Linear Search

```
for(int i=0; i < length(a); i++){
    if(a[i]== key) return i;
}
return -1;
```

Search 33

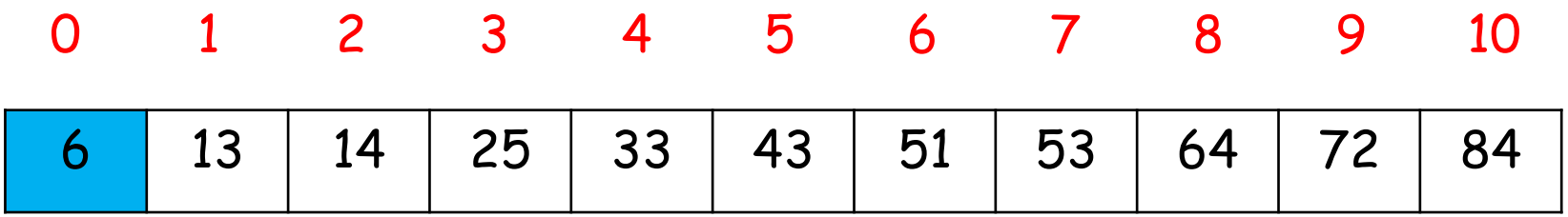
Search for 33 in the following array

6	13	14	25	33	43	51	53	64	72	84

Search 33

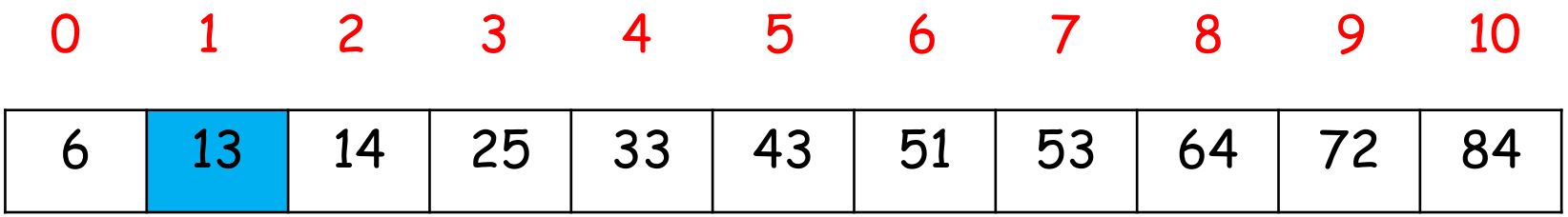
0	1	2	3	4	5	6	7	8	9	10
6	13	14	25	33	43	51	53	64	72	84

