

5	20108/25	Implementation of various searching and sorting operations in python programming.
		15 Date: 09/09/2010

2018/25 Task-5: Implement various searching and sorting operations In Python Programming

(a) Library Book Search

Aim: To search for a book in a library catalog using linear search for unsorted lists and binary search for sorted lists.

Algorithm:

1. Linear Search:

- Check each book in the list one by one.
- If found, return its position.
- If not found after checking all books, return -1.

2. Binary search:

- Requires the list to be sorted.
- Compare the target with middle element.
- If equal, return the position.
- If target is smaller, search the left half.
- If target is larger, search the right half.
- Repeat until found or search space exhausted.

Program:

```
# Library Book Search
def linear_search(books, target):
    for i in range(len(books)):
        if books[i] == target:
            return i
    return -1

def binary_search(books, target):
    low, high = 0, len(books) - 1
    while low < high:
        mid = (low + high) // 2
        if books[mid] == target:
            return mid
        elif books[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

written books problem solved  
Ex: linear search with array

Algorithm steps  
1. Input list  
2. Set position = 0  
3. If position <= list.length - 1  
    a. If list[position] == target  
        return position  
    b. Else position = position + 1  
4. Else return -1

Time complexity  
Best case: O(1)  
Worst case: O(n)

Advantages:  
1. Simple to implement.  
2. No extra space required.  
3. Works well for small lists.

Disadvantages:  
1. Not suitable for large lists.  
2. Not efficient for sorted arrays.  
3. Time complexity is O(n).

Most FST uses binary search for sorted arrays.

Most IDEs use binary search for sorted arrays.

Abstract class BinarySearch defined in search library package.

### Output:

Library Books : [python', 'Java', 'C++', 'JavaScript', 'HTML']

List is sorted : False

Enter book to search : C++

Book found at position 3 : (using linear search)

: target is found

: target is not found

: target is not found

: target is found

: target is not found

```
books = ["Python", "Java", "C++", "JavaScript", "HTML"]
print ("Library Books:", books)
is_sorted = books == sorted (books)
print ("List is sorted:", is_sorted)
target = input ("Enter book to search:")
if is_sorted:
    result = binary_search (books, target)
    method = "Binary Search"
else:
    result = linear_search (books, target)
    method = "Linear Search"
if result != -1:
    print ("Book found at position", result + 1, "using", method)
else:
    print ("Book not found")
```

### Result:

The program to implement both search algorithms, to detect that the list was not sorted and used linear search to find "C++" at position 3 was successfully executed.

1/2/25

(b) Student Grade Organizer

Aim: To sort student grades using different algorithms and display the top 3 scores.

Algorithm:-

1. Bubble Sort (Ascending):

- Compare adjacent elements & swap it in wrong order
- Repeat until no more swaps are needed.

2. Selection Sort (Descending):

- Find the max element and swap it first position.
- Repeat for remaining elements

3. Top 3 scores:

- After sorting in descending order, the first three elements are the top scores.

Program:-

```

def bubble_sort_asc(grades): n = len(grades)
    for i in range(n):
        for j in range(0, n-i-1):
            if grades[j] > grades[j+1]:
                grades[j], grades[j+1] = grades[j+1], grades[j]
    return grades

def selection_sort_desc(grades):
    n = len(grades)
    for i in range(n):
        max_idx = i
        for j in range(i+1, n):
            if grades[j] > grades[MAX_idx]:
                max_idx = j
        grades[i], grades[max_idx] = grades[max_idx], grades[i]

grades = [85, 92, 78, 90, 65, 88, 72]
print("Original grades:", grades)
asc_sorted = bubble_sort_asc(grades.copy())
desc_sorted = selection_sort_desc(grades.copy())
print("Ascending order (Bubble sort):", asc_sorted)

```

~~1970-1980~~ 1980-1990 1990-2000 2000-2010

## Output:-

Original Grades : [85, 92, 78, 90, 65, 88, 72]  
 Ascending order (Bubble sort) : [65, 72, 78, 85, 88, 90, 92]  
 Descending order (selection sort) : [92, 90, 88, 85, 78, 72, 65]  
 Top 3 scores : [92, 90, 88]

```
print("Descending order (selection sort):", desc_sorted)
print("Top 3 scores:", desc_sorted[:3])
```

Output

descending sorted list

28, 40, 42

4 P, 20, 22

8 P, 30, 32

descending list printed with

28, 40, 42

4 P, 20, 22

8 P, 30, 32

26, 36, 38

VEL TECH - CSE	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
CORD (5)	5
GRATUITY (20)	15
INSTITUTE DATE	1/25/2023

Result:-

Thus, the student grade organizer in both ascending and descending order using different algorithms are executed successfully.