VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA,BELAGAVI – 590018

KARNATAKA

Mini Project Report
On

"ARRANGING THE BOOKS IN LIBRARY USING SELECTION SORT"

SUBMITTED IN PARTIAL FULFILLMENT OF THE ASSIGNMENT

FOR THE Analysis & Design of Algorithms(BCS401)
COURSE OF IV SEMESTER

Submitted by
CHANDANA T.S
[1CG22CS022]

Guide:
Mr.Asif Ulla Khan.M. Tech.
Asst. Prof., Dept. of CSE
CIT, Gubbi.

HOD:
Dr. Shantala C P PhD.,
Head, Dept. of CSE
CIT, Gubbi.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

C.I.T
Partnering in Academic Excellence

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

2023-24

Rubric – B.E. Mini-Project [BCS401]

Course outcome	Rubric/Level	Excellent (91-100%)	Good (81-90%)	Average (61-80%)	Moderate (40-60%)	Score
CO1	Identification of project proposal (05 Marks)					
CO2	Design and Implementation (10 Marks)					
CO3	Presentation skill (05 Marks)					
CO4	Individual or in a team development					
CO5	Report (05 Marks)					
Total						

Course outcome:

CO 1: Identification of project proposal which is relevant to subject of engineering.

CO 2: Design and implement proposed project methodology.

CO 3: Effective communication skill to assimilate their project work.

CO 4: Work as an individual or in a team in development of technical projects.

CO 5: Understanding overall project progress and performance.

Student Signature **Faculty signature**

INDEX

ABSTRACT

Sorting books in a library is essential for efficient retrieval and management. This project demonstrates the use of the Selection Sort algorithm to arrange a list of books in ascending order based on their titles or identification numbers. Selection Sort operates by repeatedly finding the smallest element from the unsorted section of the list and moving it to the end of the sorted section. This method, while straightforward and easy to implement, has a time complexity of O(n^2), making it suitable for small datasets. The project outlines the algorithmic steps involved, provides a practical example of its application, and discusses its advantages and limitations in the context of library book organization.

CHAPTER 1:

INTRODUCTION

Effective organization of books is crucial in a library setting to ensure quick and easy access to information. One fundamental approach to achieving this is by sorting the books based on their titles or identification numbers. Selection Sort, a simple and intuitive sorting algorithm, is well-suited for this task, particularly in scenarios involving relatively small collections.

For instance, if we have a list of books sorted by their titles or unique identification numbers, Selection Sort would start by finding the smallest book in the entire list and placing it in the first position. Then it would proceed to find the smallest book from the remaining unsorted section and place it in the next position, continuing this process until the entire list is sorted.

While Selection Sort is easy to understand and implement, it is not the most efficient for large collections due to its O(n^2) time complexity. However, for smaller libraries or for educational purposes, it provides a clear demonstration of basic sorting principles and is a good starting point for understanding more complex sorting algorithms.

Selection Sort functions by dividing the list of books into two sections: the sorted section and the unsorted section. It selects the smallest (or largest) book from the unsorted section and places it at the end of the sorted section. This process continues until the entire list is sorted. Although Selection Sort is straightforward and easy to implement, it is less efficient for large datasets due to its O(n^2) time complexity.

This project explores the application of Selection Sort to organize a library's book collection. It demonstrates the algorithm's steps and examines its effectiveness in sorting books, providing insights into its practical use and limitations. Through this example, we highlight the role of basic sorting algorithms in data organization and their relevance in library management systems.

CHAPTER 2:

PROBLEM STATEMENT

In a library, effective book organization is crucial for efficient retrieval and management. The objective is to sort a given list of books, identified by their titles or identification numbers, in ascending order. To achieve this, the Selection Sort algorithm will be employed. Selection Sort is chosen for its simplicity and ease of implementation, making it suitable for small to moderate-sized datasets. The goal is to enhance the organization of the library's book collection to improve accessibility and systematic arrangement.

