25 September 2023 19:28

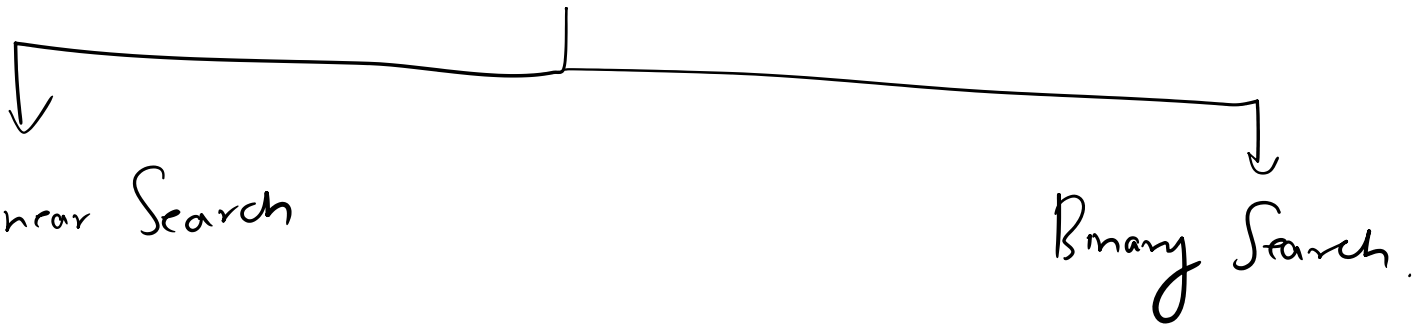
Seaching & Sorting (DSA) :-

(1) Seaching Based Algorithm

(2) Sorting Based Algorithms

(1) Searching Based Algorithms :-

lst :- [ ] , value

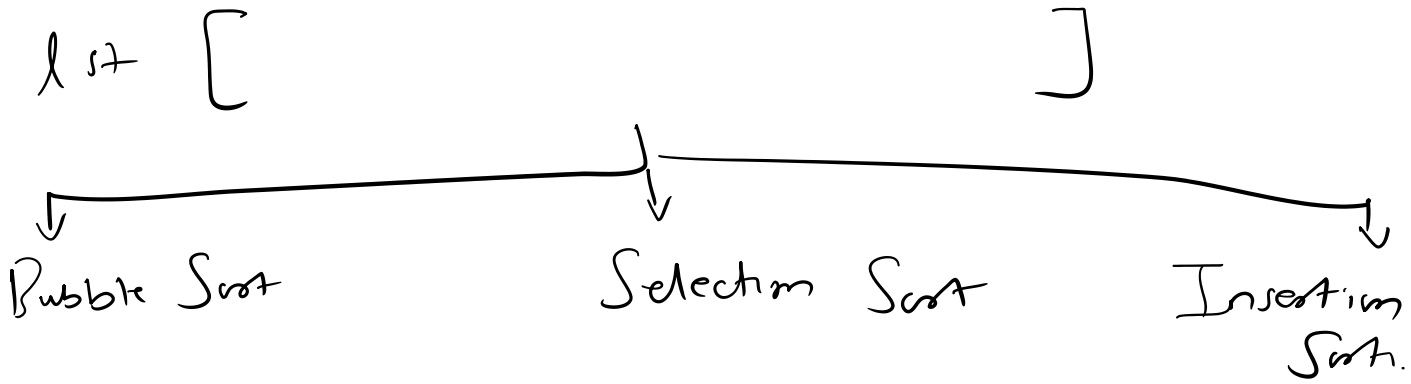


Input :- A list of nos. & a value.

Task :- Check whether the value exists in the list

O/p :- If value found in this list,
you should return a Success.
else Not found.

Sorting Algorithm :-



25 September 2023 19:40

Searching :- Linear Search Algorithm :-

Eg:- To understand the working of Linear Search.

lst:- [70, 40, 30, 11, 57, 41, 25, 14, 52]

Value to be searched :- k = 41

Task :- To find whether 41 is a present in the lst.

0	1	2	3	4	5	6	7	8
70	40	30	11	57	41	25	14	52

Soln :- lst = [70, 40, 30, 11, 57, 41, 25, 14, 52]

1st element lst[0] = k
70 ≠ 41

2nd element lst[1] = k
40 ≠ 41

3rd element lst[2] = k
30 ≠ 41

4th element lst[3] = k
11 ≠ 41

5th element lst[4] = k (?)
57 ≠ 41

6th element lst[5] = k ?
✓ 41 = 41

Print the index of value : 5

If value not present in the list

Print :- Value Not found.