Search 33



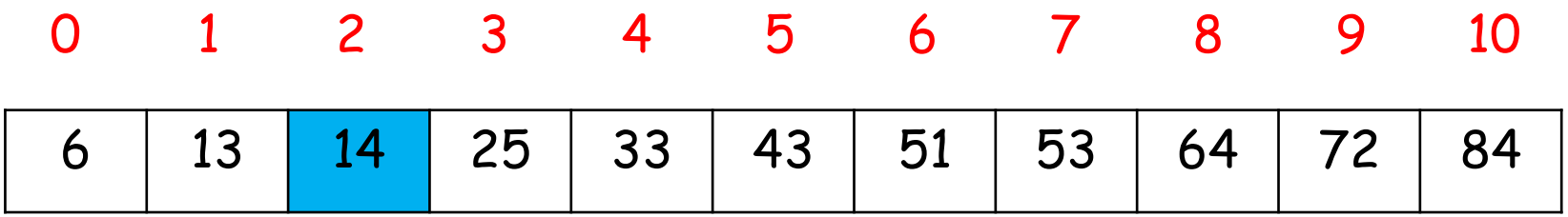
33

Search 33



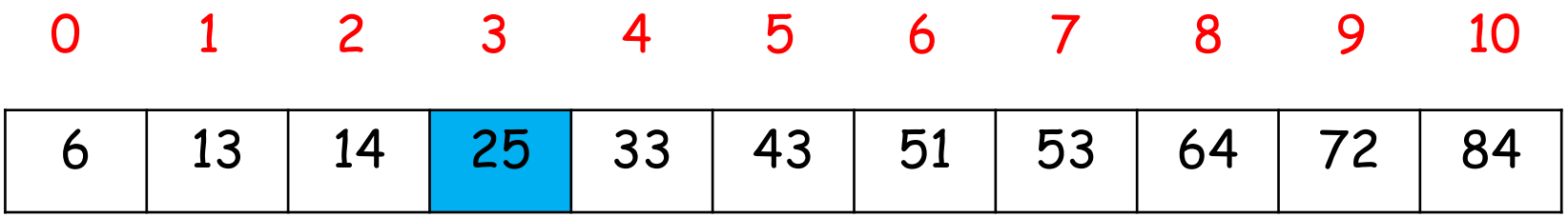
33

Search 33



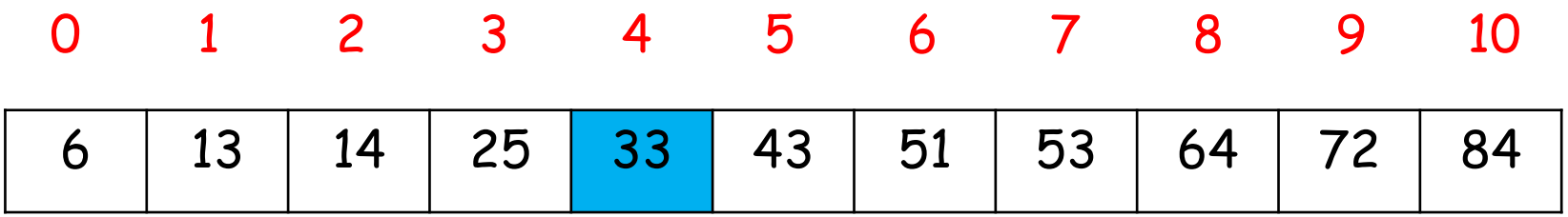
33

Search 33



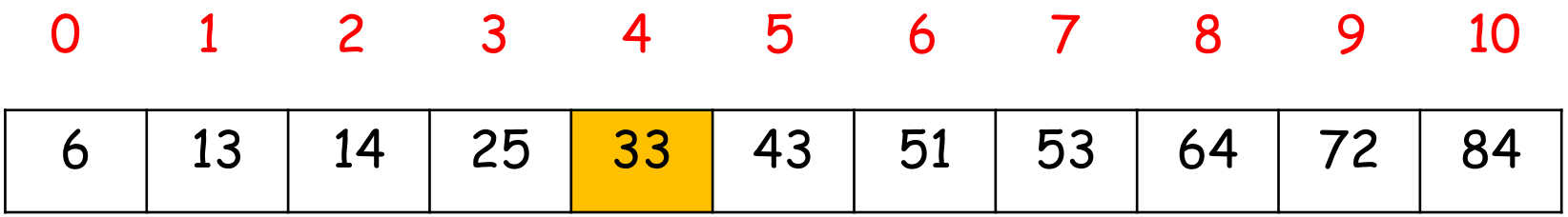
33

Search 33



33

Search 33



33

Binary Search

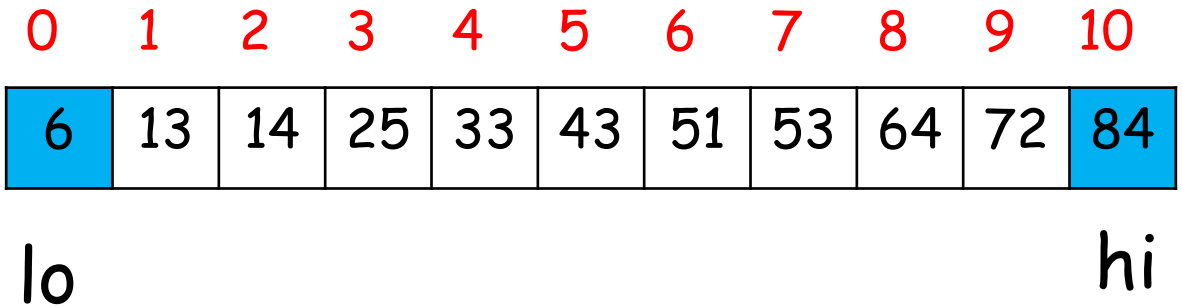
```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi - lo) / 2;
    if(key < a[mid]) hi = mid - 1;
    else if (key > a[mid]) lo = mid + 1;
    else return mid;
}
```