CHAPTER 3:

IMPLEMENTATION

Analysis:

- **Time Complexity**: O(n2)O(n^2)O(n2) in all cases, where nnn is the number of books. This is because selection sort has nested loops: one loop goes through each element (n times), and inside it, another loop finds the minimum (n-1 comparisons in worst case).
- **Space Complexity**: O(1)O(1)O(1) extra space, as selection sort is an in-place sorting algorithm.
- **Stability**: Selection sort is not stable, meaning that the relative order of equal elements may not be preserved.

Algorithm:

1. **Initialization**:

   - Set iii to 0 (start of the unsorted part of the array).

2. **Repeat the following steps until** iii **reaches** n−1n-1n−1:

   - Find the index min_index\text{min_index}min_index of the smallest book title in the unsorted part of the array A[i]A[i]A[i] to A[n−1]A[n-1]A[n−1].

     - Initialize min_index\text{min_index}min_index to iii.
     - Iterate through the array from i+1i+1i+1 to n−1n-1n−1.
     - If A[j]A[j]A[j] (where jjj is the current index) is smaller than A[min_index]A[\text{min_index}]A[min_index], update min_index\text{min_index}min_index to jjj.

3. **Swap**: Swap A[i]A[i]A[i] with A[min_index]A[\text{min_index}]A[min_index].

     - If min_index≠i\text{min_index} \neq imin_index =i, swap A[i]A[i]A[i] with A[min_index]A[\text{min_index}]A[min_index].

PROGRAM

This program provides the solution for the above problem statement:

```
#include <stdio.h>
#include <string.h>

// Function to perform selection sort on an array of strings
void selectionSort(char arr[][50], int n) {
   int i, j, min_idx;
   char minTitle[50];

   // Traverse through all book titles
   for (i = 0; i < n - 1; i++) {
       // Find the index of the smallest title in the remaining unsorted part
      min_idx = i;
      strcpy(minTitle, arr[i]);

      for (j = i + 1; j < n; j++) {
         if (strcmp(arr[j], minTitle) < 0)
            { min_idx = j;
            strcpy(minTitle, arr[j]);
         }
      }

      // Swap the found smallest title with the first title of the unsorted part
      if (min_idx != i) {
         char temp[50];
         strcpy(temp, arr[i]);
         strcpy(arr[i], arr[min_idx]);
         strcpy(arr[min_idx], temp);
       }
   }
}
```

```
// Function to print the array of book titles
void printArray(char arr[][50], int size) {
    int i;
    printf("Sorted Book Titles:\n");
    for (i = 0; i < size; i++) {
        printf("%s\n", arr[i]);
    }
}

int main() {
    // Example array of book titles
    char books[][50] = {
        "The Great Gatsby",
        "To Kill a Mockingbird",
        "1984",
         "Pride and Prejudice",
        "Moby Dick"
    };

    int n = sizeof(books) / sizeof(books[0]);

    printf("Original Book Titles:\n");
    printArray(books, n);

    // Perform selection sort
    selectionSort(books, n);

    // Print sorted array
     printArray(books, n);

    return 0;
}
```

Code explanation and data types used:

1. "`selectionSort`" Function:

   - This function sorts the array of book titles in alphabetical order. It uses the selection sort algorithm, which works by finding the smallest (or largest) item from the unsorted portion of the array and swapping it with the first unsorted item.

2. "`printArray`" Function:

   - This function prints the elements of the array. It's used to display both the original and sorted lists of book titles.

