low + j_high) // 2

            # Element found at mid point
            if (mat[i][j_mid] == x):
                return i, j_mid

            # split first half
            elif (mat[i][j_mid] > x):
                j_high = j_mid - 1

            # split second half
            else:
                j_low = j_mid + 1

    # Element not found
    return False, -1
```

Reading from File, Sorting, and Comparing Strings

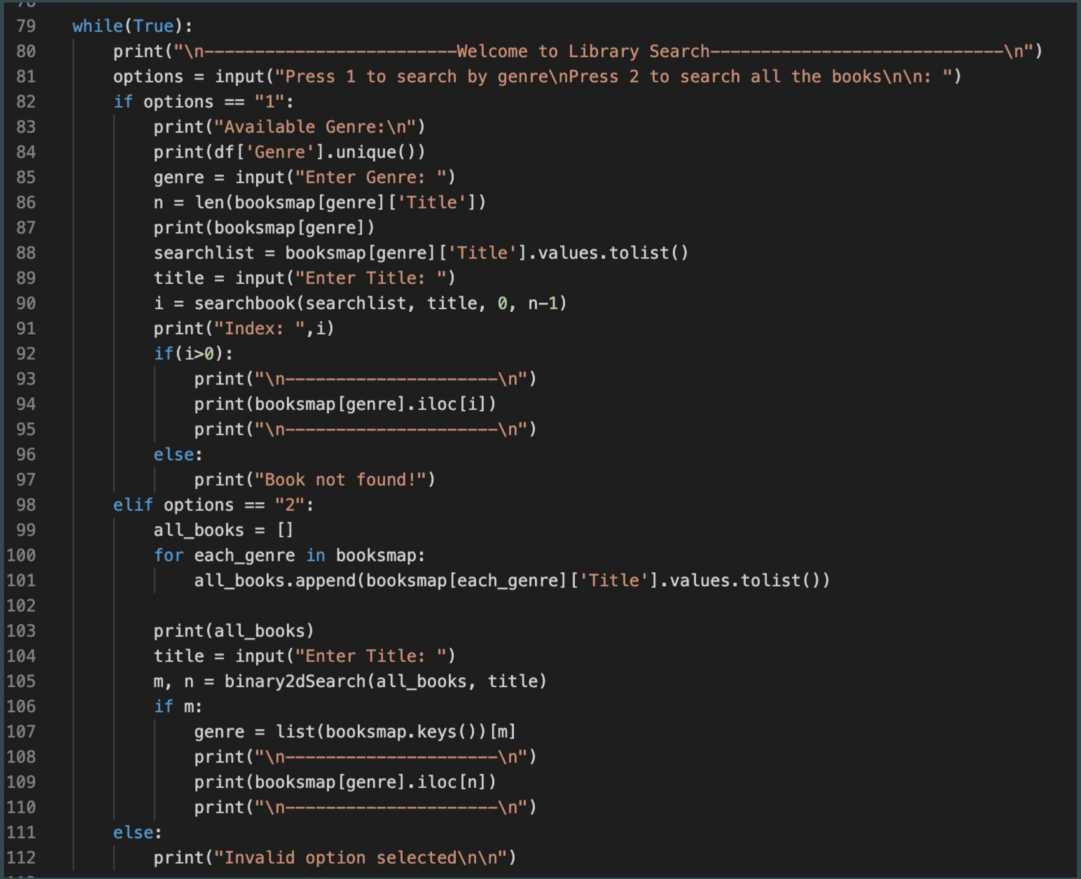
As shown below, we can see how we utilized the Python pandas library, which is used for data analysis and manipulation. Also, we can see how the genres represent the hashmap keys, whereas the books are represented as an array of books. Moving on, we have to compare the user inputted strings to what is actually being contained in the array.