Search 33

Search for 33 in the following array

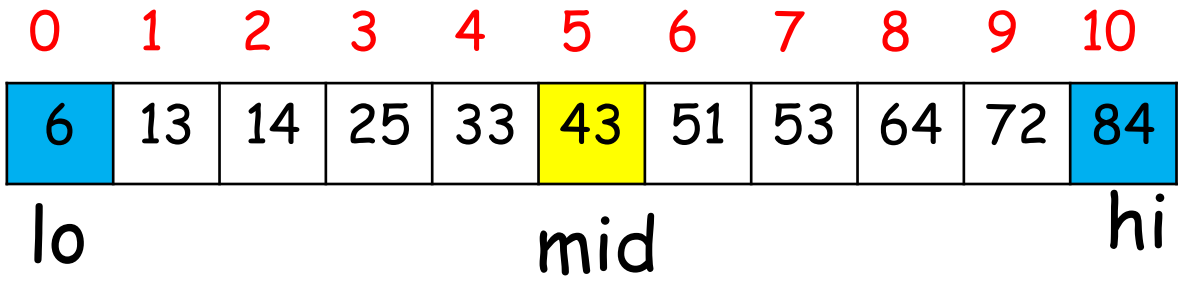
6	13	14	25	33	43	51	53	64	72	84

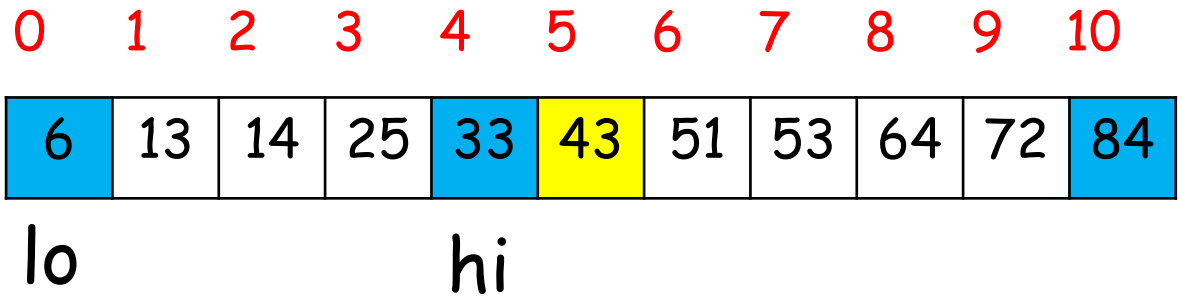
Search 33

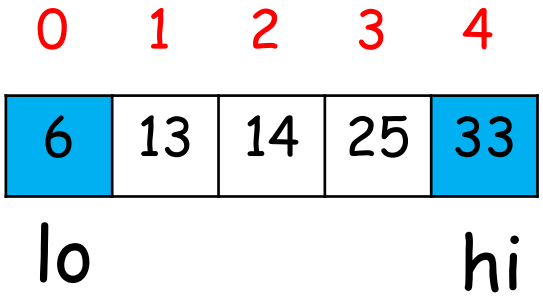



```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Search 33:1st iteration

