Assignment 2

PMKVY

QUESTIONS

1. Linear Search Algorithm

2. Python Program for Linear Search

3. Algorithm for Binary Search (Iterative Method)

4. Algorithm for Binary Search (Recursive Method)

5. Python Code for Binary Search (Iterative Method)

6. Python Code for Binary Search (Recursive Method)

7. Difference Between Linear Search and Binary Search

SOLUTIONS

Ans1:

Linear Search ( Array A, Value x)

steps:

1: Set i to 1

2: if i > n then go to step 7

3: if A[i] = x then go to step 6

4: Set i to i + 1

5: Go to Step 2

6: Print Element x Found at index i and go to step 8

7: Print element not found

8: Exit

ans 2:

def linear\_search(arr, search):

for index, element in enumerate(arr):

if element == search:

return index

return -1

arr = [5, 2, 9, 1, 5, 6]

search\_element = 9

result = linear\_search(arr, search\_element)

if result != -1:

print(f"Search element {search\_element} found at index {result}.")

else:

print("Search element not found in the array.")

ans 3:

1. ​​Initialize start as the leftmost index of the array and end as the rightmost index of the array.
2. Repeat while start is less than or equal to end:
3. Calculate the middle index: mid = (start + end) // 2.
4. If the middle element is equal to the target element, return mid.
5. If the middle element is less than the target element, update start to mid + 1.
6. If the middle element is greater than the target element, update end to mid - 1.

* If the loop completes without finding the target element, return -1 to indicate that the element is not present in the array.

Ans4:

* 1. If the start index is greater than the end index, return -1 to indicate that the target element is not present in the array.
* 2. Calculate the middle index: mid = (start + end) // 2.
* 3. If the middle element is equal to the target element, return mid.
* 4. If the middle element is less than the target element, recursively call the binary search function with start updated to mid + 1.
* 5. If the middle element is greater than the target element, recursively call the binary search function with end updated to mid - 1.

Ans5:

def binary\_search(arr, search):

start = 0

end = len(arr) - 1

while start <= end:

mid = (start + end) // 2

if arr[mid] == search:

return mid

elif arr[mid] < search:

start = mid + 1

else:

end = mid - 1

return -1

arr = [1, 2, 5, 6, 9, 12, 15, 18, 21]

search\_element = 9

result = binary\_search(arr, search\_element)

if result != -1:

print(f"Search element {search\_element} found at index {result}.")

else:

print("Search element not found in the array.")

ans 6:

def binary\_search\_recursive(arr, search, start, end):

if start > end:

return -1

mid = (start + end) // 2

if arr[mid] == search:

return mid

elif arr[mid] < search:

return binary\_search\_recursive(arr, search, mid + 1, end)

else:

return binary\_search\_recursive(arr, search, start, mid - 1)

def binary\_search(arr, search):

return binary\_search\_recursive(arr, search, 0, len(arr) - 1)

arr = [1, 2, 5, 6, 9, 12, 15, 18, 21]

search\_element = 9

result = binary\_search(arr, search\_element)

if result != -1:

print(f"Search element {search\_element} found at index {result}.")

else:

print("Search element not found in the array.")

ans 7:

Linear Search:

* Method: Like looking for a name in a phone book one by one.
* Process: Start at the beginning, check each name, and ask "Is this the name?" Keep going until you find the name or reach the end.
* Efficiency: Good for small lists, but can be slow for big lists because you have to check each item.
* Sorting: Doesn't require the list to be in any order.

Binary Search:

* Method: Like finding a word in a dictionary by flipping to the middle pages.
* Process: If the list is sorted, you start in the middle and ask "Is the item I'm looking for before or after this?" You narrow down your search by half each time until you find the item.
* Efficiency: Great for big sorted lists, because you eliminate half of the remaining options with each step.
* Sorting: Requires the list to be sorted, otherwise it won't work correctly.