```python
import pandas as pd
df = pd.read_csv('books.csv') # Library database read as csv
# create a hashmap key:[genre] value:array of [books]
# this divides the books based on genre
booksmap = {}
for genre in df['Genre'].unique():
    booksmap[genre] = df[df['Genre'] == genre].sort_values('Title').reset_index() #  Sort the Title of the books in the file by alphabetical

# Compare two string equals are not
def comparetext(str1, str2):
    i = 0
    while i < len(str1) - 1 and str1[i] == str2[i]:
        i += 1
    if str1[i] > str2[i]:
        return -1
    return str1[i] < str2[i]

```

User Input

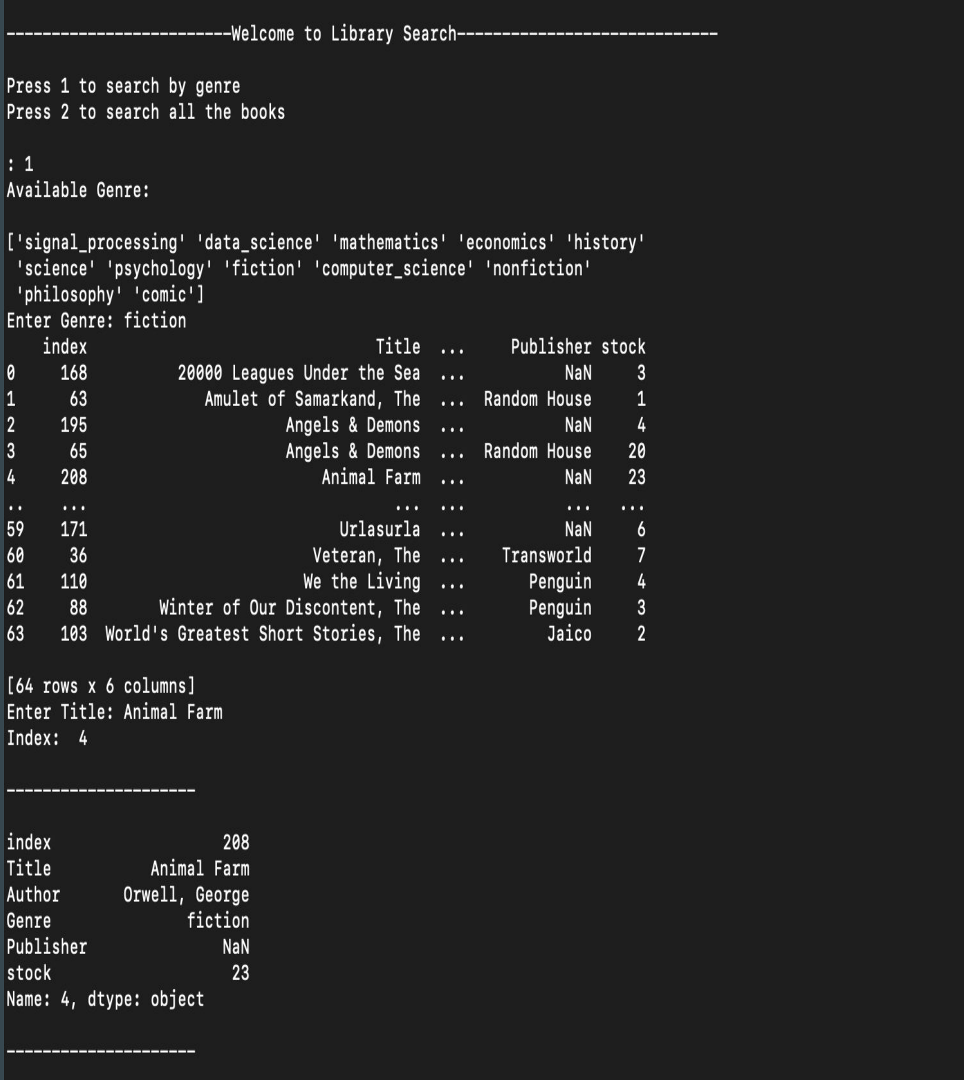
As shown on the right, users are first asked to input a 1 to search by genre, or 2 to search by book. For a genre that a user types in, the list of books would show up in alphabetical sorted order. In order to Find the specific book, 1D Binary Search is used. However, option #2 (searching by title) utilizes 2D Binary Search to help us look through the matrix.



