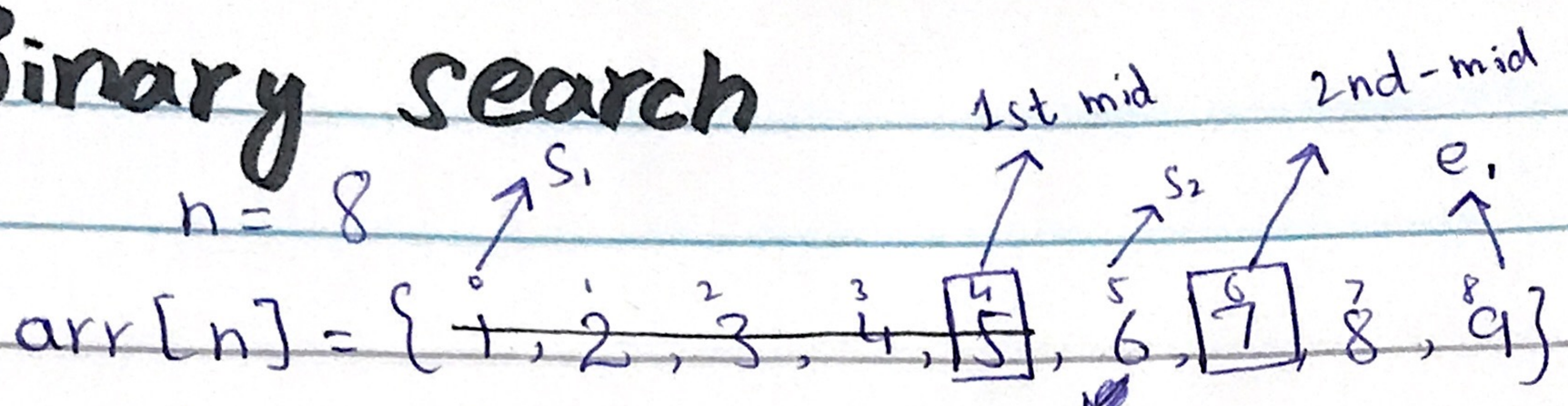
Binary search




```
int s=0 , e=n;
While(s<=e){
    mid = (ste)/2;
If (arr[mid] ==key){
        return mid
else if (arr[mid] > key)
        e = mid-1;'
else
    s= mid + 1; }
```

Output

While (0 <=8)

mid = 8/2 = (4) → 1st-mid

else // arr[mid] < key

s = 4+1 = 5 → Now starting index is at 5.

While (5 <=8)

mid = (5+8)/2 = 13/2 = (6) → 2nd

If (arr[6] == 7)

7 == 7 ✓

return mid; // index of key element in array is returned.

Time complexity of Binary Search

After 1st itera, Length of searching array → n

After 2nd itera, → n/2

After 3rd itera, → (n/2)/2 = n/2² = n/4

After k itera, → n/2^k

Let the Length of array become 1 after k iteration:

$$n/2^k = 1$$

$$n = 2^k$$

$$\log_2(n) = \log_2(2^k)$$

$$\log_2(n) = k \log_2 2 \rightarrow = 1$$

$$\Rightarrow \log_2^n = k$$

Time complexity:

$O(\log_2^n)$ → Binary search

$O(n)$ → Linear search complexity