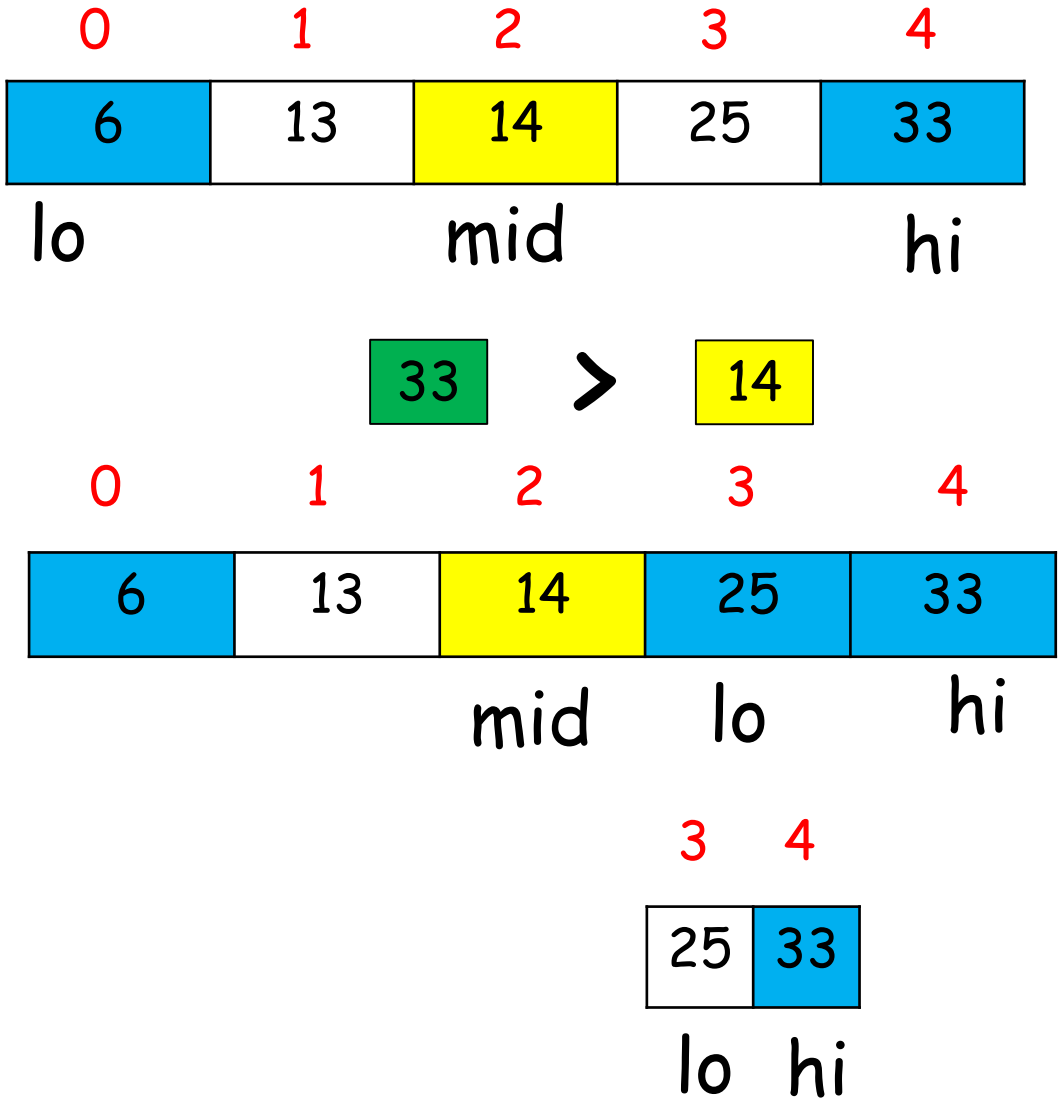






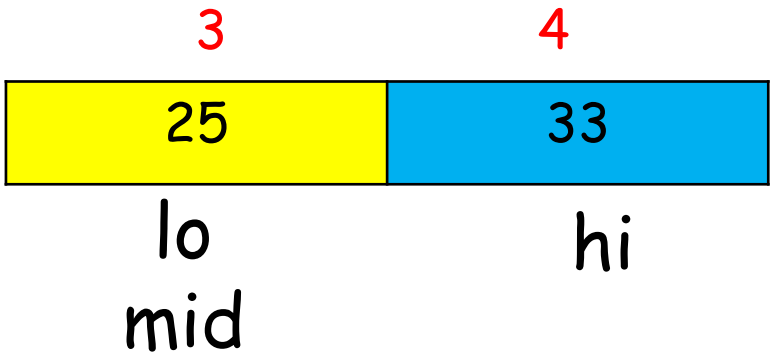

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
     int mid = lo + (hi -
          lo) / 2;
     if(key < a[mid])
          hi = mid - 1;
     else if (key > a[mid])
          lo = mid + 1;
     else
          return mid;
}
```