```
while(True):
    print("\n-------------------------Welcome to Library Search-----------------------------\n")
    options = input("Press 1 to search by genre\nPress 2 to search all the books\n\n: ")
    if options == "1":
        print("Available Genre:\n")
        print(df['Genre'].unique())
        genre = input("Enter Genre: ")
        n = len(booksmap[genre]['Title'])
        print(booksmap[genre])
        searchlist = booksmap[genre]['Title'].values.tolist()
        title = input("Enter Title: ")
        i = searchbook(searchlist, title, 0, n-1)
        print("Index: ",i)
        if(i>0):
            print("\n---------------------\n")
            print(booksmap[genre].iloc[i])
            print("\n---------------------\n")
        else:
            print("Book not found!")
    elif options == "2":
        all_books = []
        for each_genre in booksmap:
            all_books.append(booksmap[each_genre]['Title'].values.tolist())

        print(all_books)
        title = input("Enter Title: ")
        m, n = binary2dSearch(all_books, title)
        if m:
            genre = list(booksmap.keys())[m]
            print("\n---------------------\n")
            print(booksmap[genre].iloc[n])
            print("\n---------------------\n")
    else:
        print("Invalid option selected\n\n")
```

Running the Application

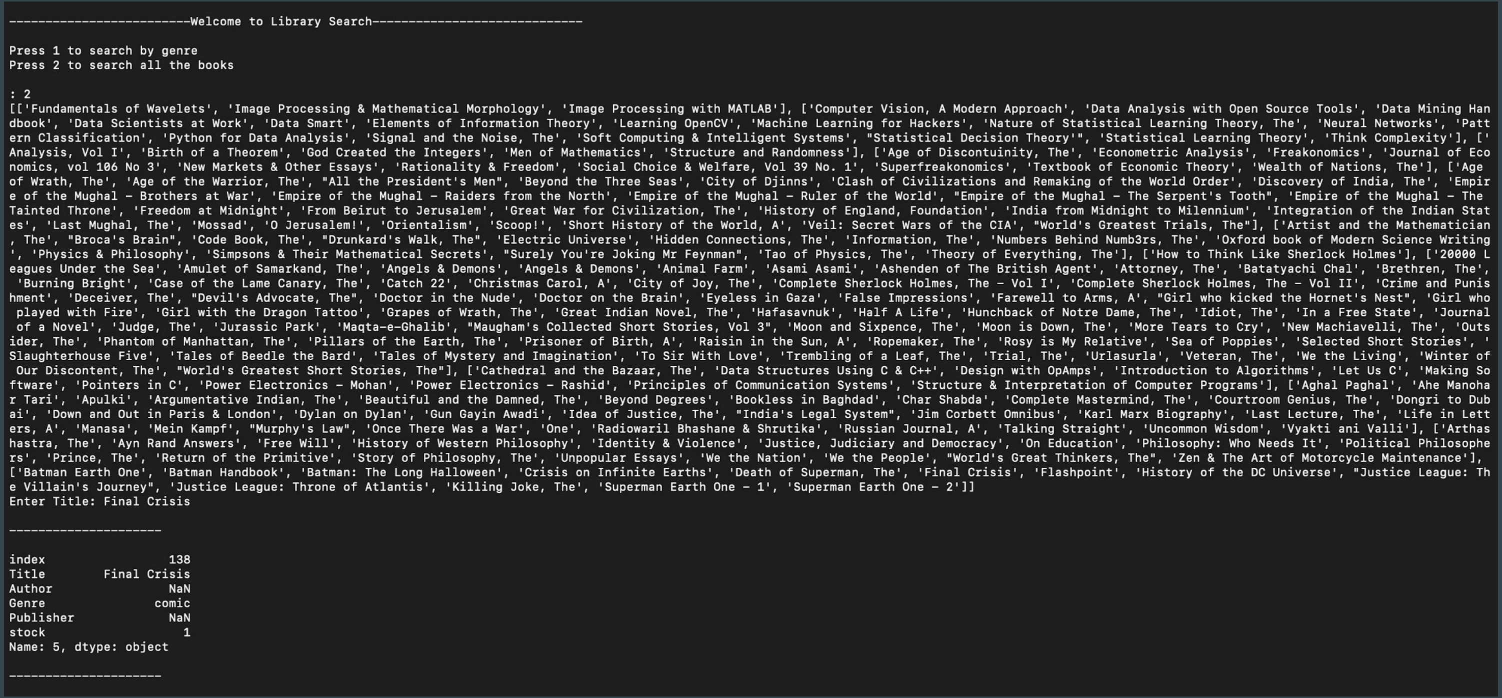
As shown, choosing a genre will allow us to see the books listed under the specific one searched, and users can type in a book title that will output a the index, title, author, genre, publisher, and stock information of it.



