

Searching and Sorting

Friday Four Square!
Outside Gates, 4:15PM

An Interesting Lecture Excerpt

Admiral Grace Hopper on nanoseconds:

<http://www.youtube.com/watch?v=JEpsKnWZrJ8>

Algorithms Revisited

- An **algorithm** is a procedure for effecting some result.
- There can be many different algorithms for solving the same problem.
- How can we compare algorithms against one another?
- What's the best algorithm for solving a given problem?

Two Famous Problems

- **Searching**

- Given an array of values, determine whether some value is contained in that array.
- Very important: finding medical records, determining if a bookstore has a copy of a book, etc.

- **Sorting**

- Given an array of values, rearrange those values to put them in sorted order.
- Enormously important: shows up in iTunes, Google, Facebook, etc.

Searching

Can I get some volunteers?

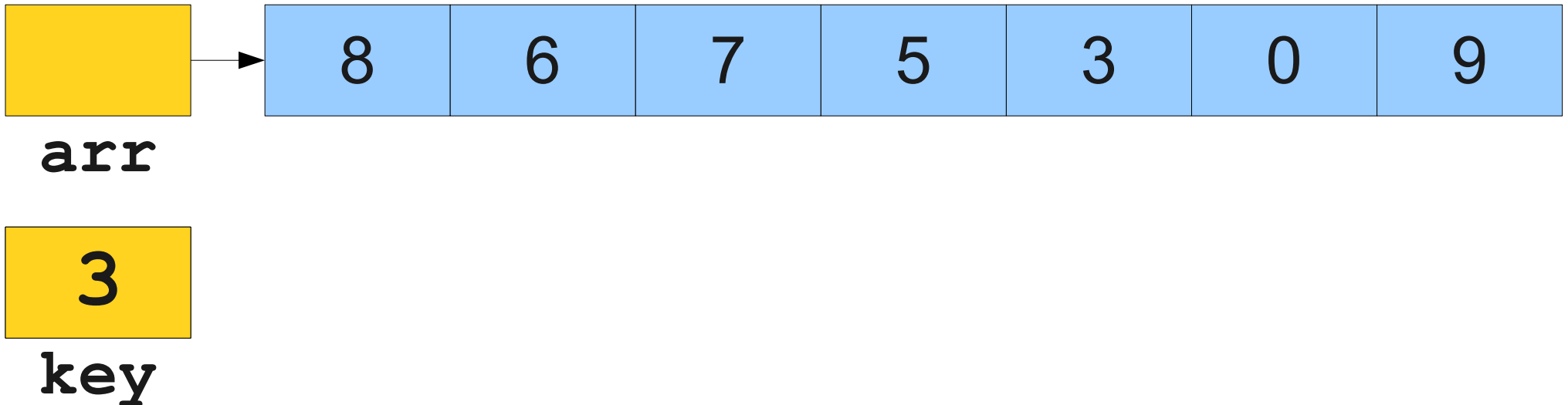
Linear Search

Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
  
    return -1;  
}
```

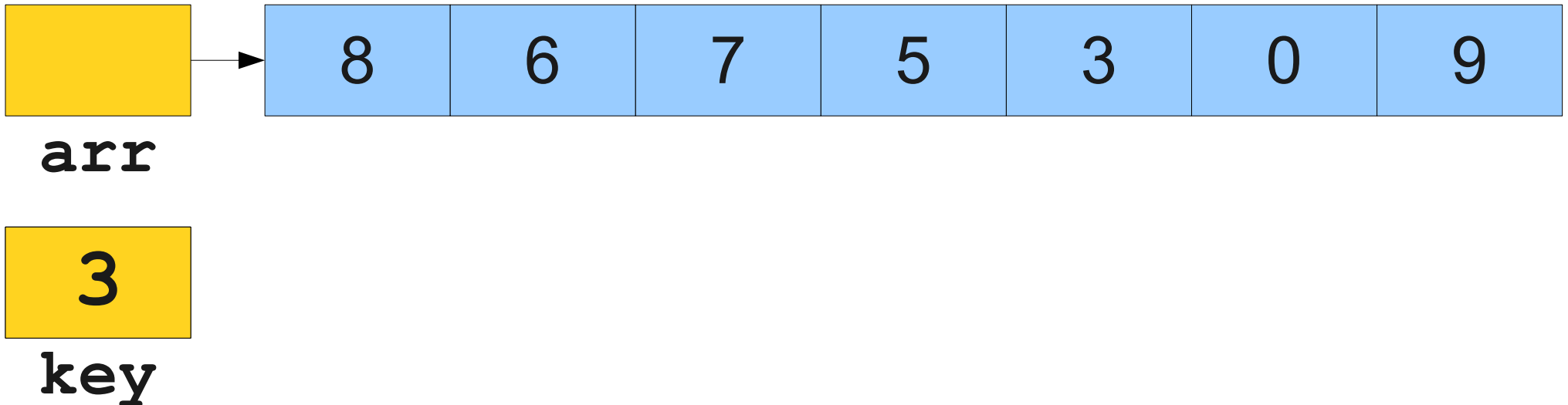
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
  
    return -1;  
}
```



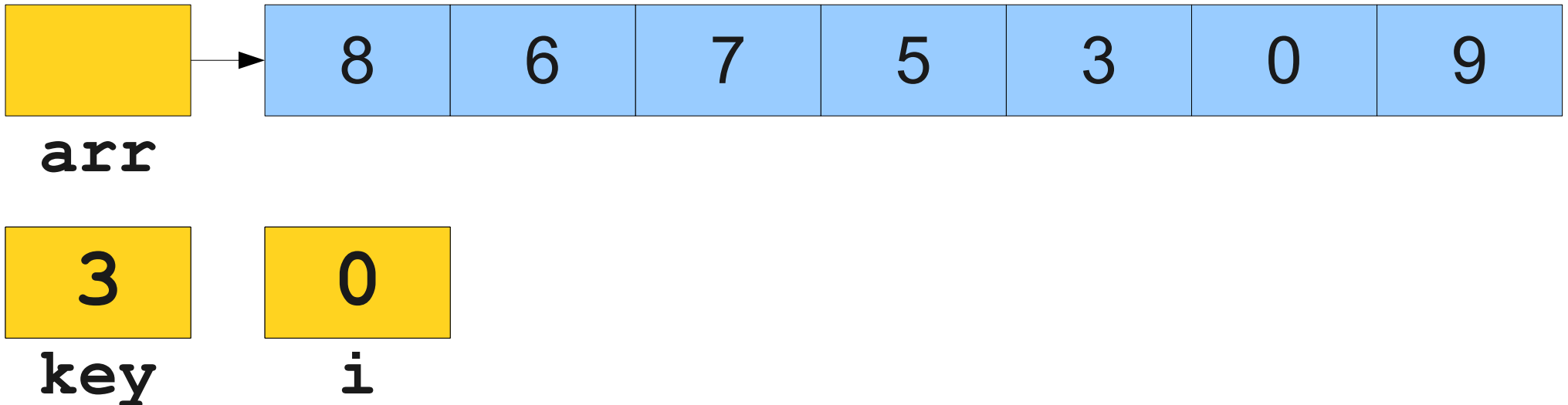
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



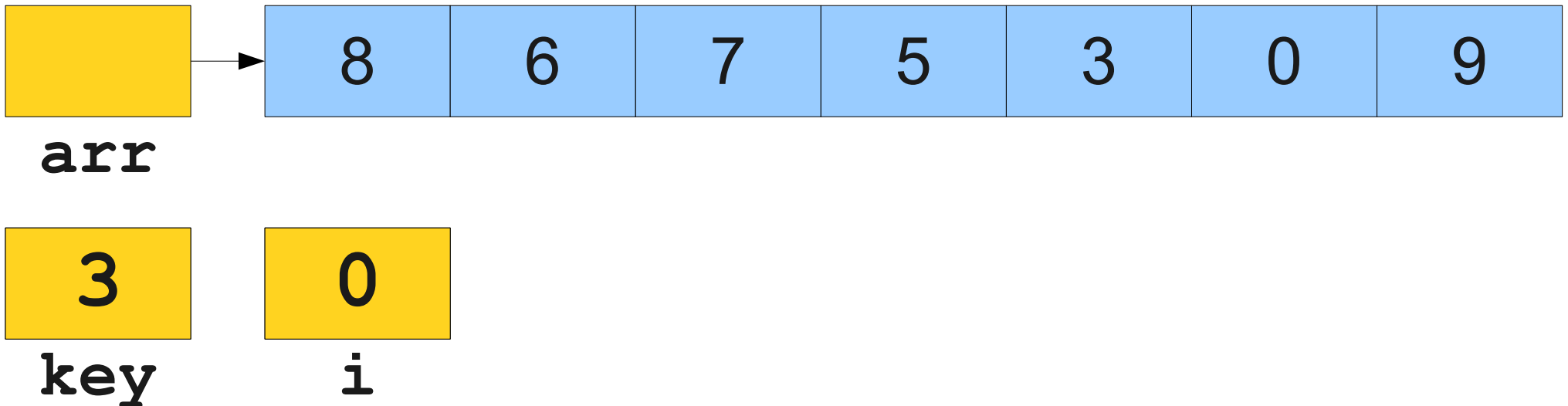
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



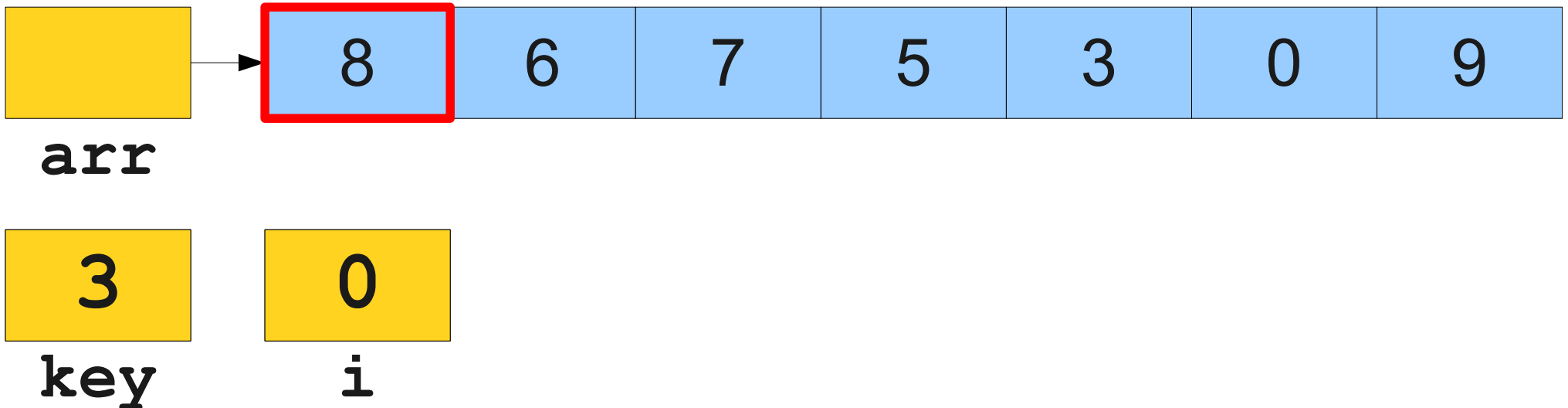
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



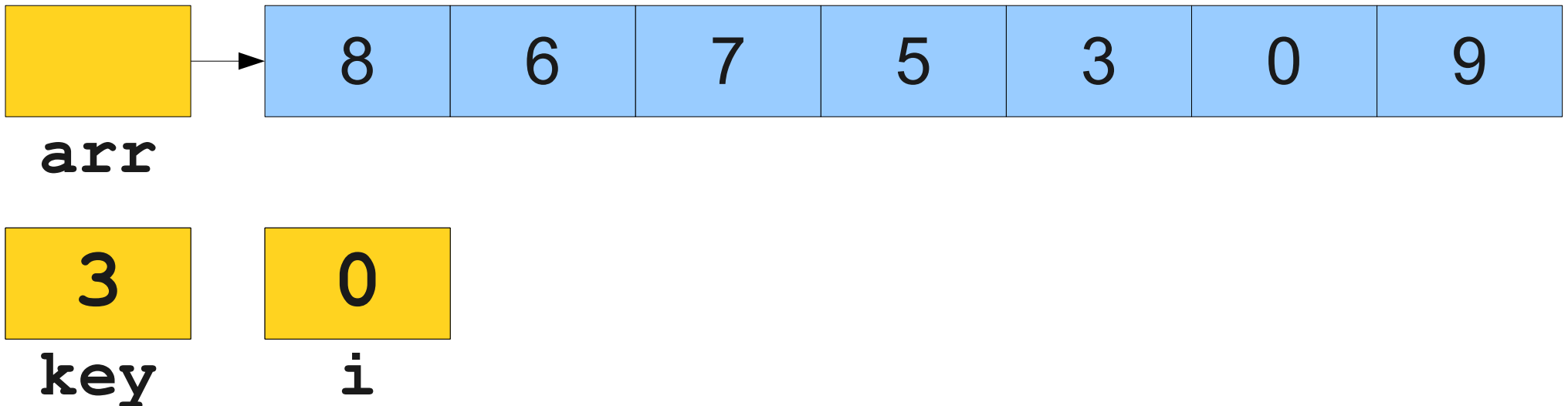
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



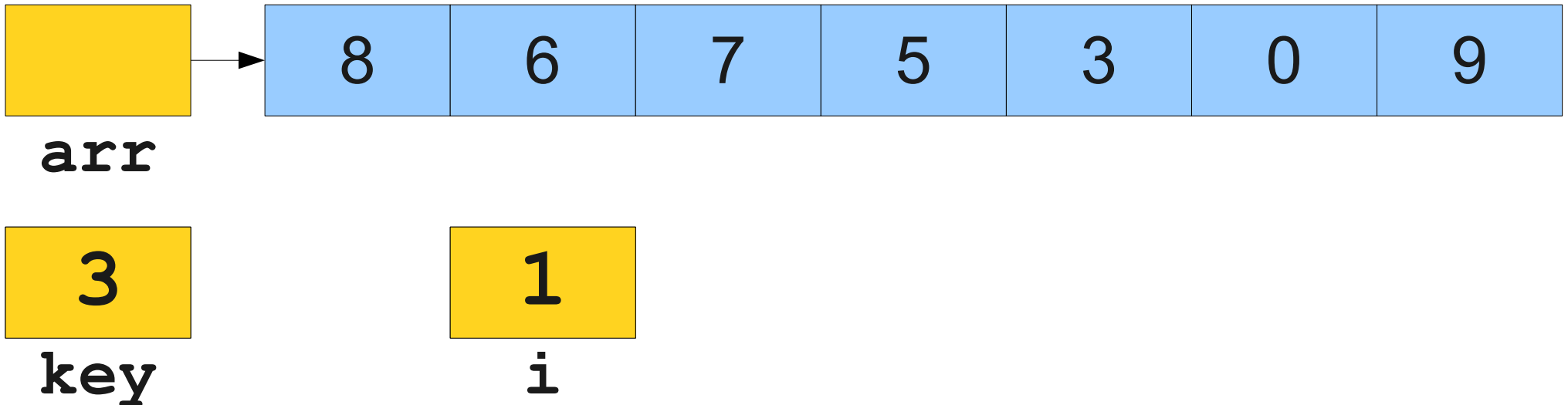
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



