- Linear Search :-

- Linear search is also called Sequential search.
- This is the one where a key k, is searched in a linear list L.
- The linear search can be performed either on ordered linear search or unordered linear search.

a) ordered linear search:

- In the ordeed linear list of data elements, if the key is to be searched, it is given as K.
- In case of unsuccessful search, it prints "key not found". And in case of successful search, it prints "key found", along with an index i.
- To search for a key element K, linear list need to be compared with each of K_i.

- So, as $k > k_i$, data elements are compared.
- And if $k \le k_i$, and if $k = k_i$, it denotes search is completed and element is found.

Algorithm :-

```
Procedure Linear-search-ordered(L, n, k)
i = 0;
while((i<n) and (k > L[i])) do
i = i+1;
end while
if ( k = L[i]) then {
{
Print ("key found");
return (i);
}
Else
Print ("key not found");
end Linear-search-ordered.
```

$k = 78$

$i = 0$

while $(0 < 6$ & $78 > 18)$

while $(0 < 6$ & $78 > 18)$ (T)

$i = 0 + 1 = 1$

$6 = n$

Example :-

$L[0:5] = \{18, 23, 56, 78, 90, 100\}$ ([0] [1] [2] [3] [4] [5])

let $k = 78$.

step 1 : 18 23 56 78 90 100

$i = 0$, $0 < 6$ and $78 > L[0]$

$0 < 6$ & $78 > 18$ (T)

$i = i + 1 = 1$

step 2 : 18 23 56 78 90 100

$i = 1$, $1 < 6$ & $78 > L[1]$

$1 < 6$ & $78 > 23$ (T) so, $i = i+1 = 2$

step 3 : 18 23 56 78 90 100

$i = 2$, $2 < 6$ & $78 > L[2]$,

$2 < 6$ & $78 > 56$ (T), so, $i = i+1 = 2+1 = 3$

step 4 : 18 23 56 78 90 100

$i = 3$, $3 < 6$ & $78 > L[3]$,

$3 < 6$ & $78 > 78$ (F), so endwhile goto next if $(78 = L[3]) \Rightarrow 78 = 78$ (T)

"key found".

Thus the element (or) key is found at $i = 3$.

Example :- let

$k = 24$,

step1 : 18 23 56 78 90 100

$i = 0$,
$0 < 6$ and $78 < L[0]$
$0 < 6$ & $24 > 18$ (T)
$i = i + 1 = 1$

step2 : 18 23 56 78 90 100

$i = 1$, $1 < 6$ & $24 > L[1]$
$1 < 6$ & $24 > 23$ (T)
$i = i + 1 = 1 + 1 = 2$

step3 : 18 23 56 78 90 100

$i = 2$, $2 < 6$ & $24 > 56$ (F)

As it is false, go to next condition,

if $(24 = L[2]) = 24 = 56$ (F)

As it is false, go to else block & prints

"key not found".

b) unordered Linear Search :-

- In the unordered linear list of data elements, the key K is found by searching the entire list on comparing the key with each of the element in the list until it finds the required element in the list.

- So, as $L[i] \neq k$, the data elements of a list are checked and compared.

- And if $L[i] = k$ then it denotes search is completed and returns "key found" else displays as "key not found".

Algorithm :-

```
Linear-Search-unordered (L, n, k)
i = 0;
while ((i < n) and (L[i] ≠ k)) do
i = i+1;
end while
if (L[i] = k) then
{
  Print ("key found");
  return (i);
}
else
  print ("key not found");
end Linear-search-unordered.
```

Example :-

$L[0:5] = \{23, 14, 98, 45, 67, 53\}$ with indices [0] [1] [2] [3] [4] [5]; $n=6$

let $k = 98$

step 1 : 23 14 98 45 67 53

$i=0$,
$0<6$ and ~~L[0]~~ $\neq 98$
$0<6$ & $23 \neq 98$ (T) so,
$i = 0+1 = 1$

step 2 : 23 14 98 45 67 53

$i=1$, $1<6$ & $L[1] \neq 98$
$1<6$ & $14 \neq 98$ (T) so, $i = 1+1 = 2$

step 3 : 23 14 98 45 67 53

$i=2$, $2<6$ & $L[2] \neq 98$
$2<6$ & $98 \neq 98$ (F) so, as it is false it goes to next.
if ($L[2] = 98$) then
if ($98 = 98$) (T)
So, "key found".
at $i = 2$

example:-

let,

k=15

step 1 : 23 14 98 45 67 53

i=0,
0<6 & L[0] ≠ 98.
0<6 & 23 ≠ 15 (T) so,
i = 0+1 = 1

step 2 : 23 14 98 45 67 53

i=1, 1<6 & L[1] ≠ 15
1<6 & 14 ≠ 15 (T) so, i = 1+1 = 2

step 3 : 23 14 98 45 67 53

i=2, 2<6 & L[2] ≠ 15
2<6 & 98 ≠ 15 (T) so, i = 2+1 = 3

step 4 :