```
-------------------------Welcome to Library Search-----------------------------

Press 1 to search by genre
Press 2 to search all the books

: 1
Available Genre:

['signal_processing' 'data_science' 'mathematics' 'economics' 'history'
 'science' 'psychology' 'fiction' 'computer_science' 'nonfiction'
 'philosophy' 'comic']
Enter Genre: fiction
    index                                Title  ...    Publisher stock
0     168          20000 Leagues Under the Sea  ...          NaN     3
1      63             Amulet of Samarkand, The  ...  Random House     1
2     195                      Angels & Demons  ...          NaN     4
3      65                      Angels & Demons  ...  Random House    20
4     208                          Animal Farm  ...          NaN    23
..    ...                                  ...  ...          ...   ...
59    171                            Urlasurla  ...          NaN     6
60     36                         Veteran, The  ...    Transworld     7
61    110                        We the Living  ...      Penguin     4
62     88        Winter of Our Discontent, The  ...      Penguin     3
63    103  World's Greatest Short Stories, The  ...        Jaico     2

[64 rows x 6 columns]
Enter Title: Animal Farm
Index:  4

---------------------


index                   208
Title           Animal Farm
Author       Orwell, George
Genre               fiction
Publisher               NaN
stock                    23
Name: 4, dtype: object

---------------------
```

Running the Application

As shown, choosing to search by book will allow us to see all the books that were already sorted into their distinctive genres, and they are organized in a matrix format, which is the reason why we need a 2D Binary Search Algorithm in this scenario.



```
-------------------------Welcome to Library Search-----------------------------