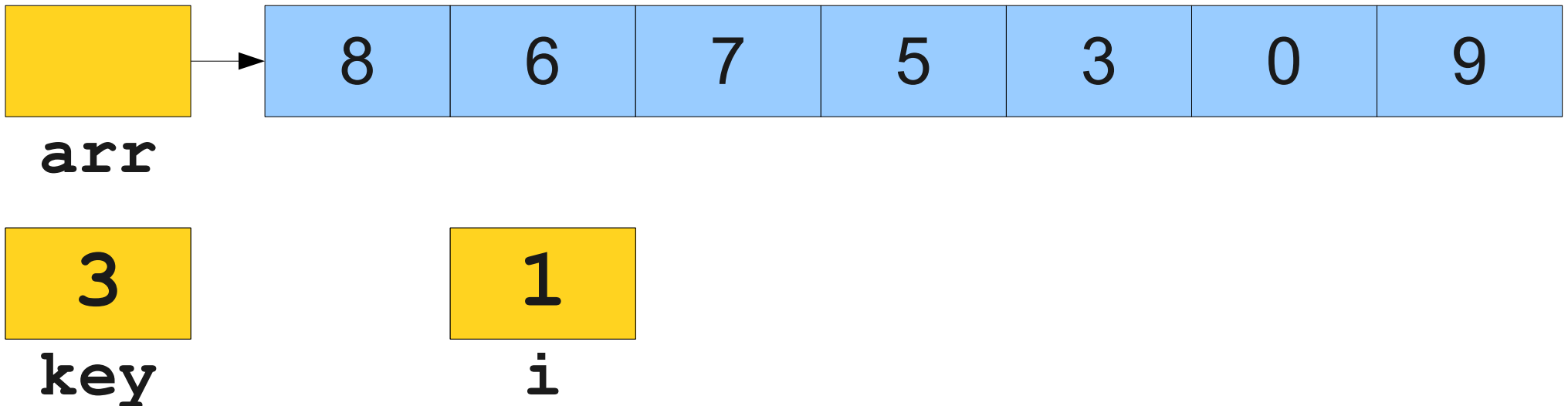
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



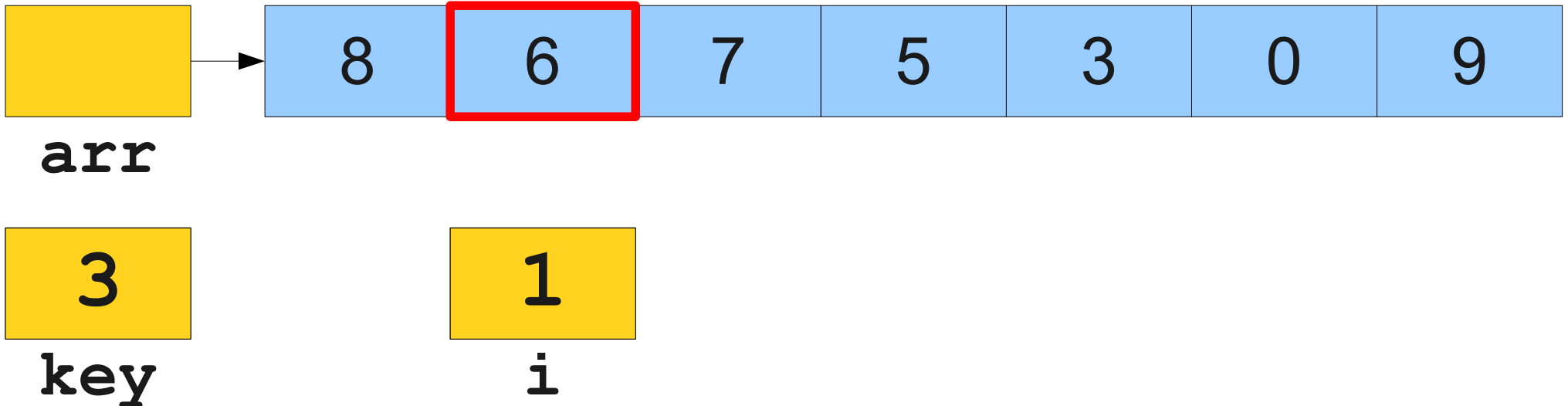
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



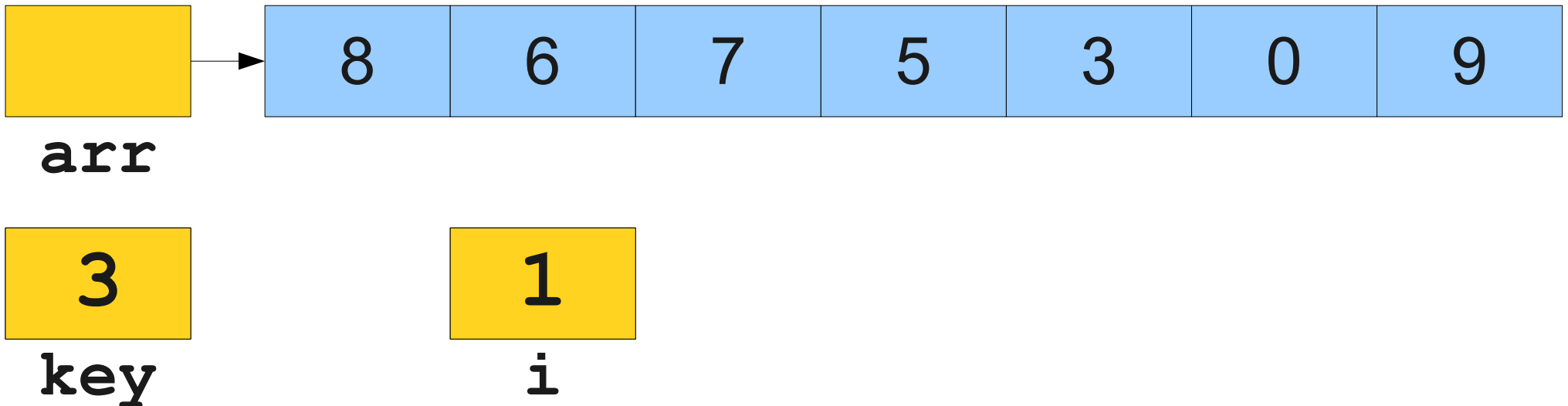
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



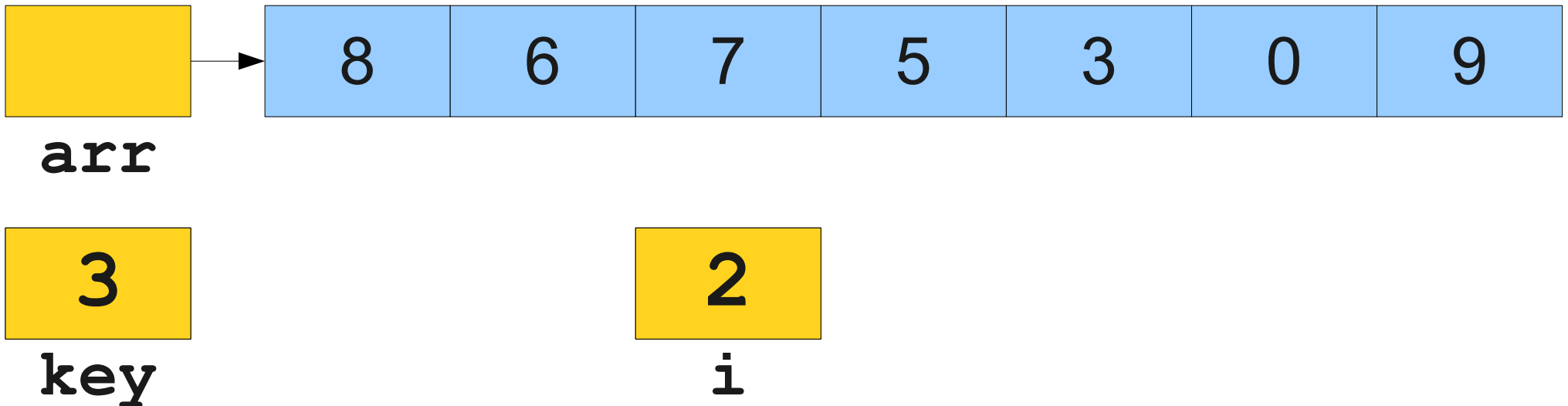
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



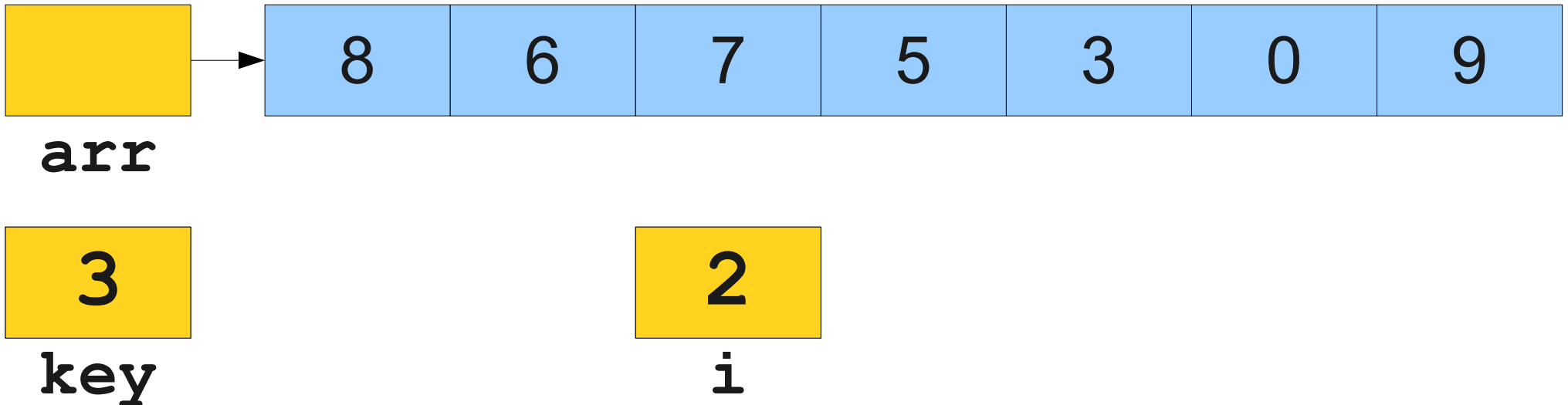
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



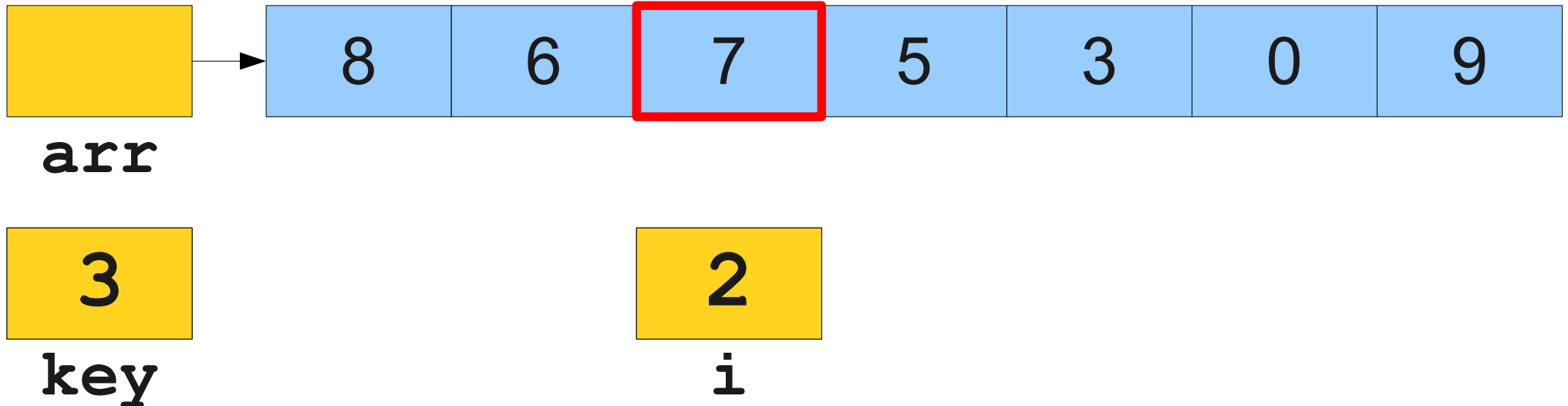
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



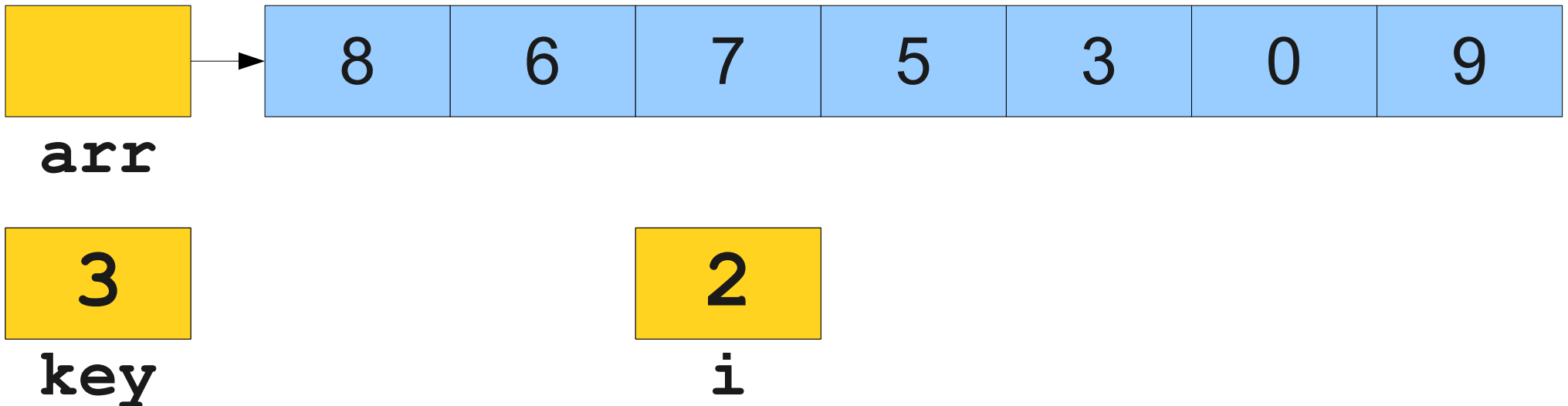
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



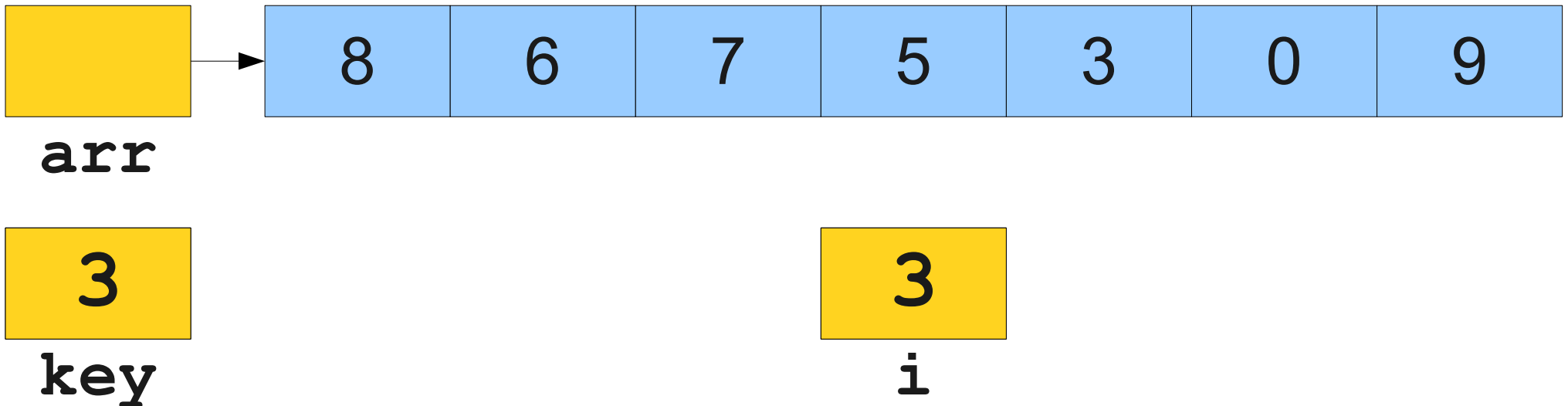
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