23 14 98 45 67 53

i=3, 3<6 & L[3] ≠ 15
3<6 & 45 ≠ 15 (T) so, i = 3+1 = 4

step 5 :

23 14 98 45 67 53

i=4, 4<6 & L[4] ≠ 15
4<6 & 67 ≠ 15 (T) so, i = 4+1 = 5

step 6 : [0] [1] [2] [3] [4] [5]
23 14 98 45 67 53

i=5, 5<6 & L[5] ≠ 15
5<6 & 53 ≠ 15 (T) so, i = 5+1 = 6

step 7 :

23 14 98 45 67 53

L[6] = 15

i=6, 6<6 & L[6] ≠ 15
(F)

as this is false this ends while loop & goes to next,

if (L[5] = 15)
53 = 15 (F) then goes to else block,

so, "key not found".

- Binary search :-

- This can be performed on ordered lists.
- This is also known as logarithmic search (or) bisection.
- Binary search searches for a key k in an ordered list L, by dividing search list and in finding median element of list kmid. This is Obtained as,

$$k_{mid} = \left\lceil \frac{(i+j)}{2} \right\rceil$$

where, Comparison of k with k_{mid} is performed as,

$$\begin{cases} k = k_{mid} & ; \text{ then binary search is done} \\ k < k_{mid} & ; \text{ then continue search in sub list } \{k_i, k_{i+1}, \ldots k_{mid-1}\} \\ k > k_{mid} & ; \text{ then continue search in sublist } \{k_{mid+1}, k_{mid+2}, \ldots k_j\} \end{cases}$$

where, i = initial element of list and
j = last | final element of list.

- In simple this binary search adopts the Divide-and-conquer method to solve given problem by dividing into smaller problems instances.

Note:

when the key is searched, it is done on only one of the sublists and hence with each division portion of the list gets as discounted (left over aside without counting).

Algorithm :-

Procedure binary_search(L, low, high, k)

low → first, high → last, k → key

if (low > high) then

{

binarysearch = 0;

Print ("key not found");

Exit();

}

else

{

$mid = \left\lfloor \frac{low + high}{2} \right\rfloor$;

Case :

1) : k = L[mid] :

{

Print ("key found");

binary-search = mid;

return L[mid];

}

2) : k < L[mid] :

binary_search = binary_search(L, low, mid-1, k);

3) : k > L[mid] :

binary-search = binary-search(L, mid+1, high, k);

end case

}

end binary-search

Example :-

$L = \{k_1, k_2 \text{ ---- } k_{15}\}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	21	34	38	45	49	67	69	78	79	82	87	93	97	99

$= \{12, 21, 34, 38, 45, 49, 67, 69, 78, \boxed{79}, 82, 87, 93, 97, 99\}$

① $k = 21$,

	low	high	mid
step1 :-	1	15	$\left(\frac{1+15}{2}\right) = \left(\frac{16}{2}\right) = 8$

i, $k = 21$, $mid = 8$

$21 = L[8]$,

$21 = 69$ (F) so,

ii, $21 < L[8]$

$21 < 69$ (T) so,

(L, low, mid-1, k)

(L, 1, 7, 21)

	low	high	mid
Step2 :-	1	7	$\left(\frac{1+7}{2}\right) = \frac{8}{2} = 4$

i, $k = 21$, $mid = 4$

$21 = L[4]$,

$21 = 38$ (F) so,

ii, $21 < L[4]$

$21 < 38$ (T) so,

(L, 1, 3, 21)

Step 3:- 1 3 $\left(\frac{1+3}{2}\right) = \frac{4}{2} = 2.$

i) k = 21, mid = 2

21 = L[2]

21 = 21 (T) so,

"key found"

and return L[2].

(d) k = 78,

low high mid

Step 1:- 1 15 $\left(\frac{1+15}{2}\right) = \left(\frac{16}{2}\right) = 8$

i) k = 78, mid = 8,

78 = L[8]

78 = 69 (F) so,

ii) 78 < L[8]

78 < 69 (F)

So,

iii) 78 > L[8]

78 > 69 (T)

So,

(L, 9, 15, 78)

low high

Step 2:-

9 15 $\left(\frac{9+15}{2}\right) = \left(\frac{24}{2}\right) = 12$

i) k = 78, mid = 12

78 = L[12]

78 = 87 (F) so,

ii) 78 < L[12]

78 < 87 (T)

So,

(L, 9, 11, 78)

low high key

iii) ~~78 > L[12~~

low - mid-1

Step 3:-

9 11 $\left(\frac{9+11}{2}\right) = \left(\frac{20}{2}\right) = 10$

i) k = 78, mid = 10.

78 = L[10]

78 = 79 (F)

so,

ii) 78 < L[10]

78 < 79 (T) so,

(L, 9, 9, 78)

Step 4 :- 9 9 $\left(\frac{9+9}{2}\right) = \frac{18}{2} = 9$

i) $k = 78$, $mid = 9$

$78 = L[9]$

$78 = 78$ (T) so,

"key found"

return L[9].