Press 1 to search by genre
Press 2 to search all the books

: 2
[['Fundamentals of Wavelets', 'Image Processing & Mathematical Morphology', 'Image Processing with MATLAB'], ['Computer Vision, A Modern Approach', 'Data Analysis with Open Source Tools', 'Data Mining Han
dbook', 'Data Scientists at Work', 'Data Smart', 'Elements of Information Theory', 'Learning OpenCV', 'Machine Learning for Hackers', 'Nature of Statistical Learning Theory, The', 'Neural Networks', 'Patt
ern Classification', 'Python for Data Analysis', 'Signal and the Noise, The', 'Soft Computing & Intelligent Systems', "Statistical Decision Theory'", 'Statistical Learning Theory', 'Think Complexity'], ['
Analysis, Vol I', 'Birth of a Theorem', 'God Created the Integers', 'Men of Mathematics', 'Structure and Randomness'], ['Age of Discontuinity, The', 'Econometric Analysis', 'Freakonomics', 'Journal of Eco
nomics, vol 106 No 3', 'New Markets & Other Essays', 'Rationality & Freedom', 'Social Choice & Welfare, Vol 39 No. 1', 'Superfreakonomics', 'Textbook of Economic Theory', 'Wealth of Nations, The'], ['Age
of Wrath, The', 'Age of the Warrior, The', "All the President's Men", 'Beyond the Three Seas', 'City of Djinns', 'Clash of Civilizations and Remaking of the World Order', 'Discovery of India, The', 'Empir
e of the Mughal - Brothers at War', 'Empire of the Mughal - Raiders from the North', 'Empire of the Mughal - Ruler of the World', "Empire of the Mughal - The Serpent's Tooth", 'Empire of the Mughal - The
Tainted Throne', 'Freedom at Midnight', 'From Beirut to Jerusalem', 'Great War for Civilization, The', 'History of England, Foundation', 'India from Midnight to Milennium', 'Integration of the Indian Stat
es', 'Last Mughal, The', 'Mossad', 'O Jerusalem!', 'Orientalism', 'Scoop!', 'Short History of the World, A', 'Veil: Secret Wars of the CIA', "World's Greatest Trials, The"], ['Artist and the Mathematician
, The', "Broca's Brain", 'Code Book, The', "Drunkard's Walk, The", 'Electric Universe', 'Hidden Connections, The', 'Information, The', 'Numbers Behind Numb3rs, The', 'Oxford book of Modern Science Writing
', 'Physics & Philosophy', 'Simpsons & Their Mathematical Secrets', "Surely You're Joking Mr Feynman", 'Tao of Physics, The', 'Theory of Everything, The'], ['How to Think Like Sherlock Holmes'], ['20000 L
eagues Under the Sea', 'Amulet of Samarkand, The', 'Angels & Demons', 'Angels & Demons', 'Animal Farm', 'Asami Asami', 'Ashenden of The British Agent', 'Attorney, The', 'Batatyachi Chal', 'Brethren, The',
 'Burning Bright', 'Case of the Lame Canary, The', 'Catch 22', 'Christmas Carol, A', 'City of Joy, The', 'Complete Sherlock Holmes, The - Vol I', 'Complete Sherlock Holmes, The - Vol II', 'Crime and Punis
hment', 'Deceiver, The', "Devil's Advocate, The", 'Doctor in the Nude', 'Doctor on the Brain', 'Eyeless in Gaza', 'False Impressions', 'Farewell to Arms, A', "Girl who kicked the Hornet's Nest", 'Girl who
 played with Fire', 'Girl with the Dragon Tattoo', 'Grapes of Wrath, The', 'Great Indian Novel, The', 'Hafasavnuk', 'Half A Life', 'Hunchback of Notre Dame, The', 'Idiot, The', 'In a Free State', 'Journal
 of a Novel', 'Judge, The', 'Jurassic Park', 'Maqta-e-Ghalib', "Maugham's Collected Short Stories, Vol 3", 'Moon and Sixpence, The', 'Moon is Down, The', 'More Tears to Cry', 'New Machiavelli, The', 'Outs
ider, The', 'Phantom of Manhattan, The', 'Pillars of the Earth, The', 'Prisoner of Birth, A', 'Raisin in the Sun, A', 'Ropemaker, The', 'Rosy is My Relative', 'Sea of Poppies', 'Selected Short Stories', '
Slaughterhouse Five', 'Tales of Beedle the Bard', 'Tales of Mystery and Imagination', 'To Sir With Love', 'Trembling of a Leaf, The', 'Trial, The', 'Urlasurla', 'Veteran, The', 'We the Living', 'Winter of
 Our Discontent, The', "World's Greatest Short Stories, The"], ['Cathedral and the Bazaar, The', 'Data Structures Using C & C++', 'Design with OpAmps', 'Introduction to Algorithms', 'Let Us C', 'Making So
ftware', 'Pointers in C', 'Power Electronics - Mohan', 'Power Electronics - Rashid', 'Principles of Communication Systems', 'Structure & Interpretation of Computer Programs'], ['Aghal Paghal', 'Ahe Manoha
r Tari', 'Apulki', 'Argumentative Indian, The', 'Beautiful and the Damned, The', 'Beyond Degrees', 'Bookless in Baghdad', 'Char Shabda', 'Complete Mastermind, The', 'Courtroom Genius, The', 'Dongri to Dub
ai', 'Down and Out in Paris & London', 'Dylan on Dylan', 'Gun Gayin Awadi', 'Idea of Justice, The', "India's Legal System", 'Jim Corbett Omnibus', 'Karl Marx Biography', 'Last Lecture, The', 'Life in Lett
ers, A', 'Manasa', 'Mein Kampf', "Murphy's Law", 'Once There Was a War', 'One', 'Radiowaril Bhashane & Shrutika', 'Russian Journal, A', 'Talking Straight', 'Uncommon Wisdom', 'Vyakti ani Valli'], ['Arthas
hastra, The', 'Ayn Rand Answers', 'Free Will', 'History of Western Philosophy', 'Identity & Violence', 'Justice, Judiciary and Democracy', 'On Education', 'Philosophy: Who Needs It', 'Political Philosophe
rs', 'Prince, The', 'Return of the Primitive', 'Story of Philosophy, The', 'Unpopular Essays', 'We the Nation', 'We the People', "World's Great Thinkers, The", 'Zen & The Art of Motorcycle Maintenance'],
['Batman Earth One', 'Batman Handbook', 'Batman: The Long Halloween', 'Crisis on Infinite Earths', 'Death of Superman, The', 'Final Crisis', 'Flashpoint', 'History of the DC Universe', "Justice League: Th
e Villain's Journey", 'Justice League: Throne of Atlantis', 'Killing Joke, The', 'Superman Earth One - 1', 'Superman Earth One - 2']]
Enter Title: Final Crisis

---------------------

index                 138
Title        Final Crisis
Author                NaN
Genre               comic
Publisher             NaN
stock                   1
Name: 5, dtype: object

---------------------
```

Searching a book in a library can be very efficiently done using Divide and Conquer algorithm if the library is organized correctly in the real world.

If the library is not organized and the books are randomly placed, one can only pray to find the book they’re looking for even if it is present in the library.

Thank you for listening to
our presentation!