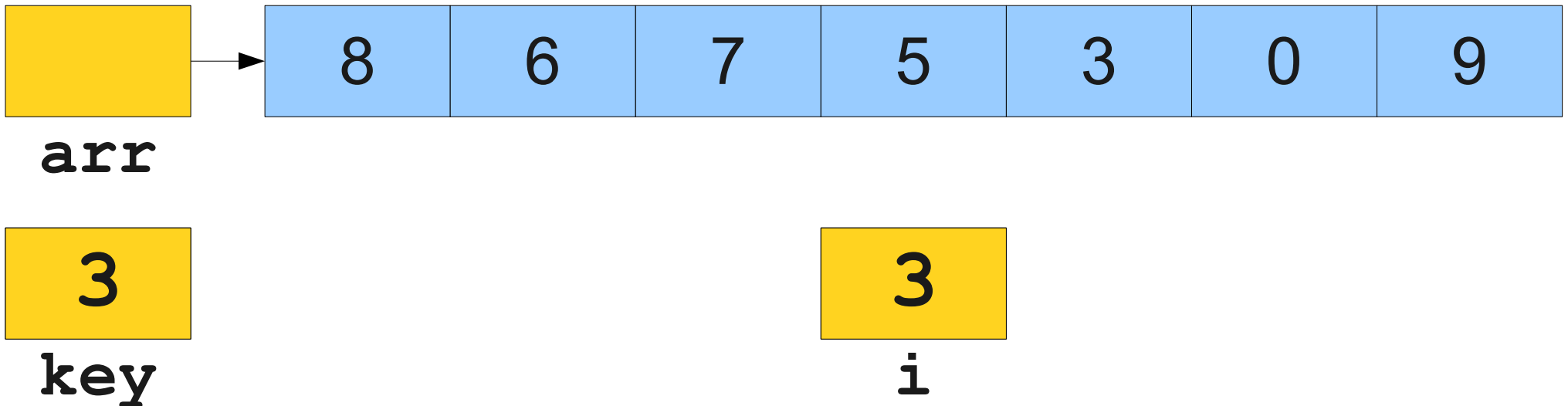
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



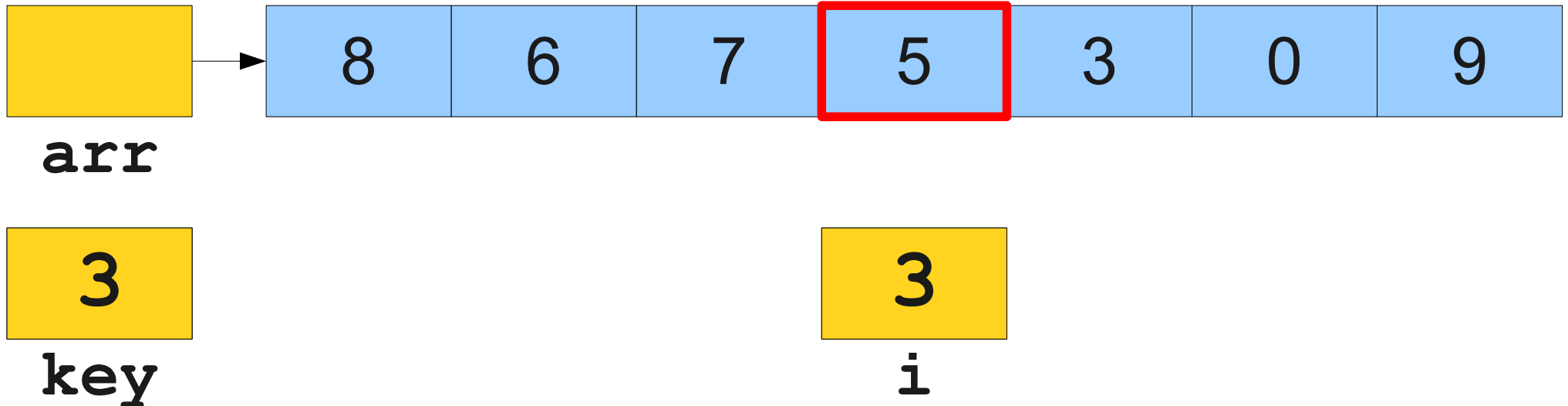
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



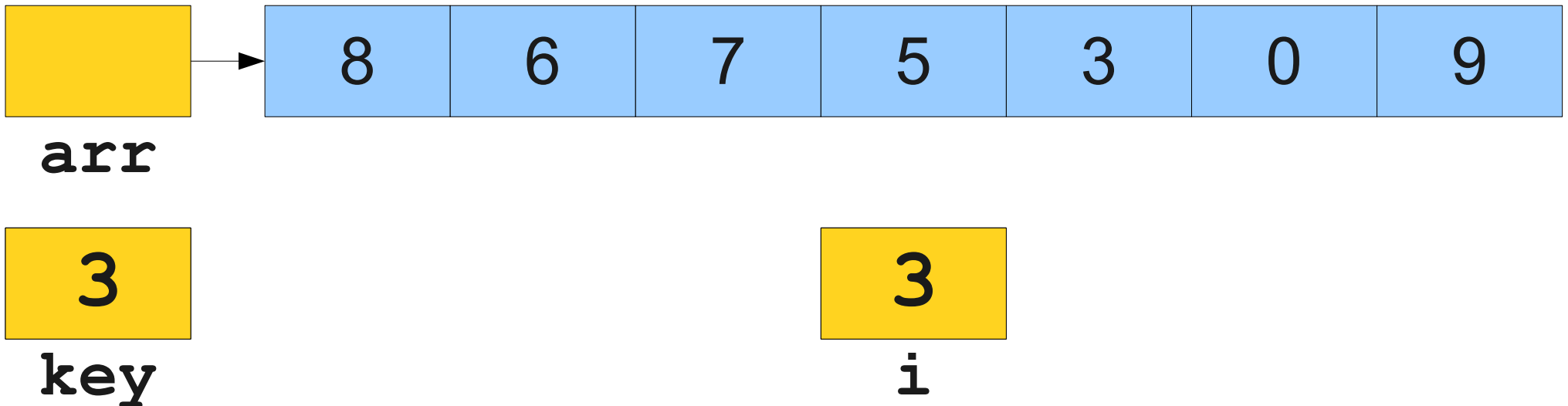
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



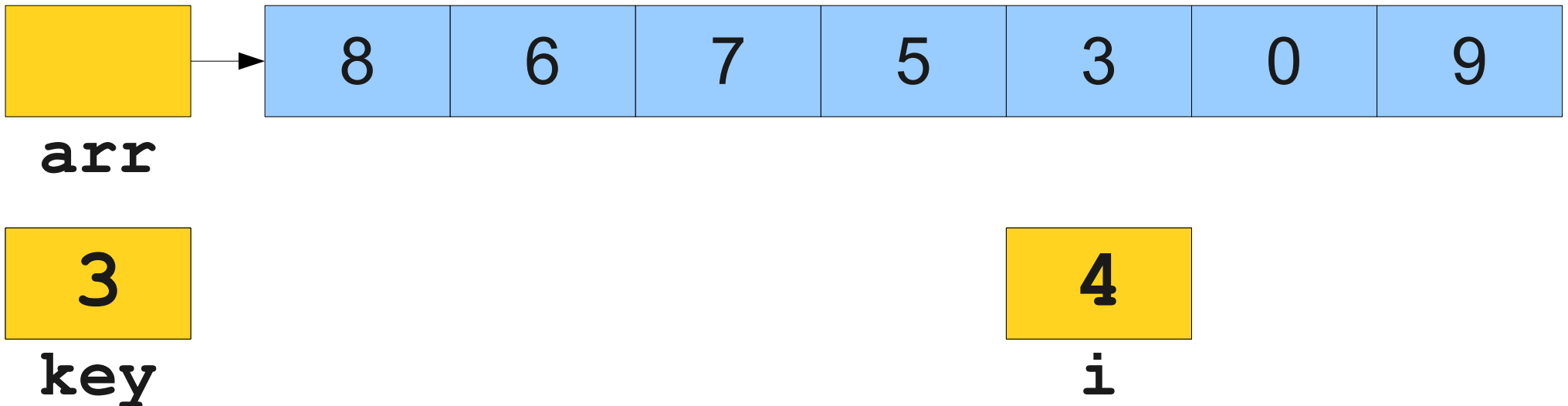
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



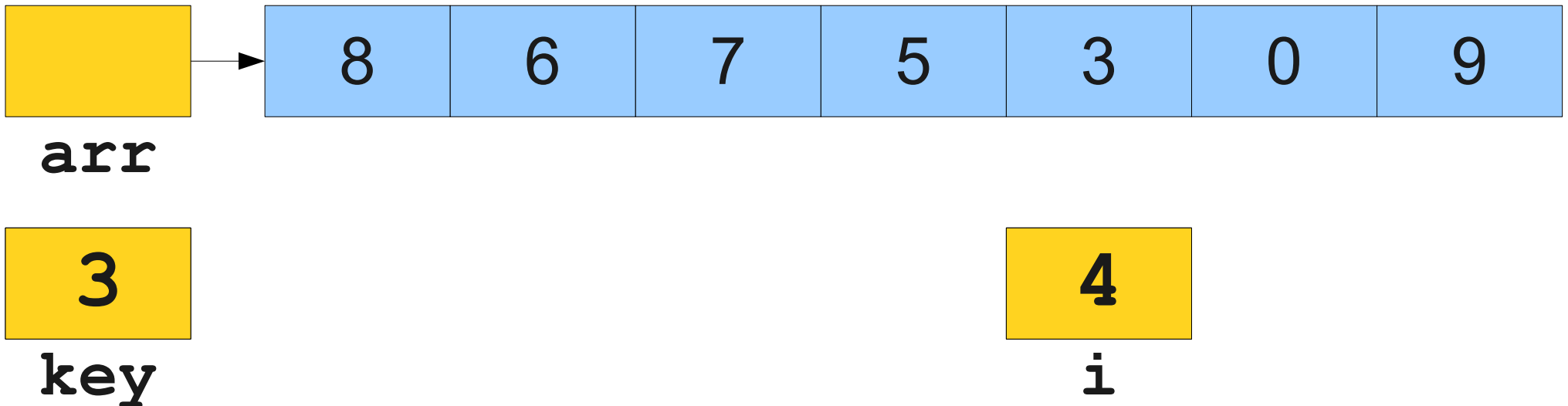
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



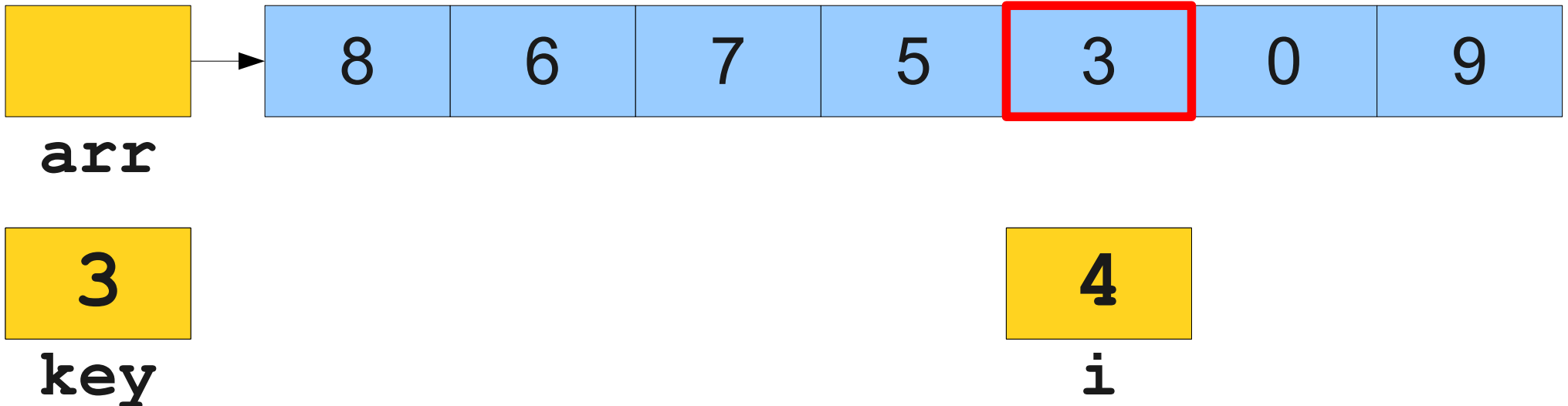
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



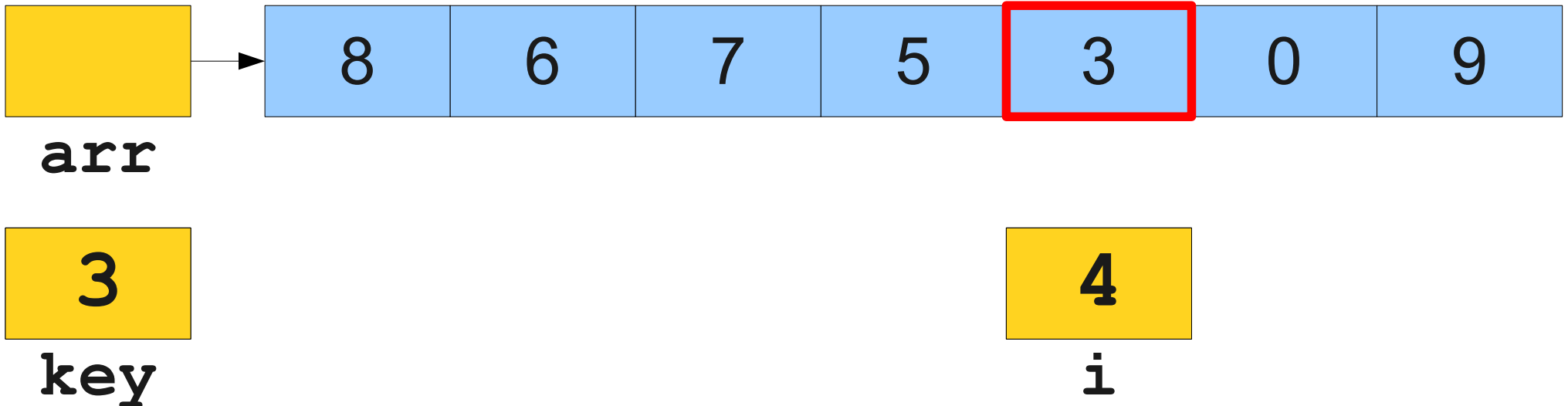
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