Search 33:2nd iteration

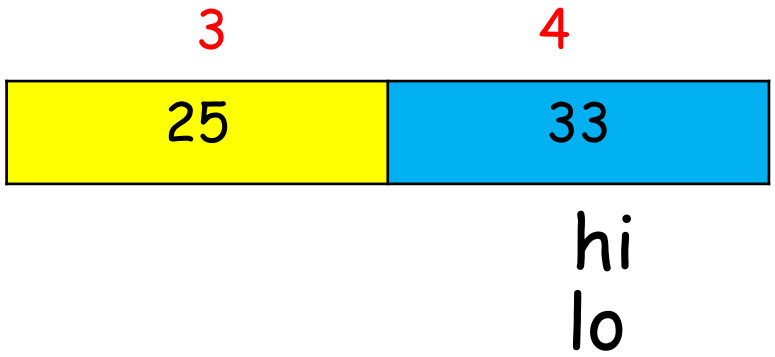


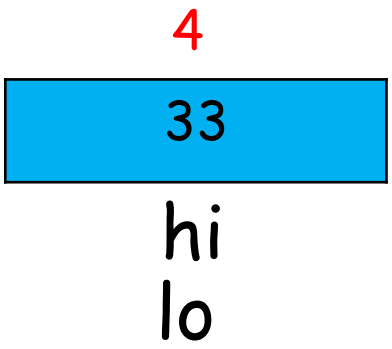

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Search 33:3rd iteration



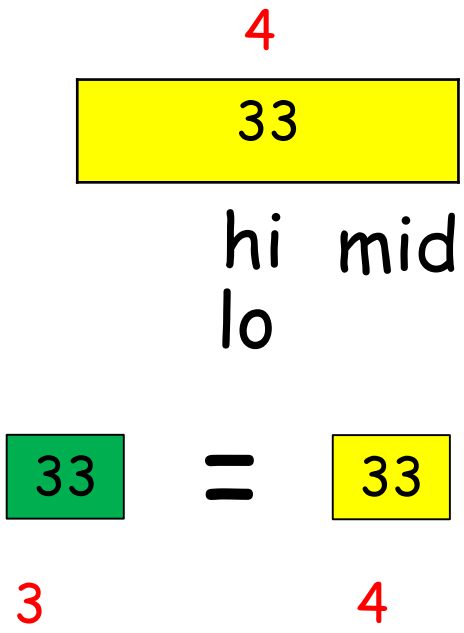
33 > 25






```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

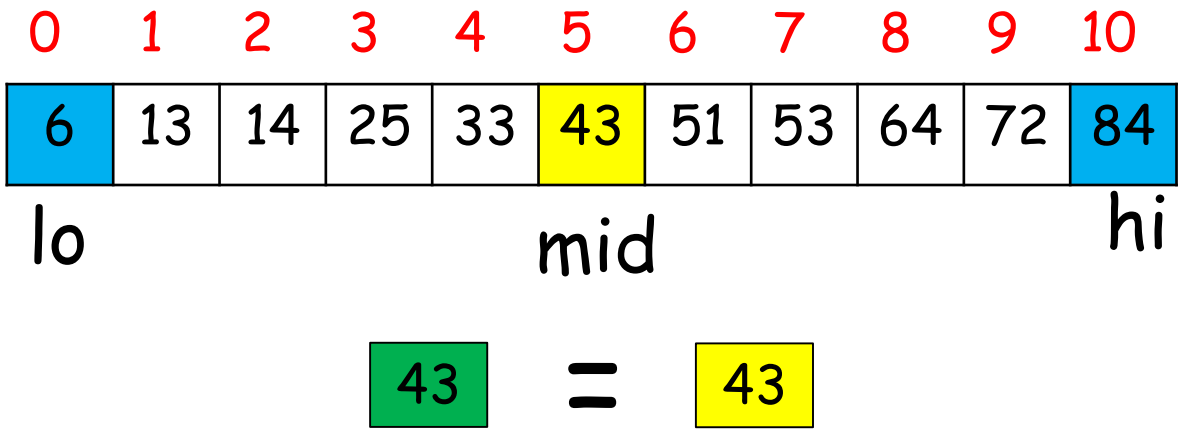
Search 33:4th iteration




