20/8/25

Task #1 Implent various searching and sorting operations in python programming

Library Book search (Linear Search/ Binary search)

Aim:

To write a python program that searches for a book entered by the user using linear search and, if the list is sorted allows searching using Binary search.

Algorithm:

Linear search:

- Start with the first element in the list
- Compare each element with the search item
- if a match is found, return its position.
- if not found till the end, return 'not found',

Binary search

- Set low = 0, high = len(list) - 1
- Find mid = (low + high) // 2
- if list[mid] == key, return mid.
- if key < list[mid], search in the left half
- if key > list[mid], search in the right half.
- Repeat until found or low > high.

Program:

```
def linear_search(book_list, key):
    for i in range(len(book_list)):
        if book_list[i].lower() == key.lower():
            return i
    return -1

def binary_search(book_list, key):
    low, high = 0, len(book_list)-1
    while low <= high:
        mid = (low + high) // 2
```

output:

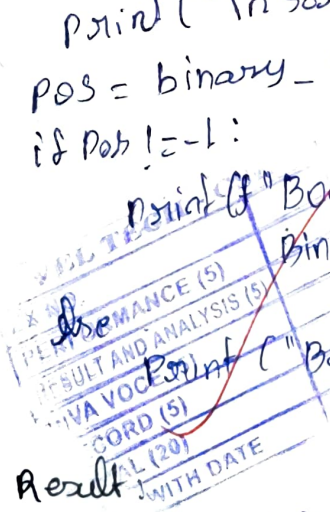
Books in library: ['python programming', 'data structure', 'algorithms', 'machine learning', 'Artificial intelligence']

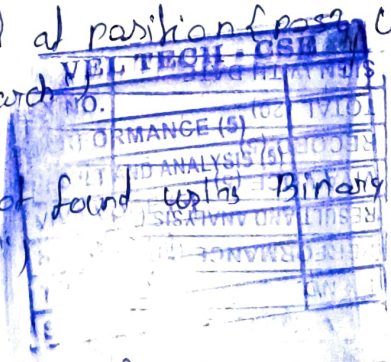
Enter the book to search: data science
Book found at position 1 using linear search
Sorted books: ['algorithm', 'artificial intell', 'data science', 'machine learning', 'python programming']
Book found at position 2 using binary search

```
        if book_list[mid].lower() == key.lower():
            return mid
        elif key.lower() < book_list[mid].lower():
            high = mid - 1
        else:
            low = mid + 1
    return -1

books = ["python programming", "data structures",
"algorithms", "machine learning", "Artificial
intelligence"]
print("Books in library:", books)
search_book = input("Enter the book to search:")
pos = linear_search(books, search_book)
if pos != -1:
    print(f"Book found at position {pos} using
          Linear Search")
else:
    print("Book not found using Linear
          search")

book.sort()
print("In Sorted Books:", books)
pos = binary_search(books, search_book)
if pos != -1:
    print(f"Book found at position {pos} using
          Binary search")
else:
    print("Books not found using Binary
          search")
```





Result

The program to search for a book using linear and binary search executed successfully

b, Student Grade Organizer (Bubble selection sort)

Aim:

To write a program that sorts student's grades using Bubble sort and selection sort, and displayed the top 3 scores.

Algorithm:

Bubble sort (Ascending):

1. Start
2. Input the list of grades.
3. Repeat for each element:
   - Compare adjacent elements.
   - Swap if they are in the wrong order.
4. Continue until the list is sorted
5. Stop

Selection sort (Descending):

1. Start
2. Input the list of grades
3. For each index i:
   - Find the maximum element in the unsorted part.
   - Swap it with element at index i.
4. Continue until the list is sorted
5. Stop

Program

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return n arr
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        max_idx = i
```

output.

Grades in Ascending order (Bubble sort):

[45, 56, 67, 78, 88, 99]

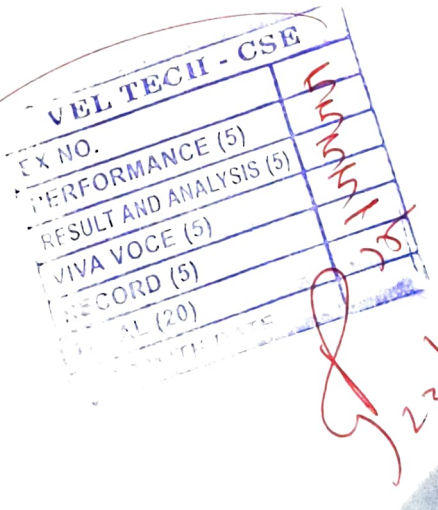
Grades in descending order (selection sort):

[99, 88, 78, 67, 56, 45]

Top 3 scores: [99, 88, 78]

```
    for j in range(i+1, n):
        if arr[j] > arr[max_idx]:
            max_idx = j
    arr[i], arr[max_idx] = arr[max_idx], arr[i]
return arr

grades = [45, 78, 88, 56, 99, 67]
ascend = bubble_sort(grades.copy())
print("Grades in Ascending order (Bubble sort):",
      ascend)
descend = selection_sort(grades.copy())
print("Grades in Descending order (selection sort):",
      descend)
print("Top 3 scores:", descend[:3])
```



Result

Thus, the program to sort student grades in ascending order using Bubble sort, descending order using selection sort and display top 3 scores are successfully executed