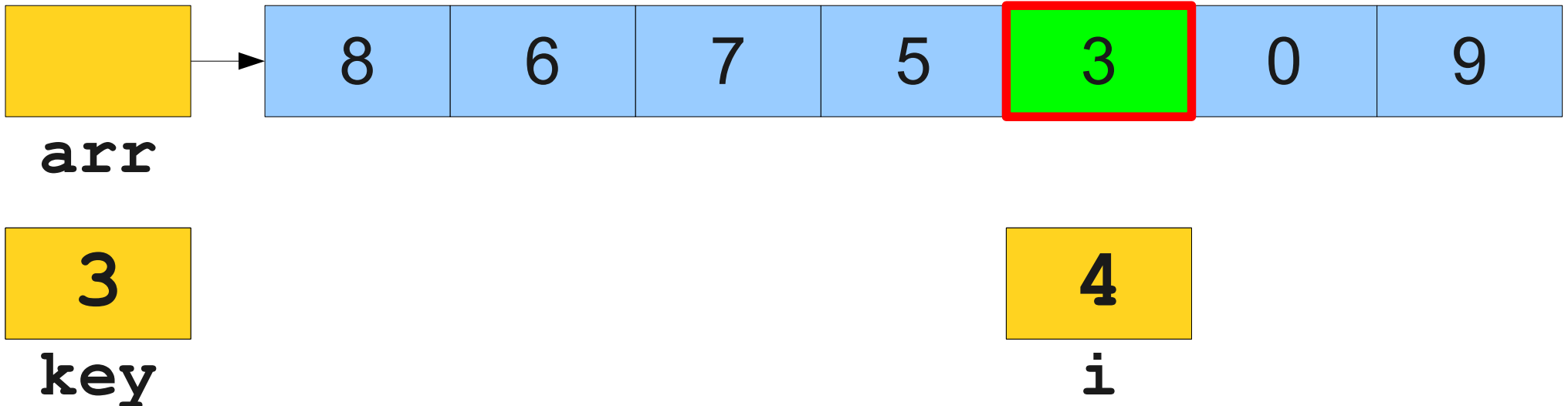
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



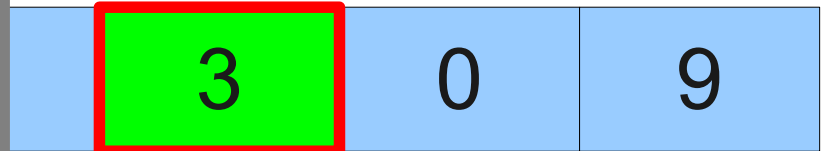
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
}
```



key

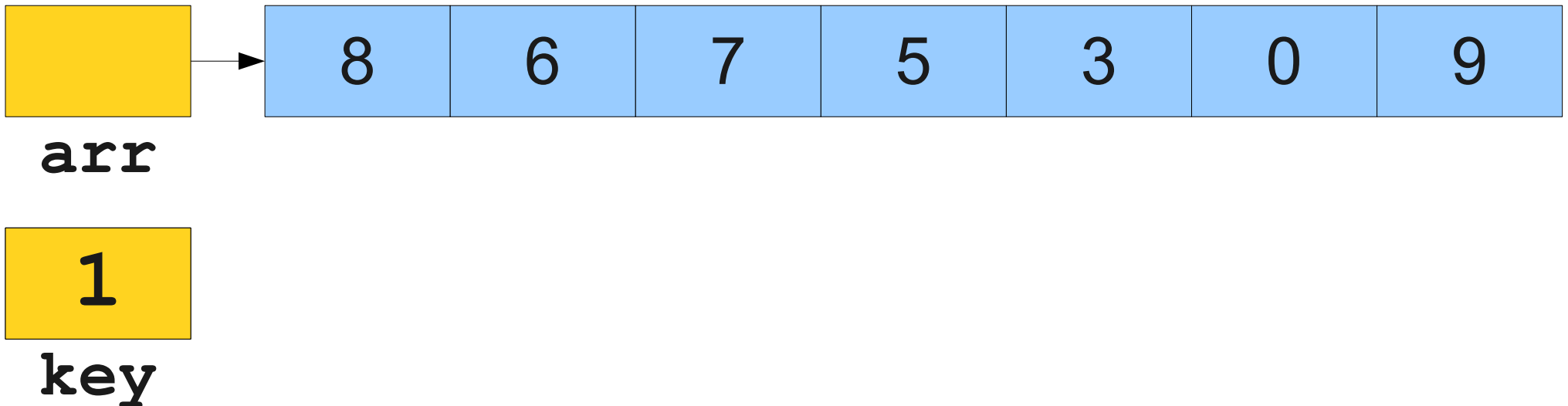
i

Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
  
    return -1;  
}
```

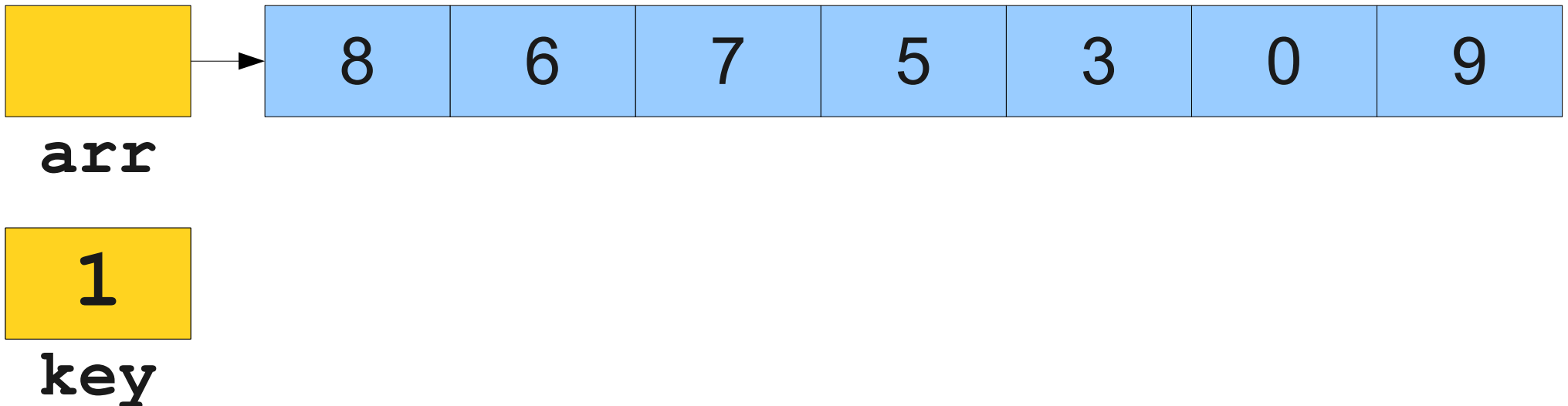
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
  