```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Output: 4

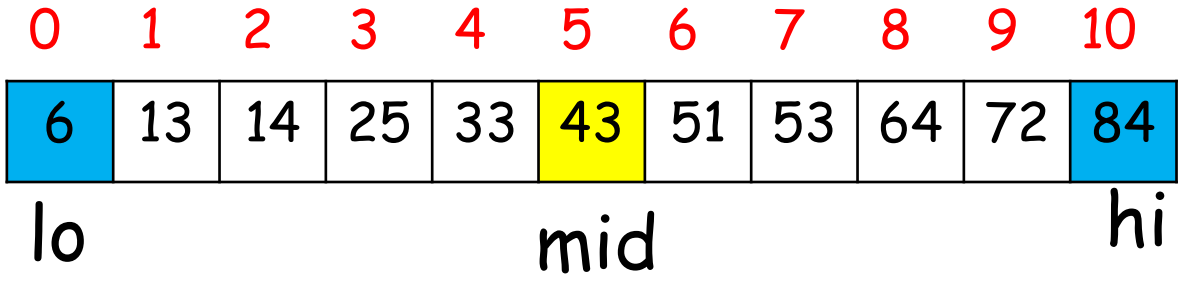
Search 43:1st iteration

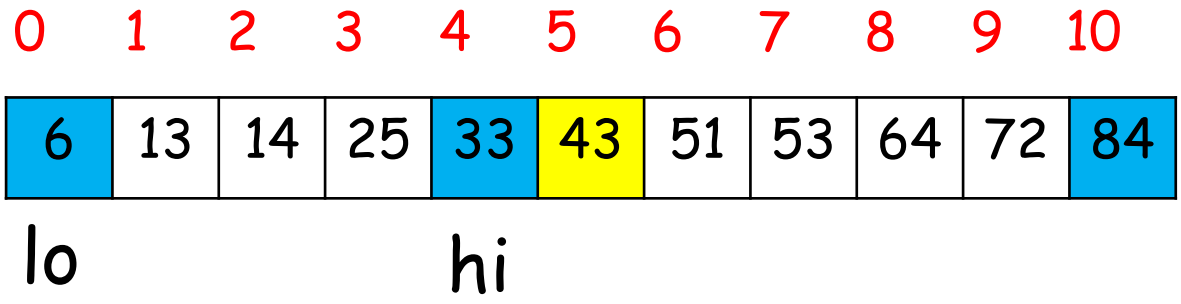


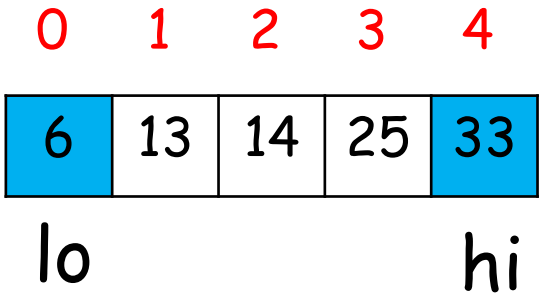

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Output = 5

Search 14:1st iteration

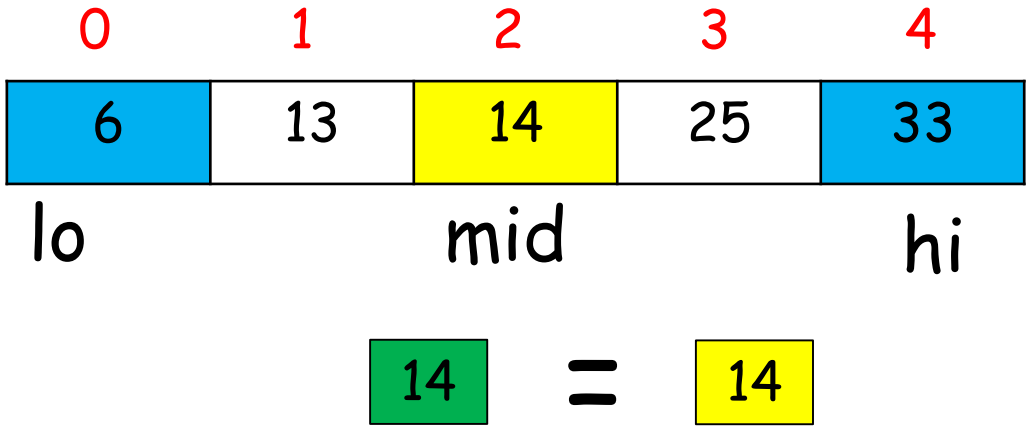







```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
     int mid = lo + (hi -
          lo) / 2;
     if(key < a[mid])
          hi = mid - 1;
     else if (key > a[mid])
          lo = mid + 1;
     else
          return mid;
}
```

