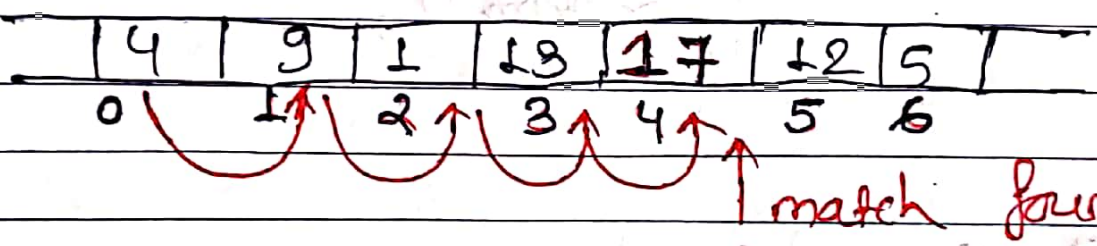
Searching Algorithm

① Linear Search

→ Linear search is a searching algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.



Time complexity is O(n).

// Program for linear search.

```
bool Linear_Search (int Arr[], int Key){
    int i=0;
    while (i<6){
        if (Arr[i]==Key){
            return 1;
        }
        else {
            i=i+1;
        }
    }
    return 0;
}
```

(ii) Binary Search

→ Binary Search is fast searching algorithm. This search algorithm works on divide and conquer principle. For this algorithm data should be in sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the

middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduce to zero.

$O(\log n)$

Arr

3	5	9	26	31	35	42	44
0	1	2	3	4	5	6	7

search = 42

low = 0

high = 7

$mid = \frac{low + high}{2} = \frac{0+7}{2}$

$= 3$

check, Arr[3] == 42 → no,

so,

low = mid + 1 = 3 + 1 = 4

high = 7

$mid = \frac{4+7}{2} = 5$

check, Arr[5] == 42 → no,

low = mid + 1 = 5 + 1 = 6

high = 7

$$mid = \frac{6+7}{2} = 6$$

check, Arr[6] == 42 → yes.

// program for Binary search

```
bool Binary_search (int Arr[], int key, int low,
                                        int high){
    int mid;
    mid = (low + high) / 2;
    if ( low <= high ){
        if ( key == Arr[mid] ){
            return 1;
        }
        else if ( key > Arr[mid] ){
            return Binary_search (Arr, key, mid+1,
                                        high);
        }
        else if ( key < Arr[mid] ){
            return Binary_search (Arr, key, low, mid-1);
        }
    }
    return 0;
}
```

O(log n)