    return -1;  
}
```



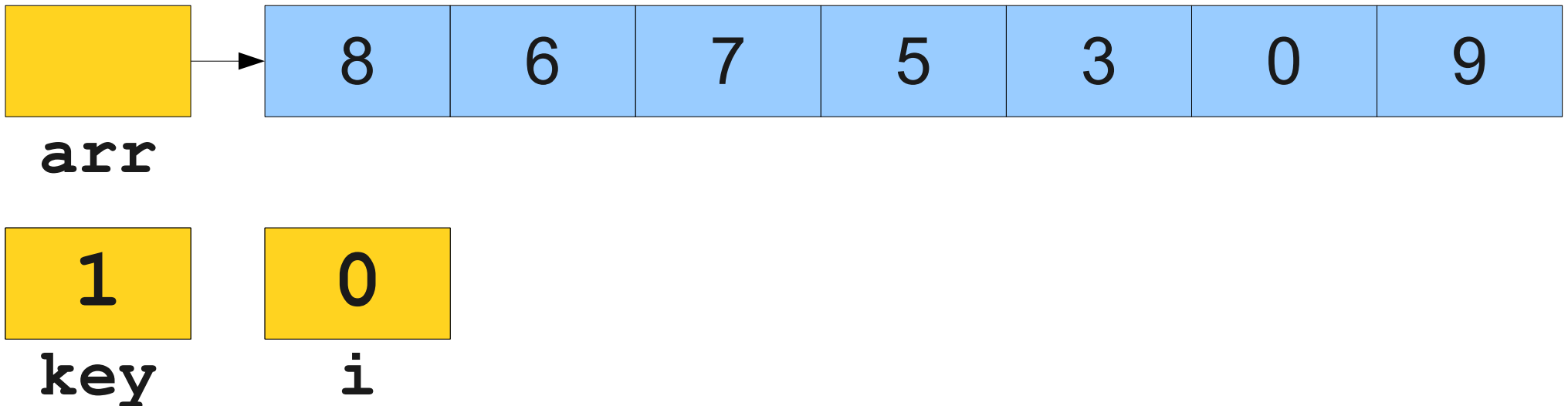
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



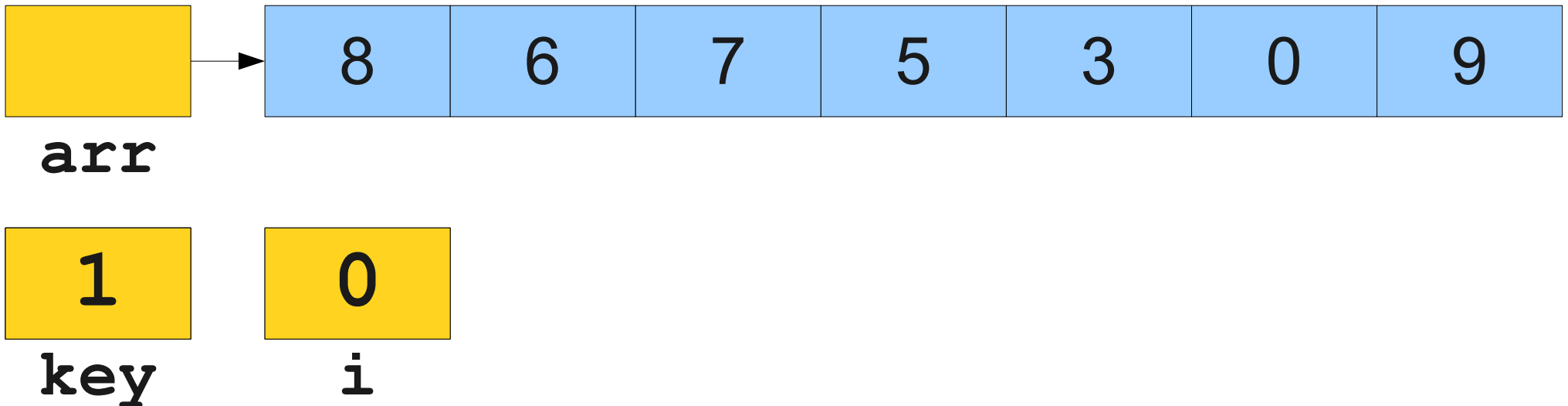
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



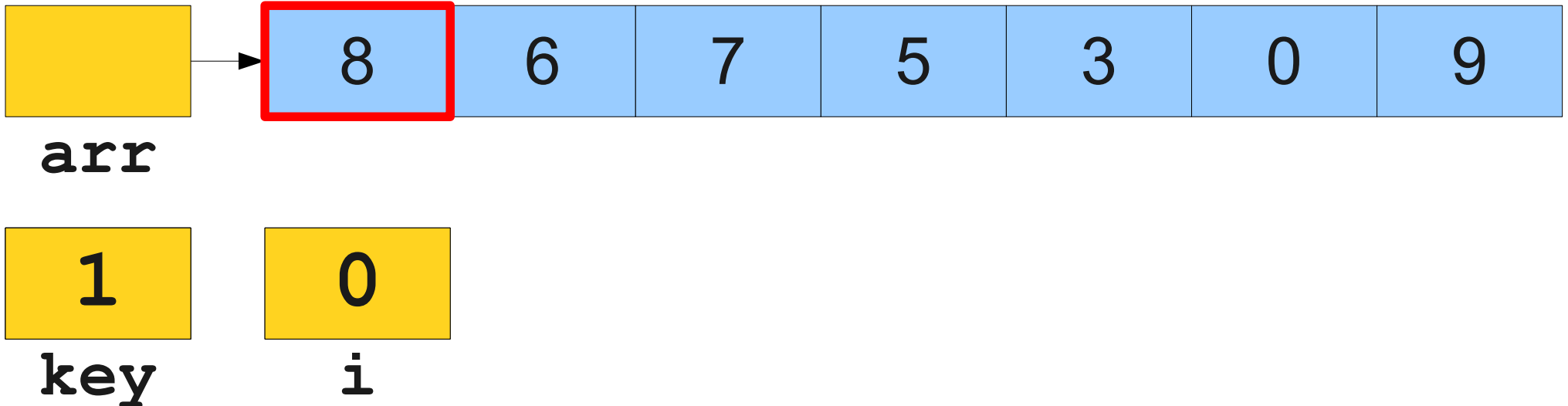
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



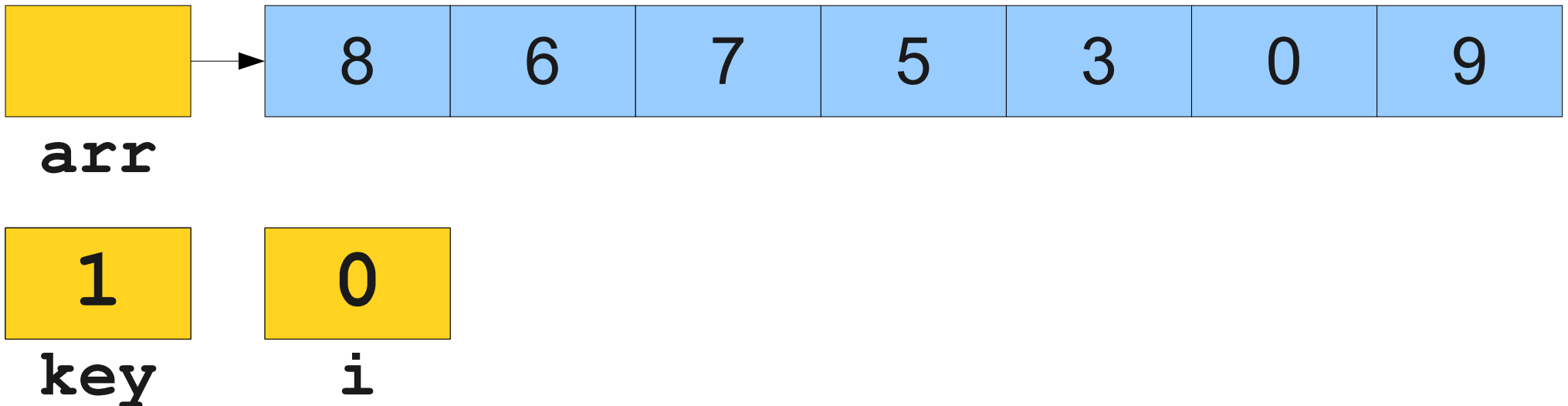
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



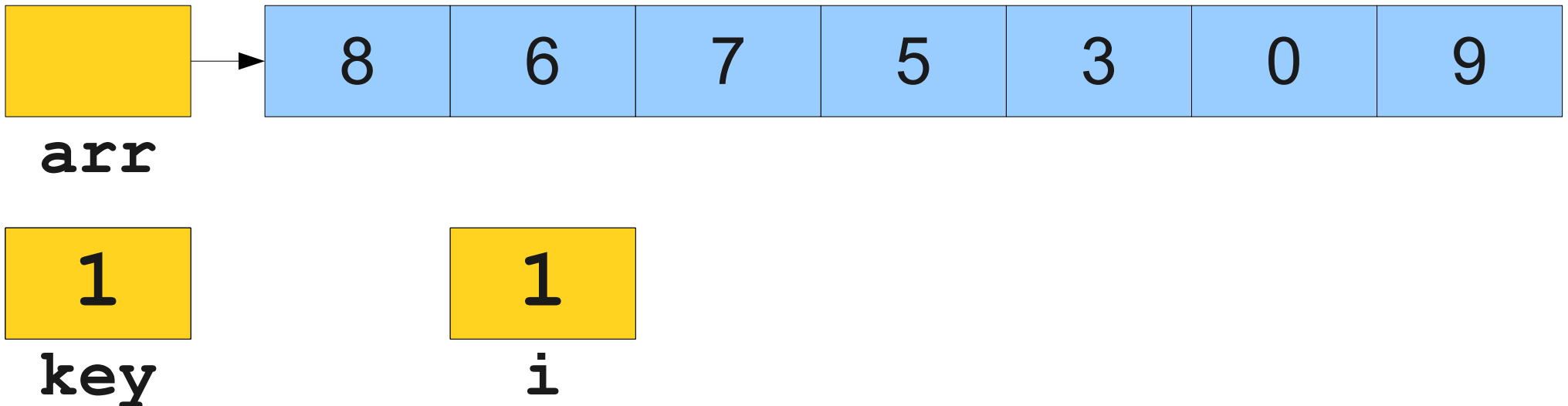
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



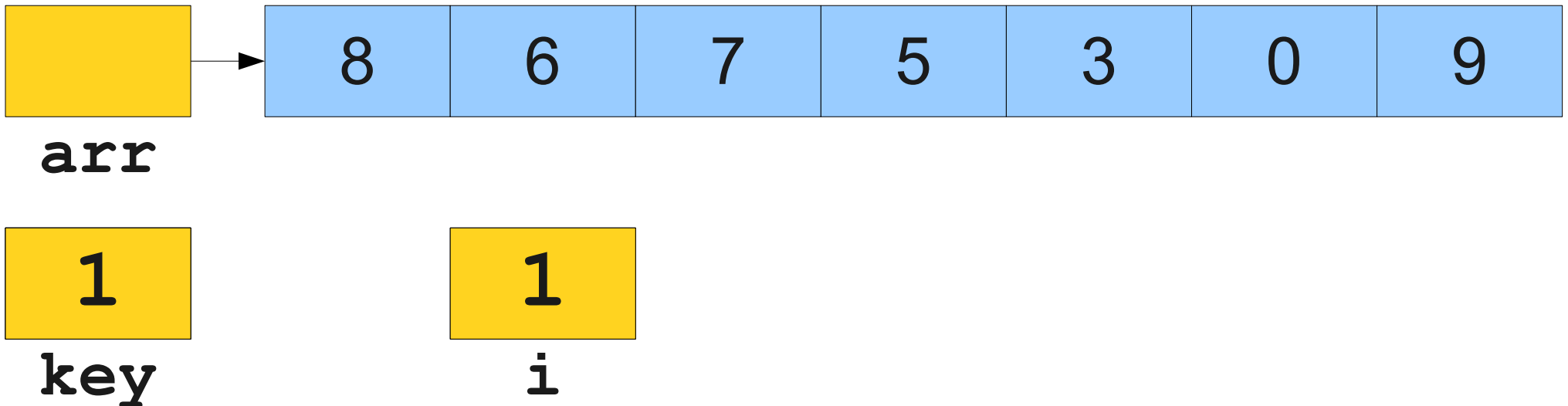
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



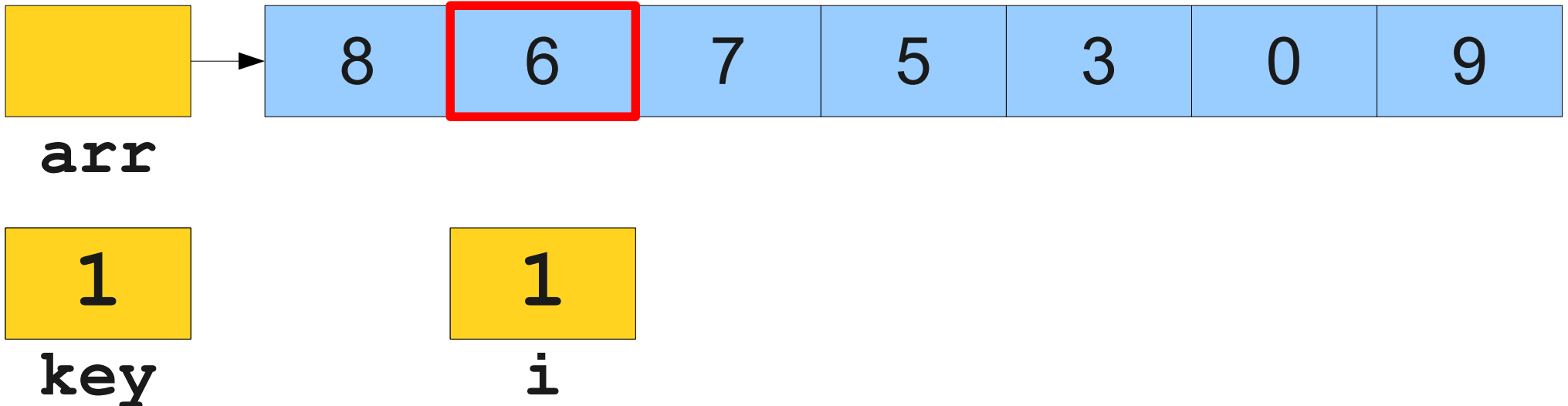
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



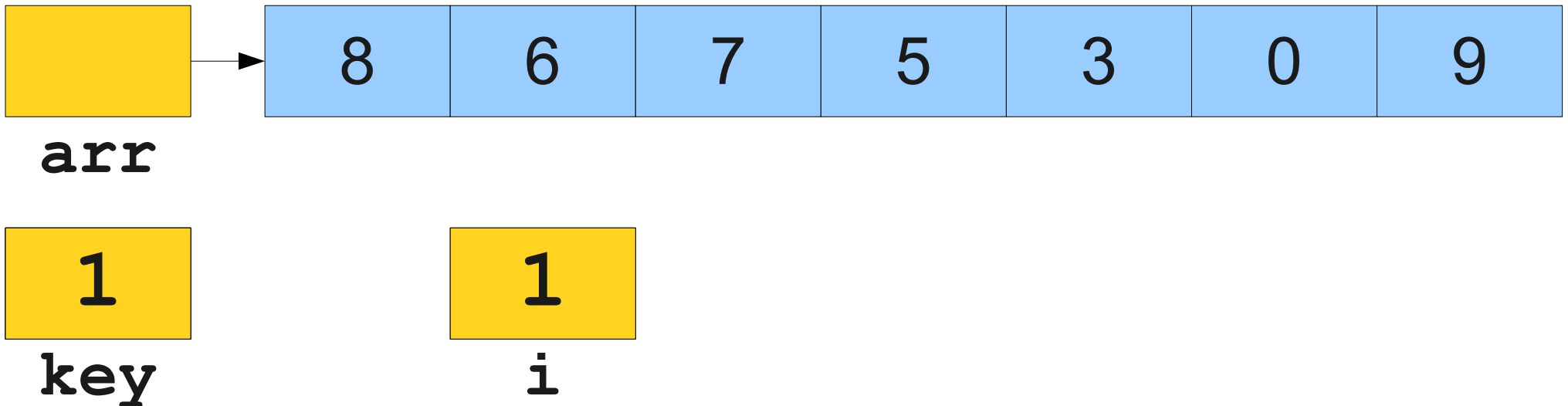
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



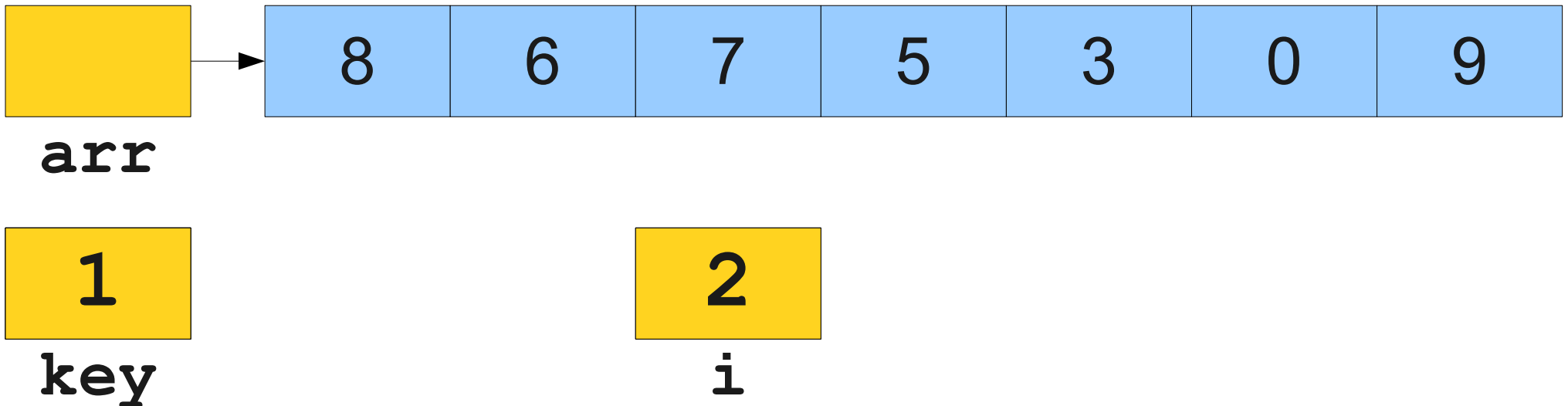
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



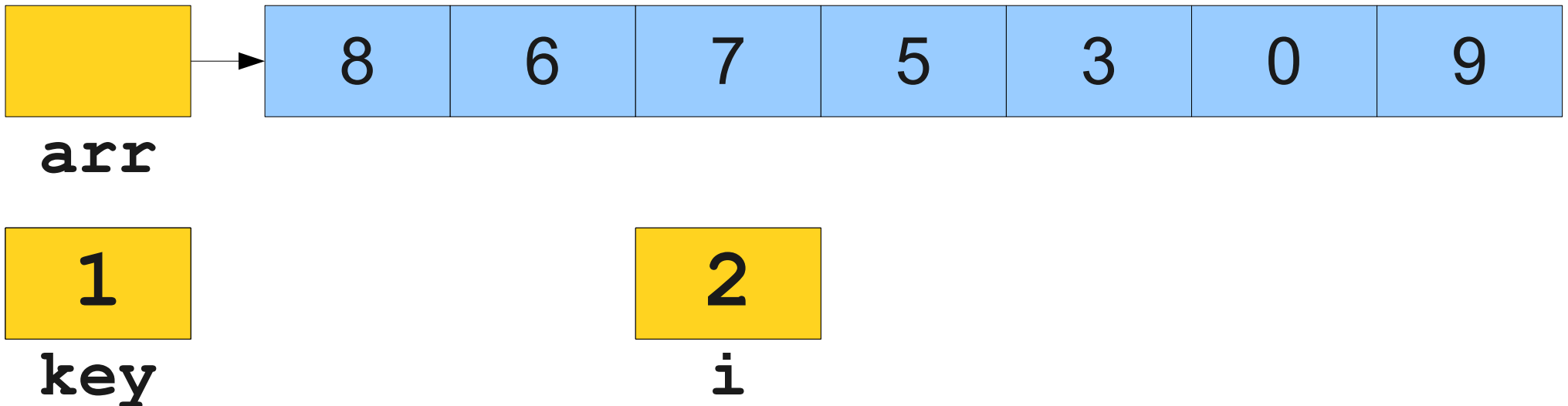
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



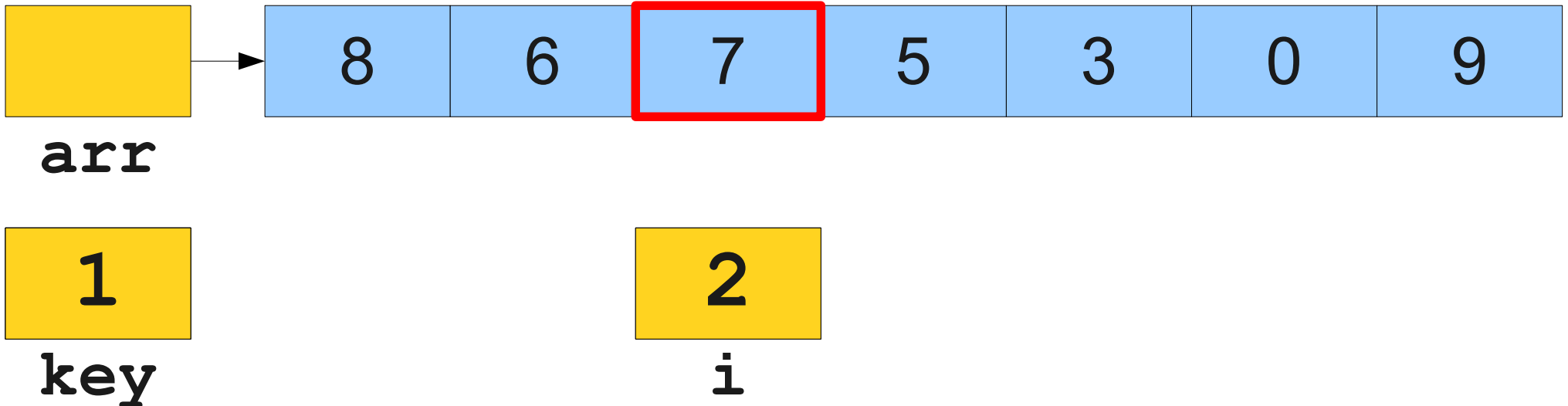