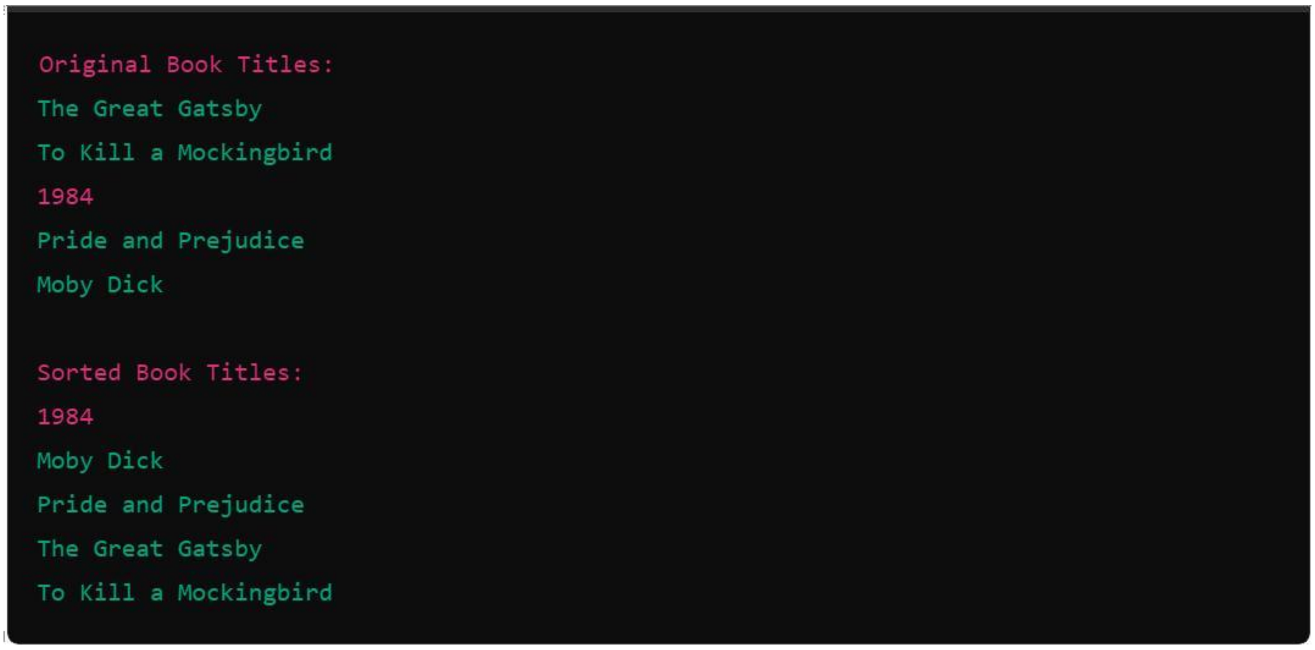
3. "`main`" Function:

   - Initializes an example array of book titles.
   - Calculates the number of books.
   - Prints the original list of book titles.
   - Sorts the book titles using the `selectionSort` function.
   - Prints the sorted list of book titles.

CHAPTER 4:

RESULT/SCREENSHOTS

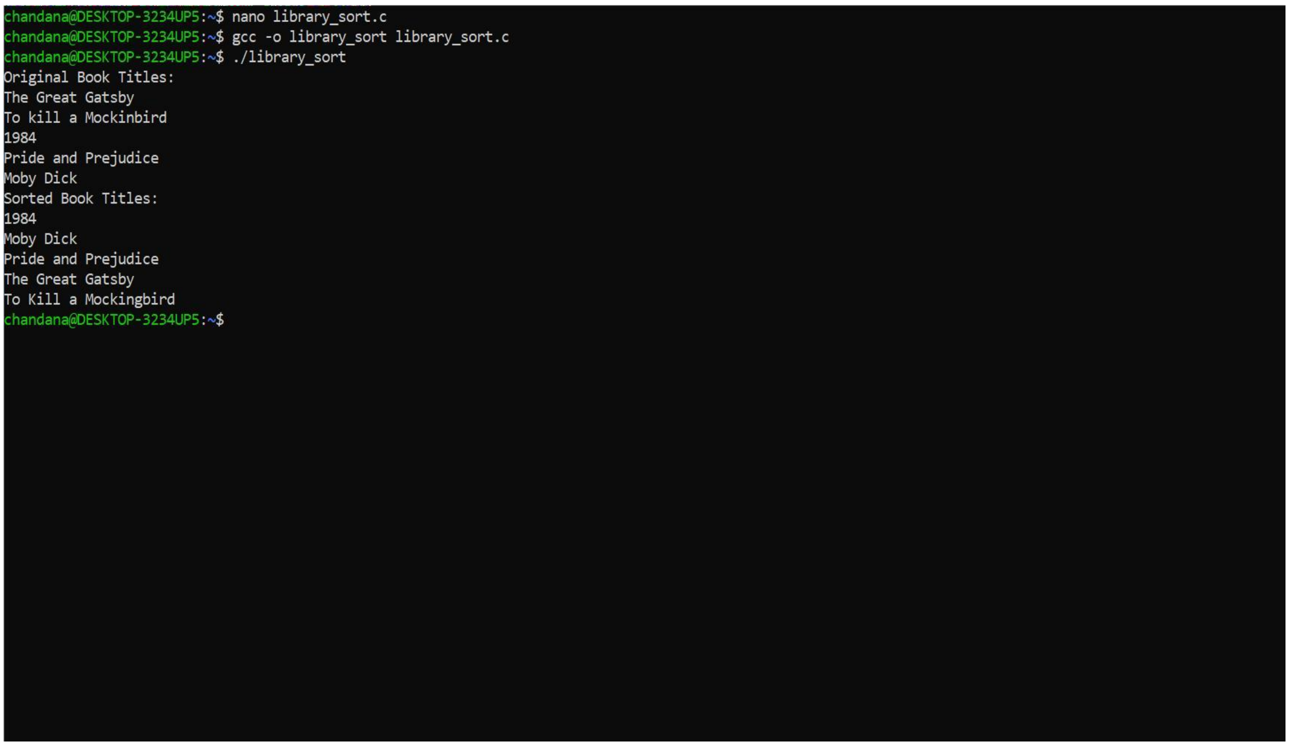
SCREENSHOT 1:



```
Original Book Titles:
The Great Gatsby
To Kill a Mockingbird
1984
Pride and Prejudice
Moby Dick

Sorted Book Titles:
1984
Moby Dick
Pride and Prejudice
The Great Gatsby
To Kill a Mockingbird
```

SCREENSHOT 2:



```
chandana@DESKTOP-3234UP5:~$ nano library_sort.c
chandana@DESKTOP-3234UP5:~$ gcc -o library_sort library_sort.c
chandana@DESKTOP-3234UP5:~$ ./library_sort
Original Book Titles:
The Great Gatsby
To kill a Mockinbird
1984
Pride and Prejudice
Moby Dick
Sorted Book Titles:
1984
Moby Dick
Pride and Prejudice
The Great Gatsby
To Kill a Mockingbird
chandana@DESKTOP-3234UP5:~$
```

CHAPTER 5:

CONCLUSION

In this project, we implemented the Selection Sort algorithm to organize a library's book collection in ascending order based on their titles. Selection Sort, while straightforward and easy to understand, demonstrated its utility in sorting small to moderate-sized datasets. The algorithm operates by repeatedly finding the smallest element in the unsorted portion of the list and moving it to the sorted section, which effectively ordered the book titles alphabetically.

Through the application of Selection Sort, we observed that while the algorithm ensures accurate sorting, its O(n^2) time complexity makes it less suitable for very large datasets due to slower performance. For larger collections, more advanced sorting algorithms such as Merge Sort or Quick Sort may be more efficient. Nevertheless, Selection Sort serves as a useful educational tool to grasp fundamental sorting principles and its implementation can aid in straightforward book organization tasks within a library setting.

Overall, the use of Selection Sort in this context provides a clear and manageable solution for arranging books, ensuring that the library's collection is well-organized and easily accessible.

.

REFERENCE

1. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
2. https://www.geeksforgeeks.org/selection-sort/
3. https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_program_in_c.htm
4. "Data Structures and Algorithm Analysis in C" by Mark Allen Weiss