Pseudo Code :-

```
function linear_search(list, target):
    for each item in list:
        if item equals target value:
            return the index of the item.
    return -1  // target value not found.
```

Binary Search:-

eg:- [10, 12, 24, 29, 39, 40, 51, 56, 59]
(indices 0 1 2 3 4 5 6 7 8)

target value :- 56

start index = 0
end index = 8
middle index = $\frac{0+8}{2} = 4$

{Compare the value at middle index & the target value

arr[4] = 39
K = 56
39 < 56 }

(1) Check whether value at middle index = k value
(2) Check whether value at middle index > k value
(3) Check whether value at middle index < k value

start index = 5
end index = 8
middle_index = $\frac{5+8}{2} = \frac{13}{2} \simeq 7$ (6.5)

(1) Check value at middle index = k value
(2) Check value at middle index > k value [search left]
(3) check value at middle index < k value. [search right]

arr[7] = 56, k = 56.

arr[7] = 26 , k = 26.

Print the index of target = 7

eg:-2] $\underset{0}{10}, \underset{1}{15}, \underset{2}{23}, \underset{3}{37}, \underset{4}{49}, \underset{5}{71}, \underset{6}{82}, \underset{7}{98}, \underset{8}{105}$

target value = 23

Soln:- st_ind = 0, end_ind = 8

mid_ind = $\frac{0+8}{2} = 4$

arr[4] = 49, K = 23.

Since :- K < arr[mid_ind] search for K only to the left

st_ind = 0, end_ind = 3 $= \frac{0+3}{2} = 1.5 \approx 2$

arr[2] = 23, K = 23

Print the index value = 2

eg 3 :- $\underset{0}{25}, \underset{1}{5}, \underset{2}{30}, \underset{3}{15}, \underset{4}{48}, \underset{5}{63}, \underset{6}{22}, \underset{7}{75}$

target value = 22

st_ind = 0
end_ind = 7

mid_ind = $\frac{0+7}{2} = 4$

arr[4] = 48 K = 22

48 > 22

Note :- For Binary Search,

Array must be sorted

Practical Code

Pseudo Code :-

```
function binary_search(sorted_list, target):
    start_ind = 0
    end_ind = length of the list - 1

    While start_ind <= end_ind:
        mid_ind = (start_index + end_ind)/2

        if sorted_list[mid_ind] == target:
            return mid_ind

        elif sorted_list[mid_ind] < target:
            start_ind = mid_ind + 1  // target is in the right half

        else:
            end_ind = mid_ind - 1  // target is in the left half

    return -1
```

Bubble Sort :-

eg :- [13, 32, 26, 35, 10]

I st Iteration :-

1st time :- 13, 32 — 13 < 32 ✓

2nd time :- 32, 26 — 32 < 26 ✗

[13, 26, 32, 35, 10]

3rd :- 32, 35 — 32 < 35 ✓

4th :- 35, 10 — 35 < 10 ✗

[13, 26, 32, 10, 35]

II nd Iteration :-

13 26 — 13 < 26 ✓

26 32 — 26 < 32 ✓

32 10 — 32 < 10 ✗ [13, 26, 10, 32, 35]

32 35 — 32 < 35 ✓

[13, 26, 10, 32, 35]

III rd iteration :-

13 26 — 13 < 26 ✓

26 10 — 26 < 10 ✗

[13, 10, 26, 32, 35]

26 32 — 26 < 32 ✓

32 35 — 32 < 35 ✓

[13, 10, 26, 32, 35]

IV th Iteration :-

13 10 — 13 < 10 ✗

IV th Iteration :- 13 10 13 < 10 ×

[10, 13, 26, 32, 35]

13 26 13 < 26 ✓

26 32 26 < 32 ✓

32 35 32 < 35 ✓

[10, 13, 26, 32, 35]

Selection Sort :-

eg :- [12, 29, 25, 8, 32, 17, 40]
(0, 1, 2, 3)

Sort the above list.

Soln :- 12 < 29 ✓ 12 < 25 ✓

12 < 8 ✗ [8, 29, 25, 12, 32, 17, 40]
(0, 1, 2, 3, 4, 5, 6)

8 < 29 ✓ 8 < 25 ✓, 8 < 12 ✓, 8 < 32 ✓, 8 < 17 ✓, 8 < 40 ✓

29 < 25 ✗ [8, 25, 29, 12, 32, 17, 40]

25 < 29 ✓, 25 < 12 ✗

[8, 12, 29, 25, 32, 17, 40]

12 < 29 ✓, 12 < 25 ✓, 12 < 32 ✓ 12 < 17 ✓, 12 < 40 ✓

29 < 25 ✗ [8, 12, 25, 29, 32, 17, 40]

25 < 29 ✓, 25 < 32 ✓, 25 < 17 ✗

[8, 12, 17, 29, 32, 25, 40]

17 < 29 ✓, 17 < 32 ✓, 17 < 25 ✓,

29 < 32 ✓ 29 < 25 ✗

[8, 12, 17, 25, 32, 29, 40]

25 < 32 ✓, 25 < 29 ✓ 25 < 40 ✓

25 < 32 ✓, 25 < 29 ✓, 25 < 40 ✓

3 < < 29 ×

[8, 12, 17, 25, 29, 32, 40]

29 < 32 ✓ 29 < 40 ✓

32 < 40 ✓

Selection Sort:-

[12, 31, 25, 8, 32, 17]