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



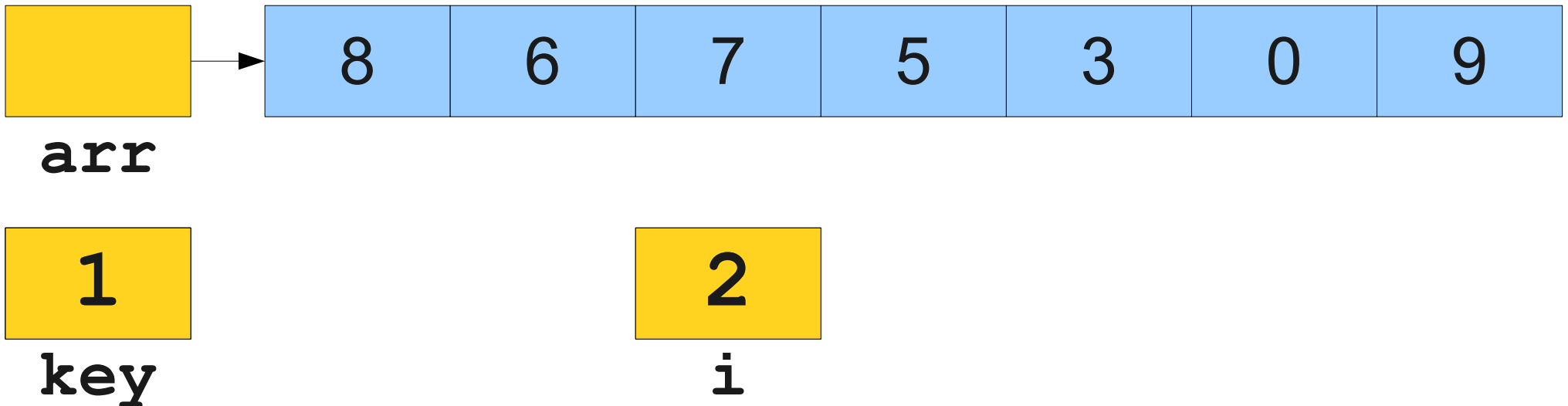
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



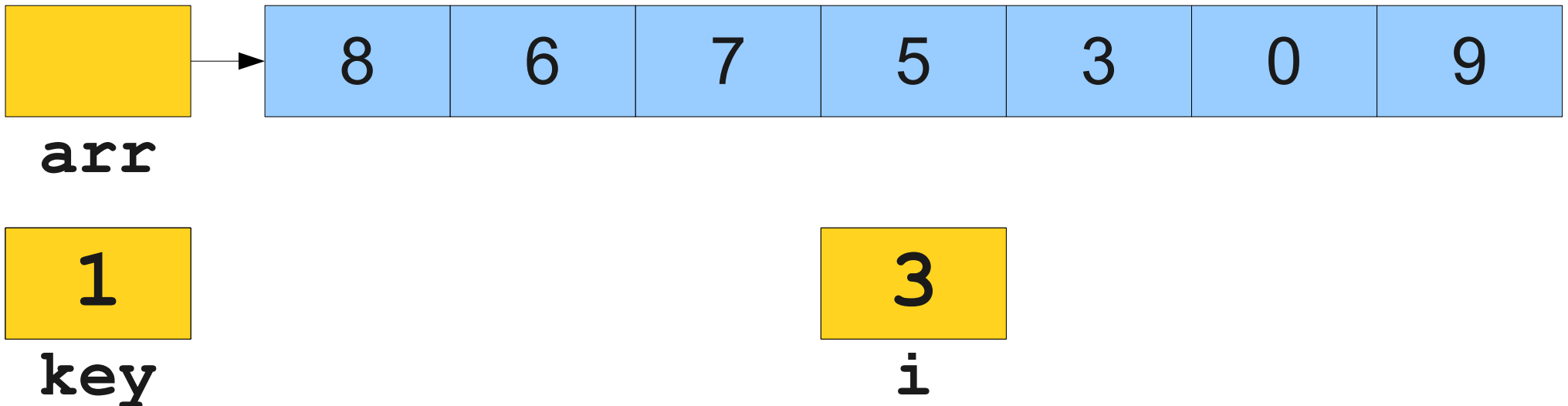
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



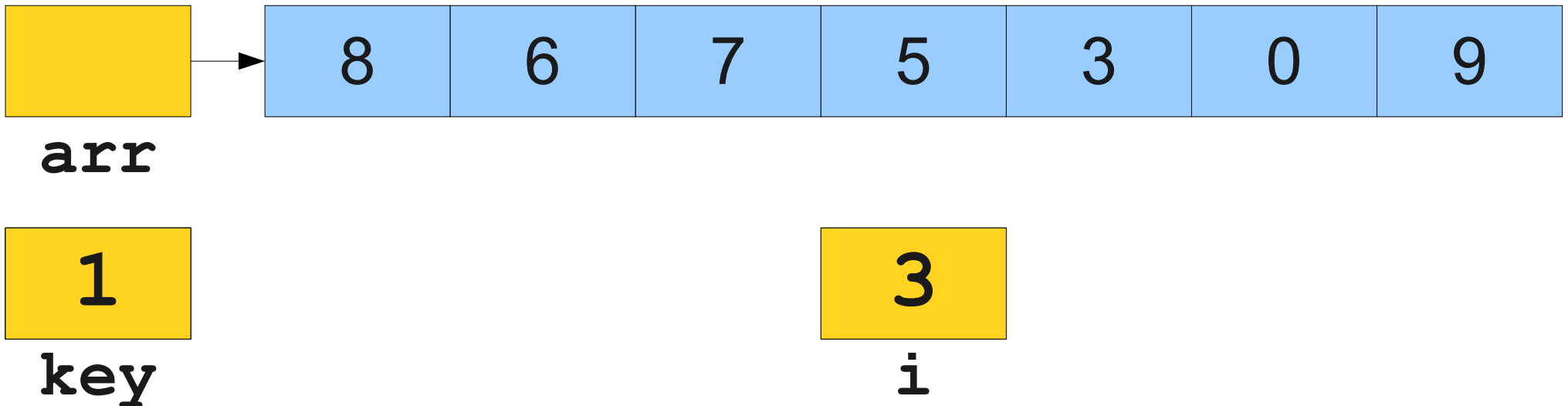
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



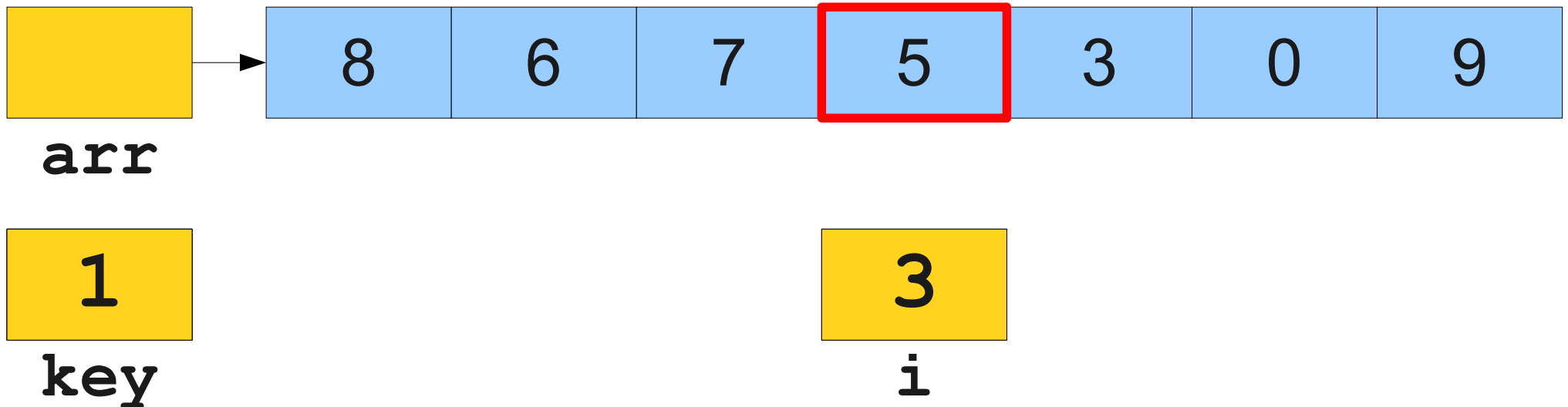
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



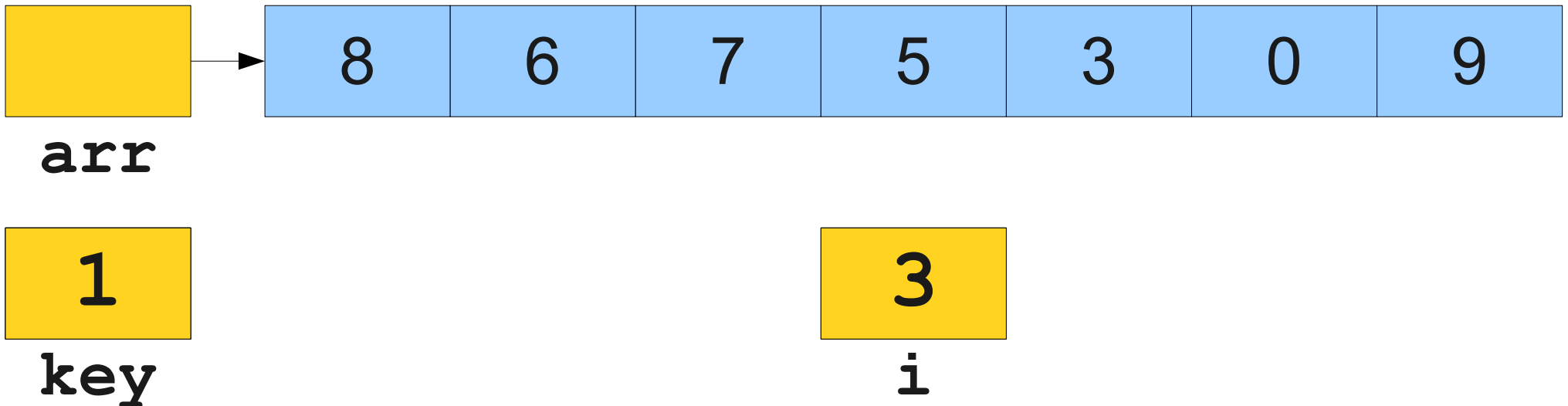
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



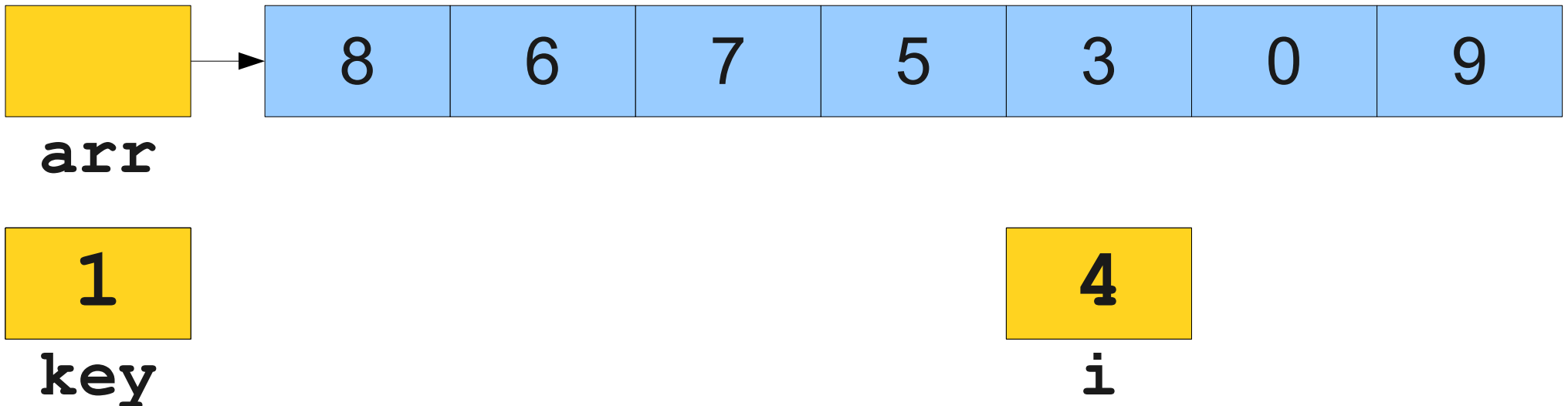
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



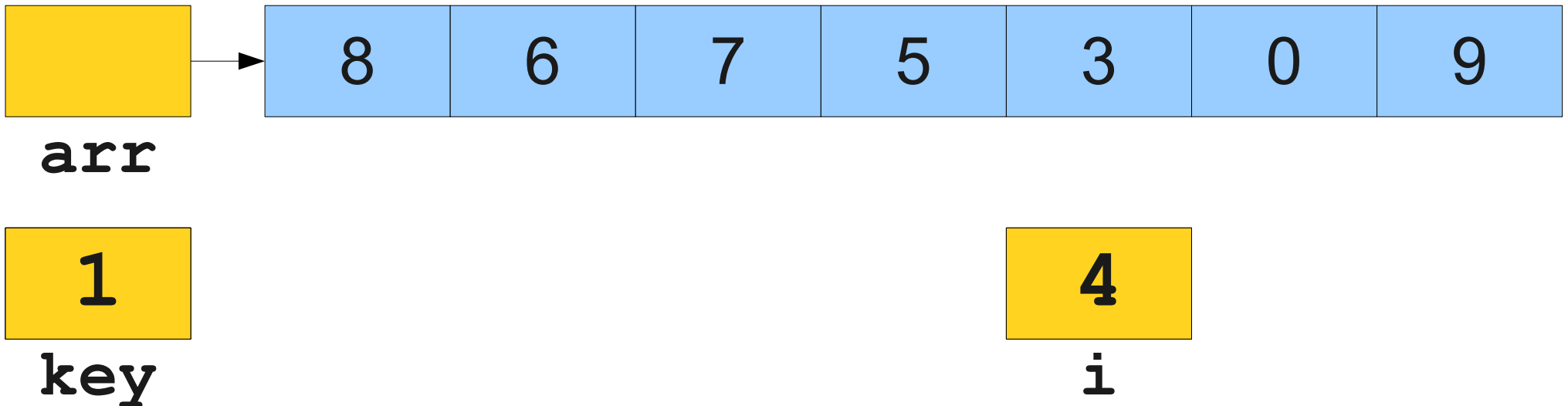
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



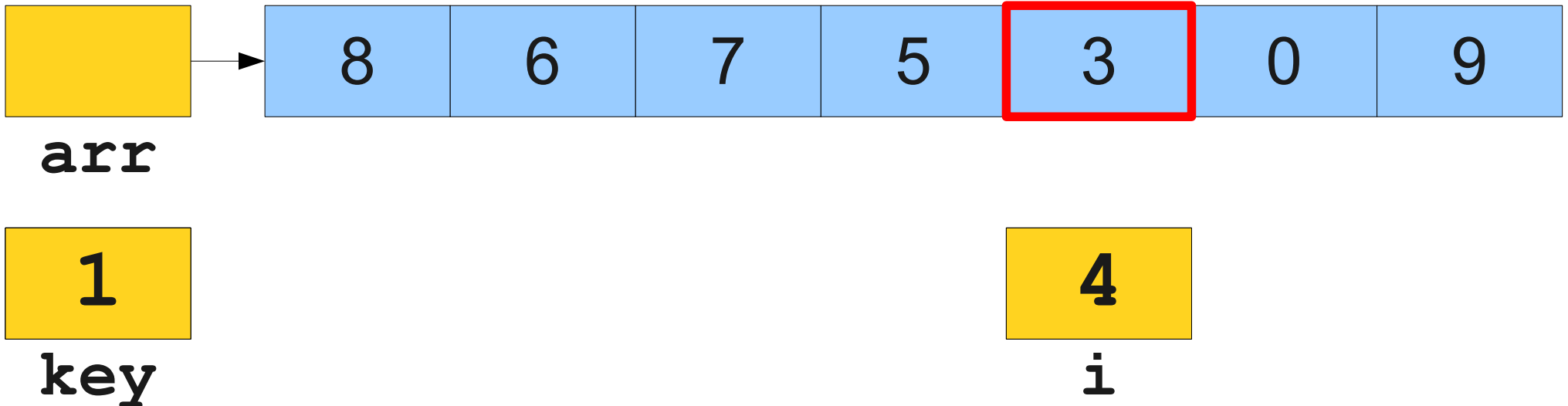
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



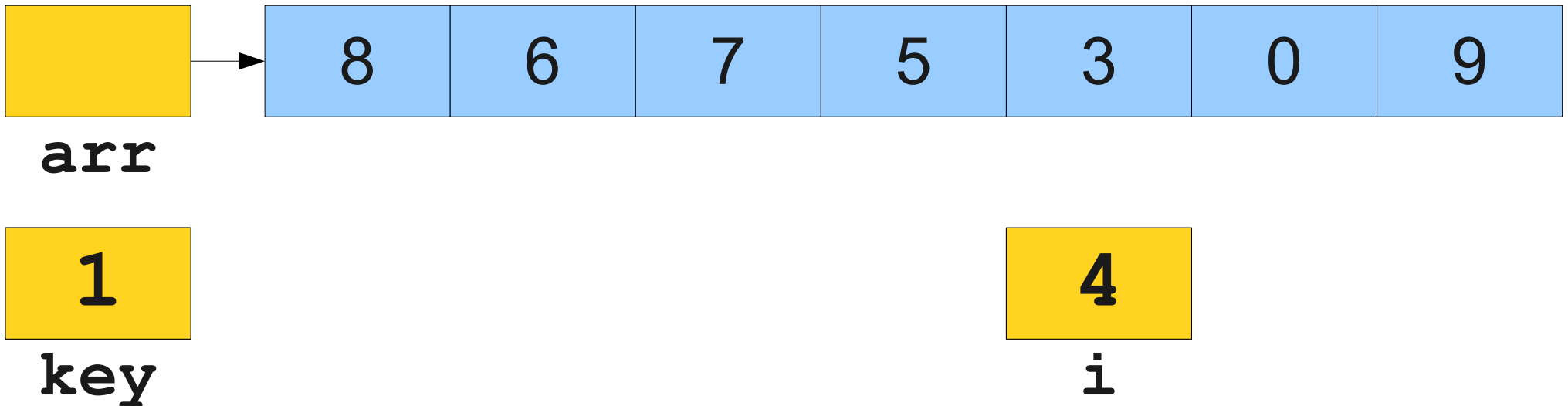
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



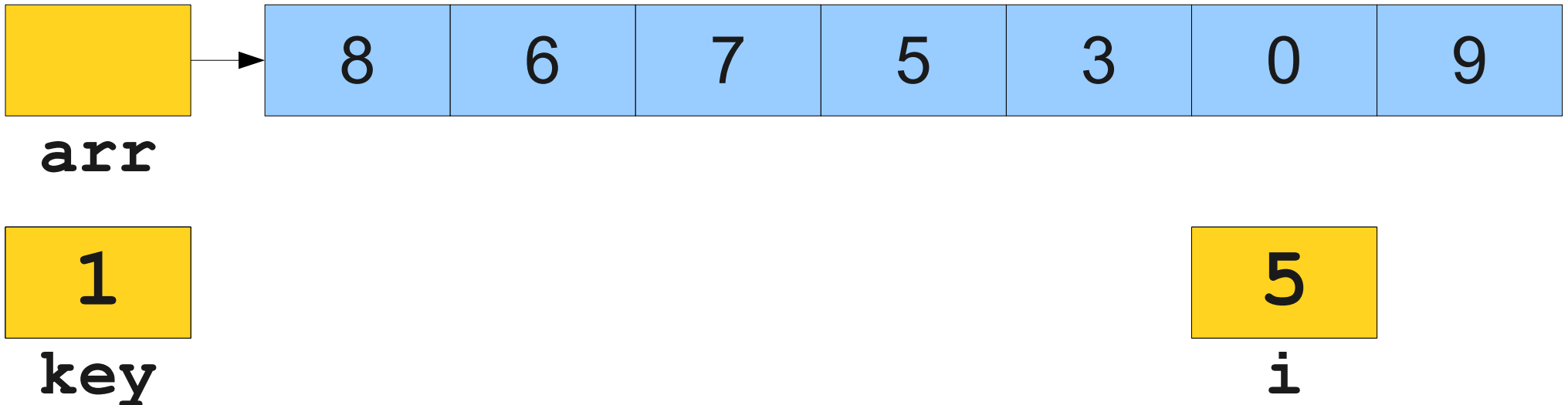
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