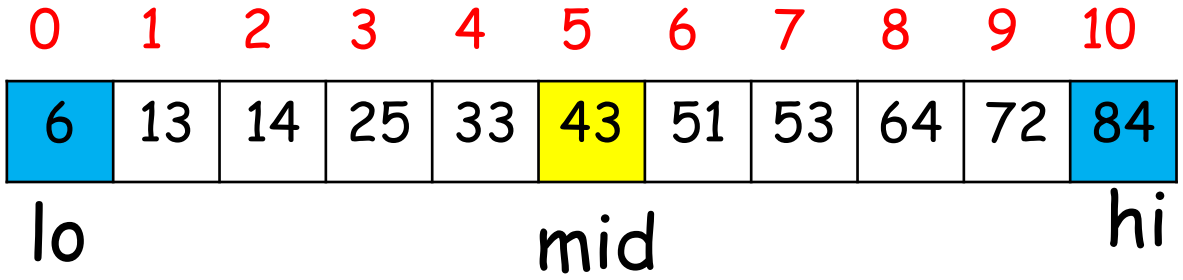
Search 14:2nd iteration



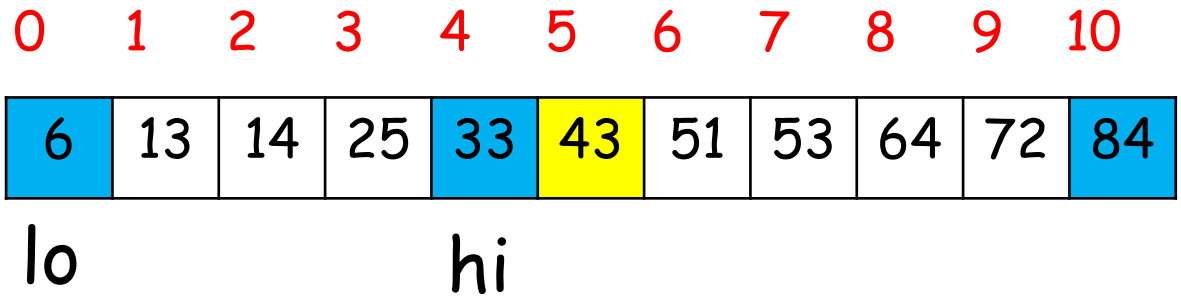

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

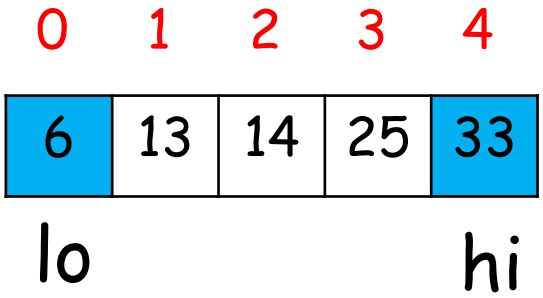
Output = 2

Search 34:1st iteration



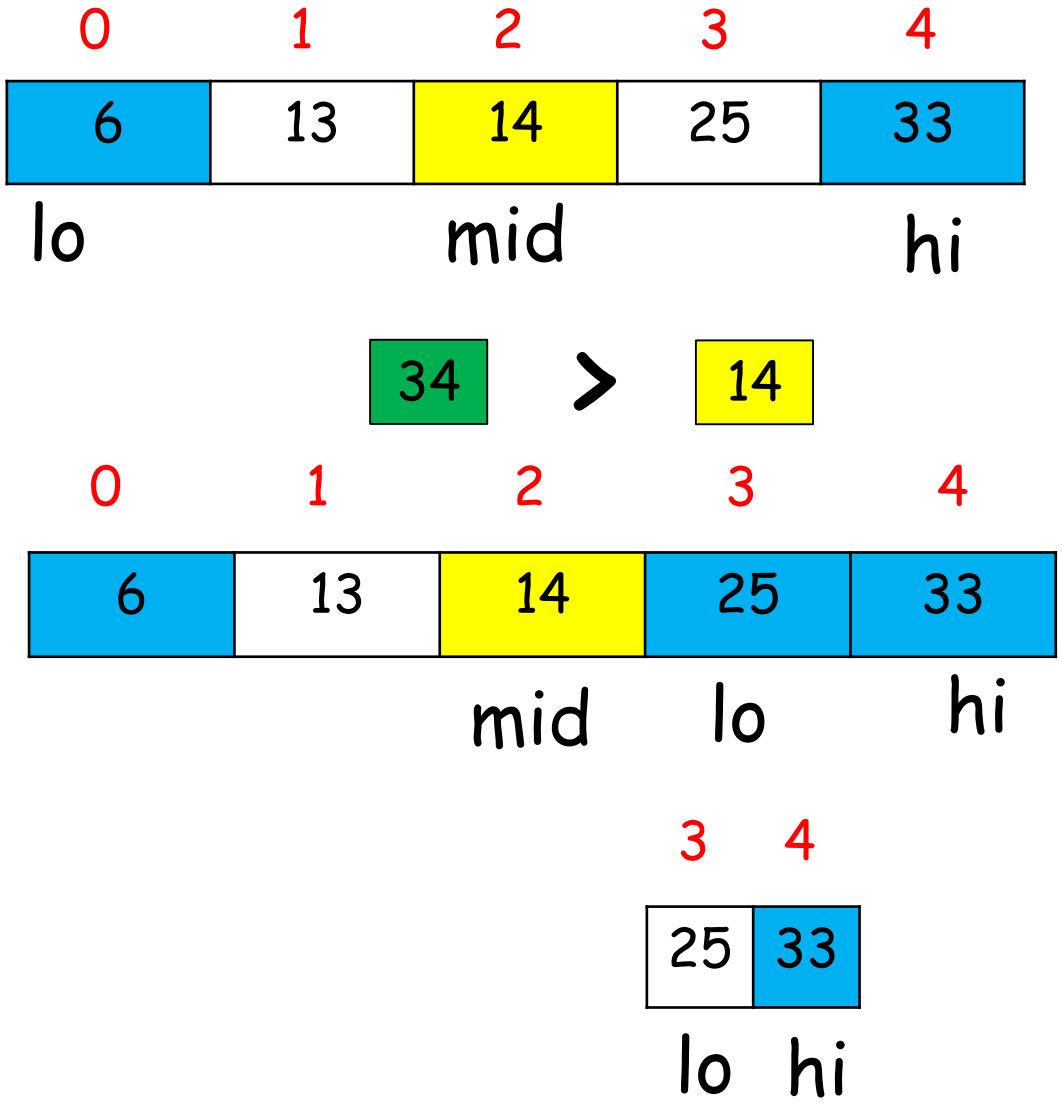
34 < 43





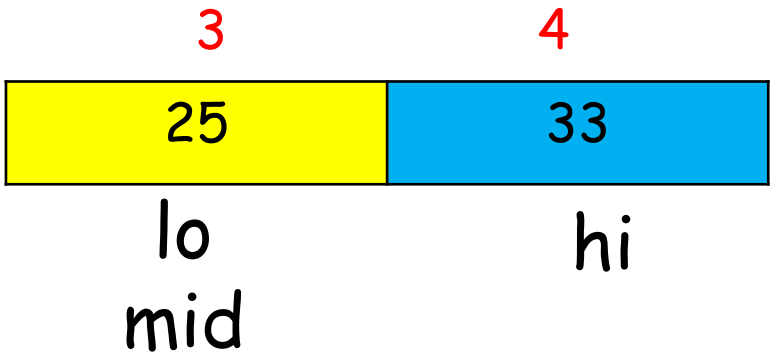

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
     int mid = lo + (hi -
          lo) / 2;
     if(key < a[mid])
          hi = mid - 1;
     else if (key > a[mid])
          lo = mid + 1;
     else
          return mid;
}
```

Search 33:2nd iteration

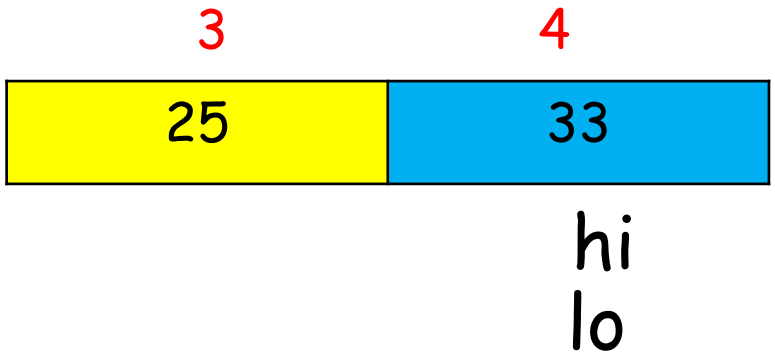


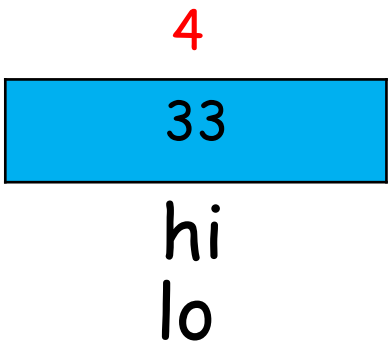

```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Search 33:3rd iteration



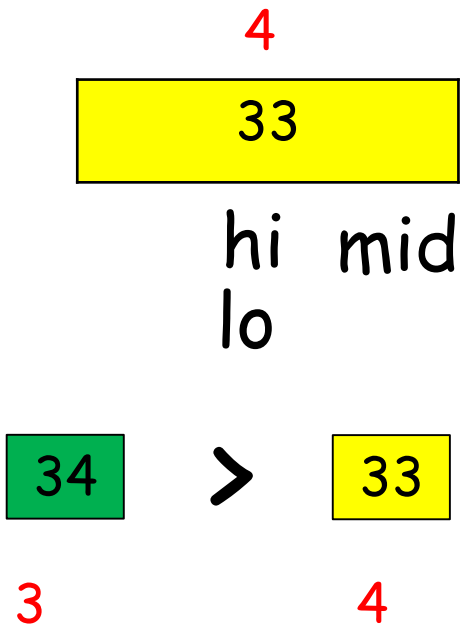
34 > 25






```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Search 33:4th iteration




```
int lo = 0;
int hi = array_size - 1;
while (lo <= hi) {
    int mid = lo + (hi -
          lo) / 2;
    if(key < a[mid])
          hi = mid - 1;
    else if (key > a[mid])
          lo = mid + 1;
    else
          return mid;
}
```

Finding time complexity of Linear Search

```
public int linear search(int[] a, int key) {
        for(int i=0; i < a.length; i++){