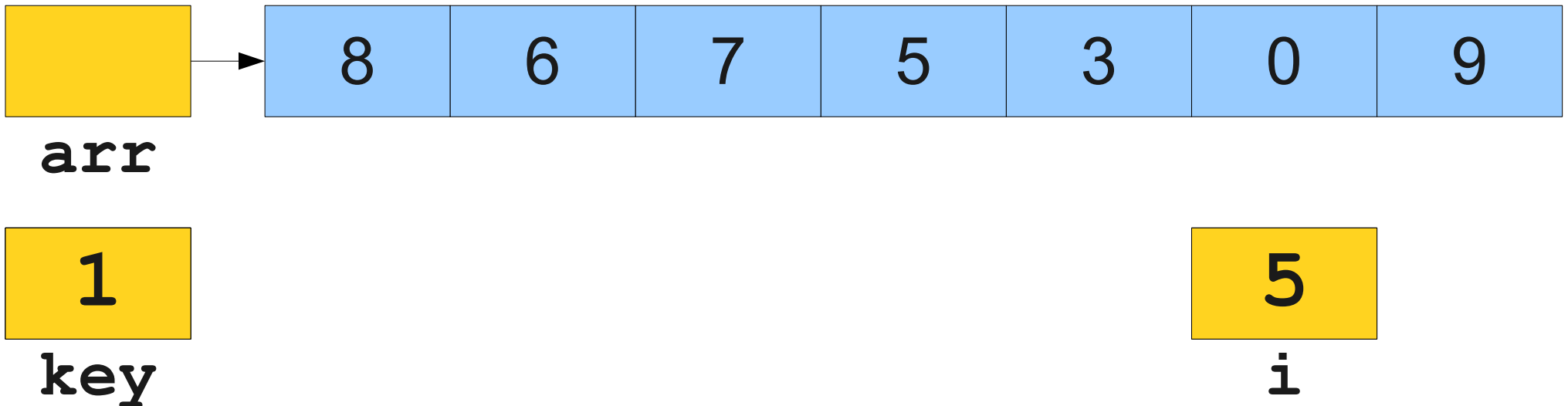
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



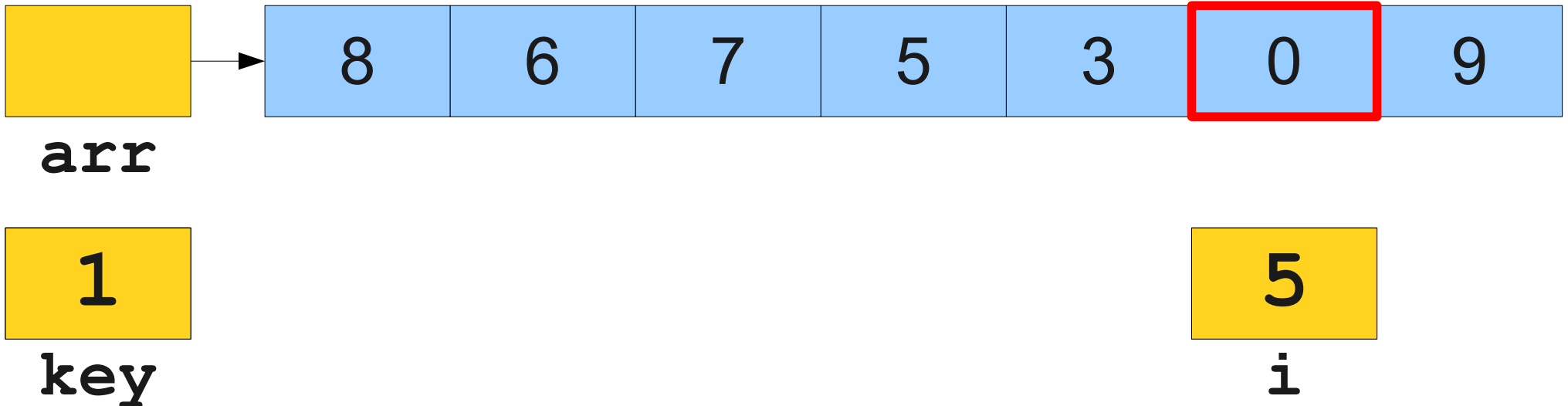
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



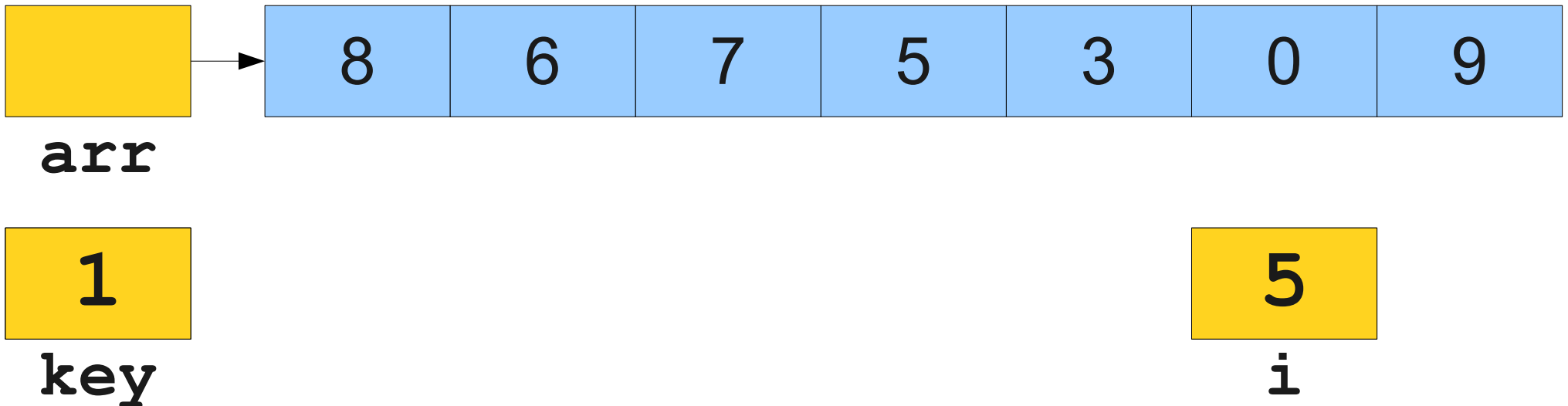
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



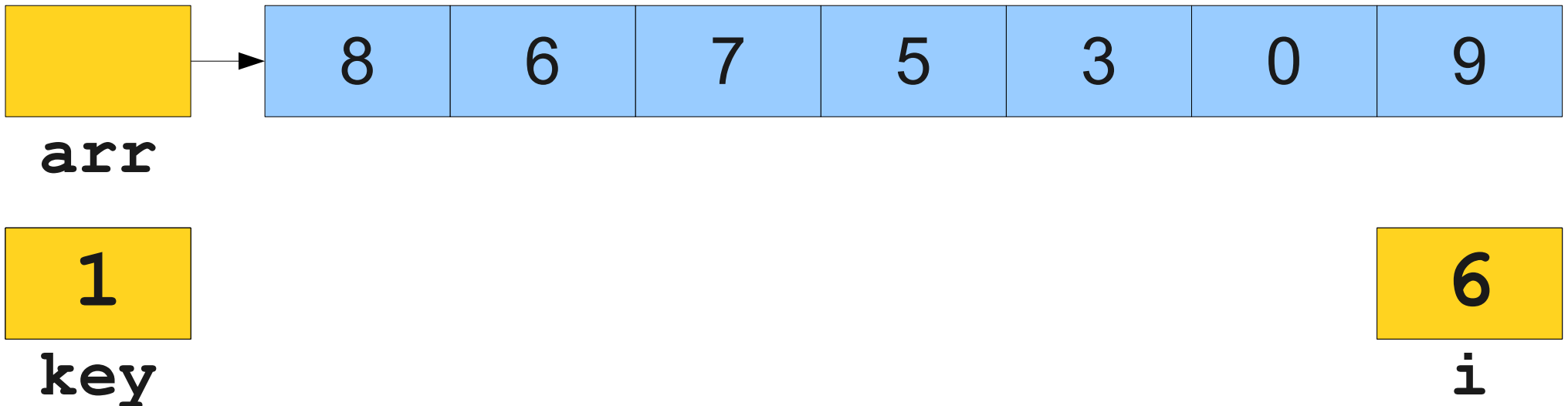
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



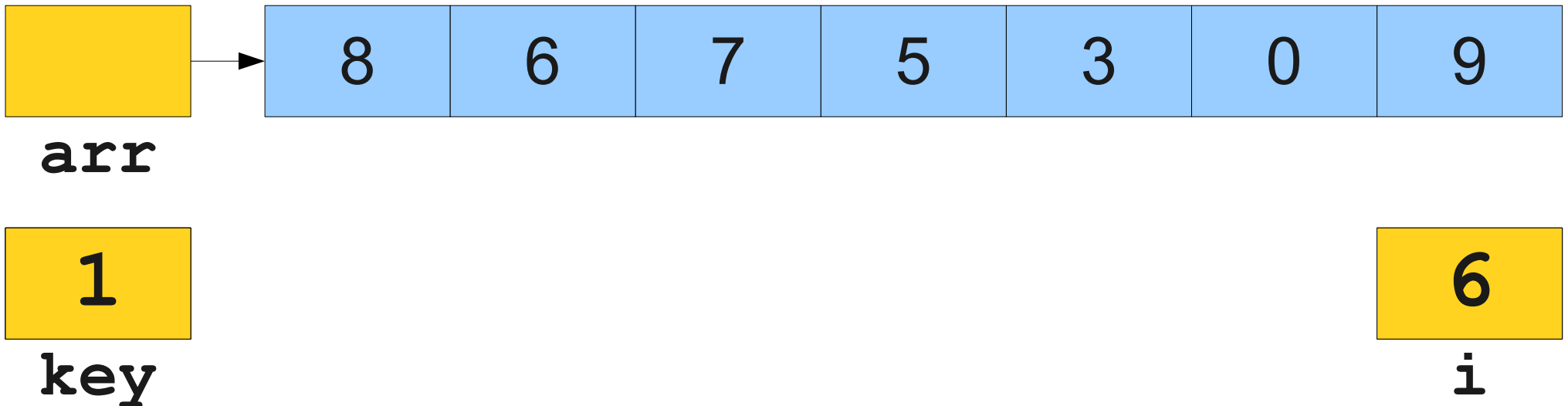
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



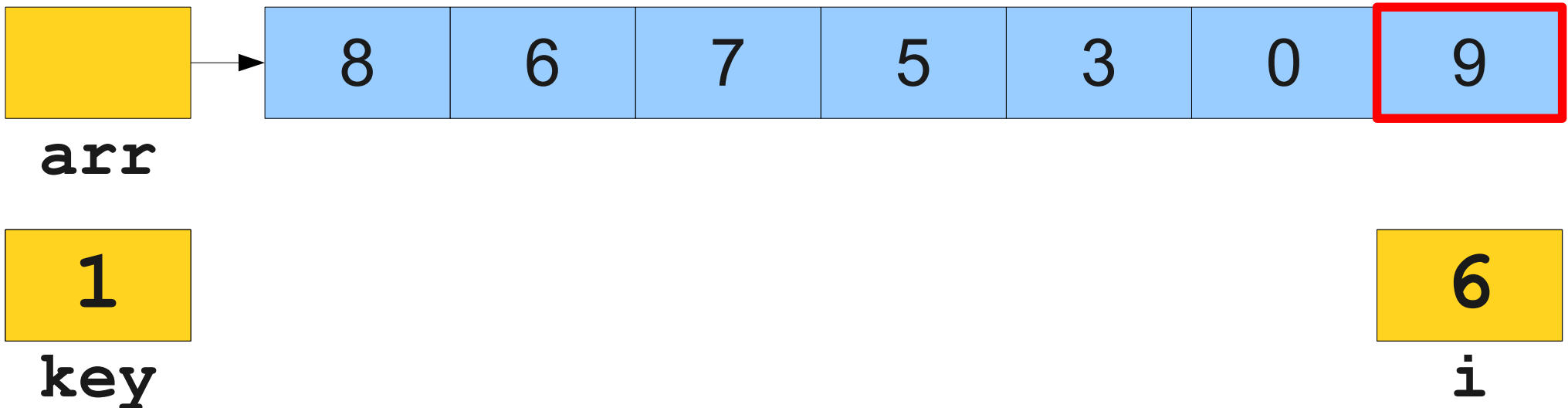
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



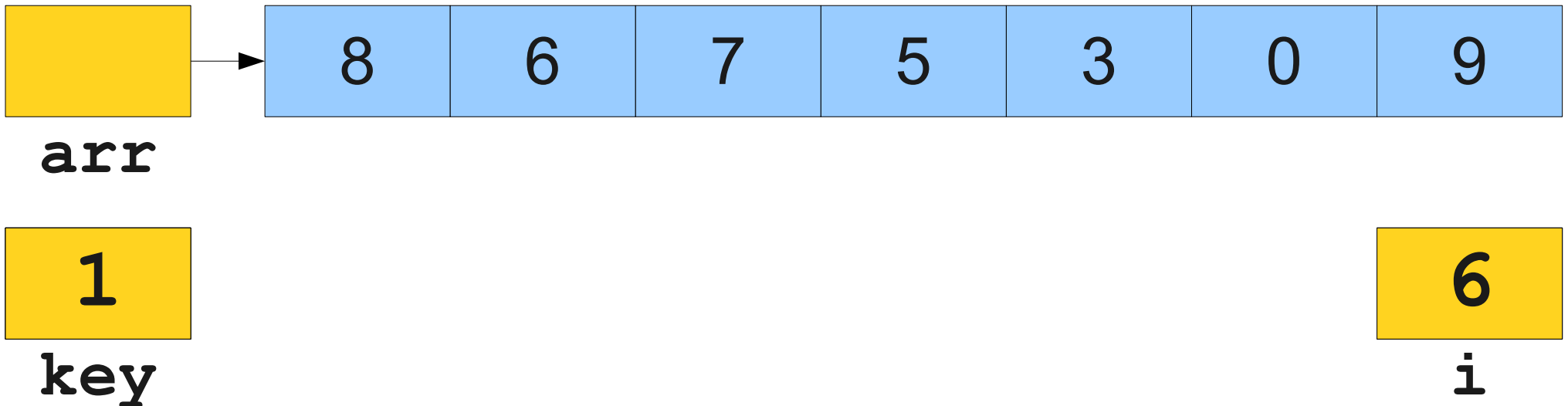
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



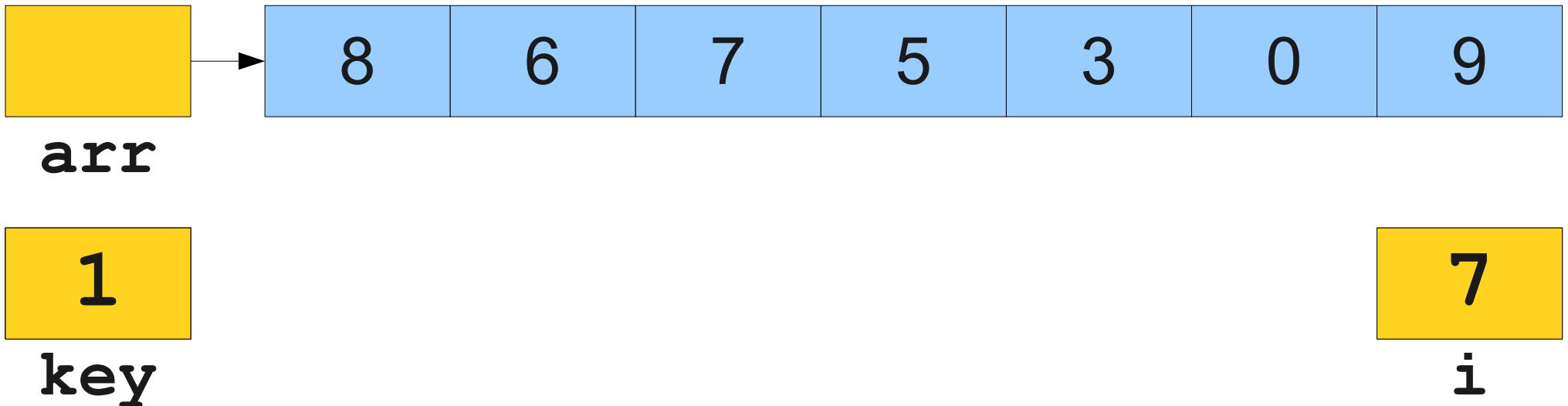
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



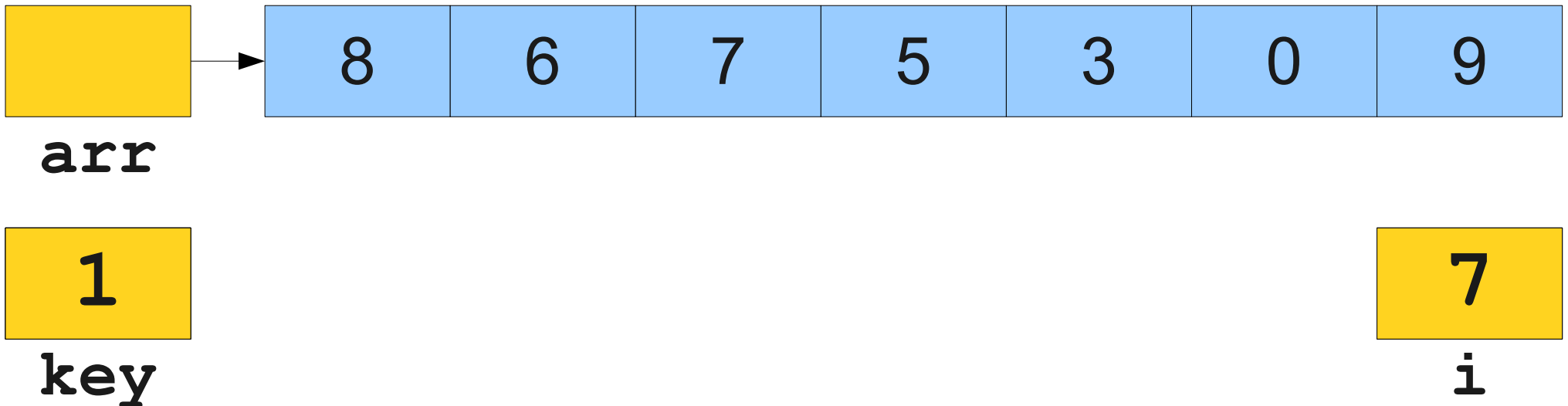
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```



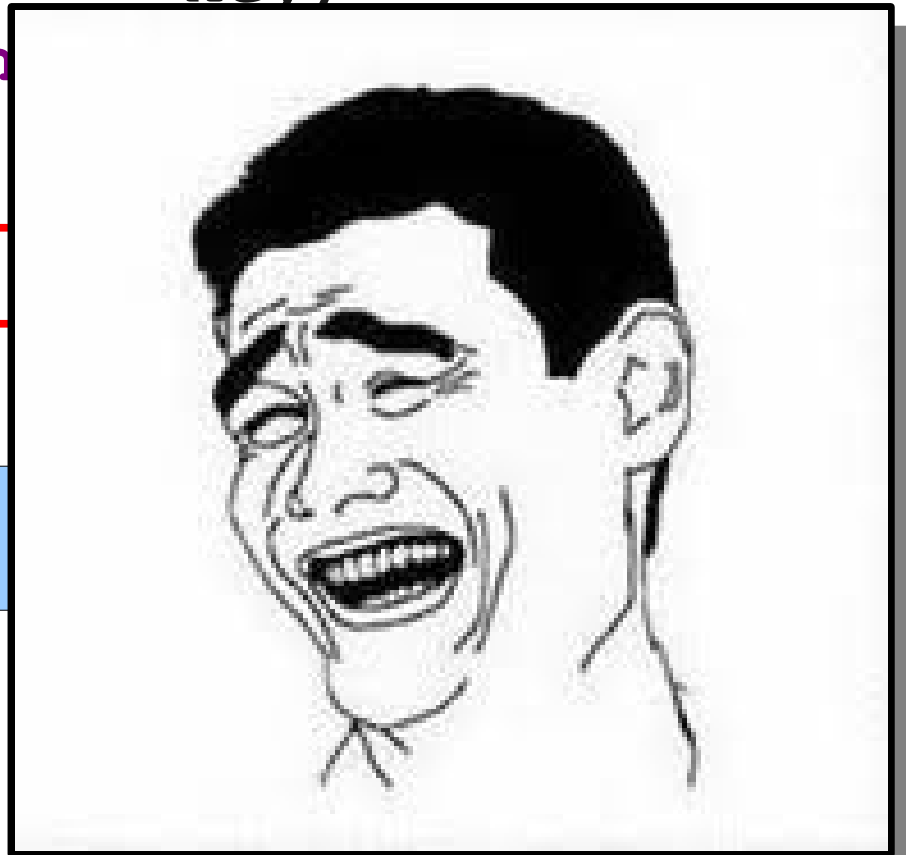
Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```

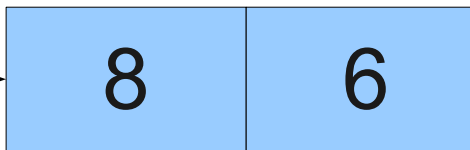


Linear Search