                if(a[i]== key) return i;
        }
        return -1;
}
```

Approximately 3 operations per iteration
$f(n) = 3n + 2$

Finding time complexity of Linear Search: Average case

$f(n) = 3n + 2$

n = no of iterations/ location of key

$$f(1) = 3.1 + 2$$
$$f(2) = 3.2 + 2$$
$$f(3) = 3.3 + 2$$
$$\vdots$$
$$f(n) = 3.n + 2$$

$$f(1)+ f(2)+... + f(n) = 3.(1+2+... + n) + (2+ 2+ ... + 2)$$

$$= 3\, n(n+1)\, /2 + 2.n$$

$$= \tfrac{1}{2}\,(3n^2 + 7n)$$

No of cases = n

Average case time complexity = $\tfrac{1}{2}\,(3n^2 + 7n)\, /n$

$$= \tfrac{1}{2}\,(3n + 7)$$

Big O notation: $O(n)$

Finding time complexity of Binary Search: Average case

```
public int binary search(int[] a, int key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            // Key is in a[lo..hi] or not present.
            int mid = lo + (hi - lo) / 2;
            if      (key < a[mid]) hi = mid - 1;
            else if (key > a[mid]) lo = mid + 1;
            else return mid;
        }
        return -1;
    }
```

Approximately 3 comparisons per iteration

Finding time complexity of Binary Search: Average case

List size $n = 2^k$

$k = \log_2 n$

List size	No of comparisons
2^k	= 3
2^{k-1}	= 3
2^{k-2}	= 3
⋮	
⋮	
⋮	
1	= 3

$= (3 + 3 + \ldots + 3)$

$= 3(k + 1)$

$= 3(\log_2 n + 1)$

Big O notation: $O(\log_2 n)$