```
private int linearSearch(int[] arr, int key) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == key)  
            return i;  
    }  
    return -1;  
}
```

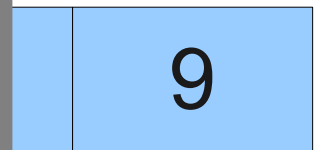


arr



8

6



9



1

key



7

i

Searching II

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

Binary Search

```
private int binarySearch(int[] arr, int key) {
    int lhs = 0;
    int rhs = arr.length - 1;

    while (lhs <= rhs) {
        int mid = (lhs + rhs) / 2;

        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            lhs = mid + 1;
        else
            rhs = mid - 1;
    }
    return -1;
}
```


Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

arr →

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

arr →

1

2

3

5

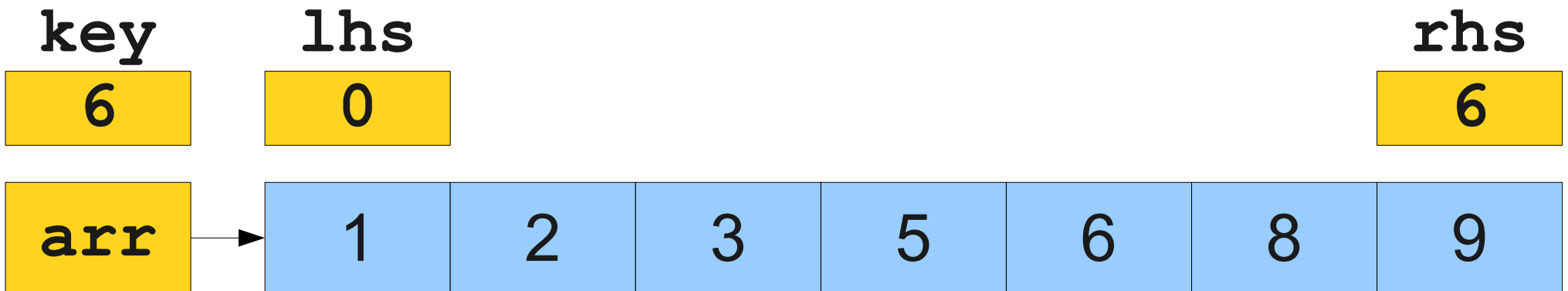
6

8

9

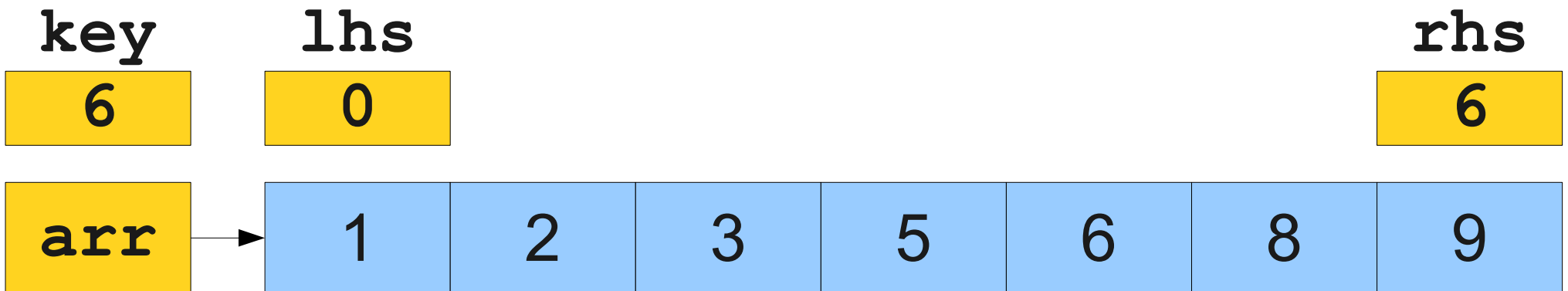
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



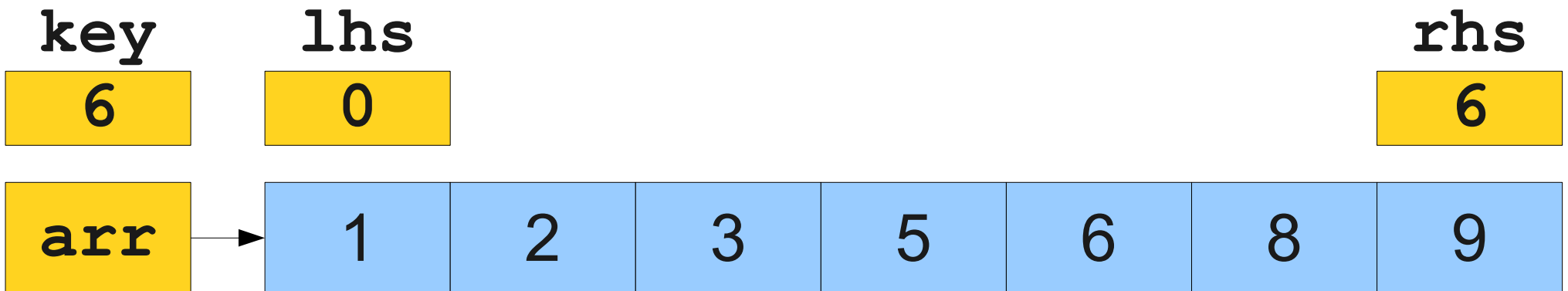
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



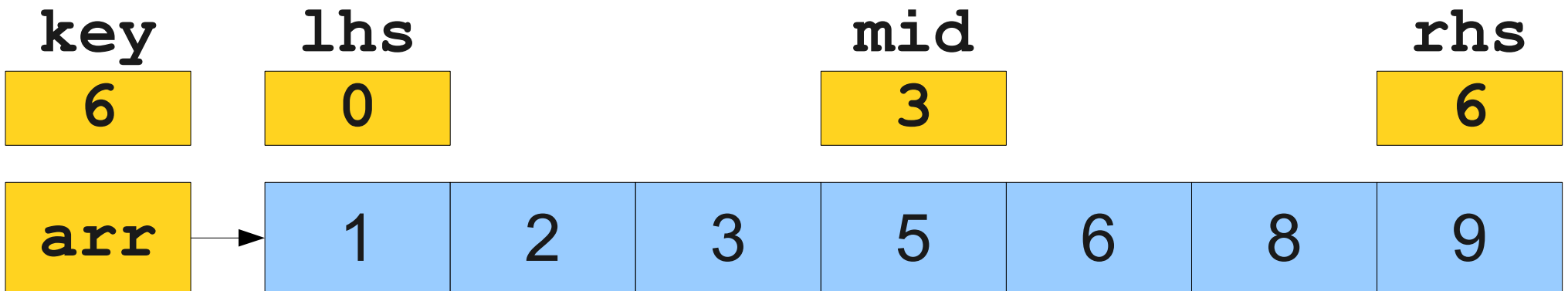
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



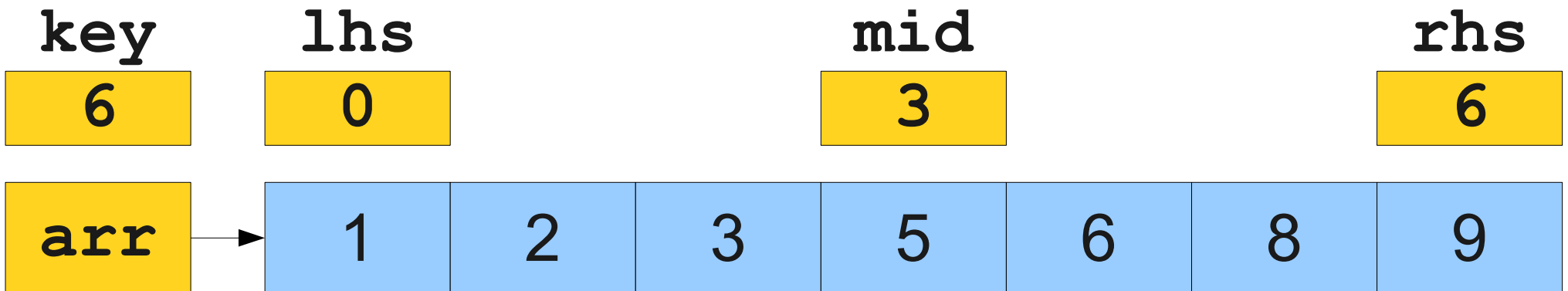
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



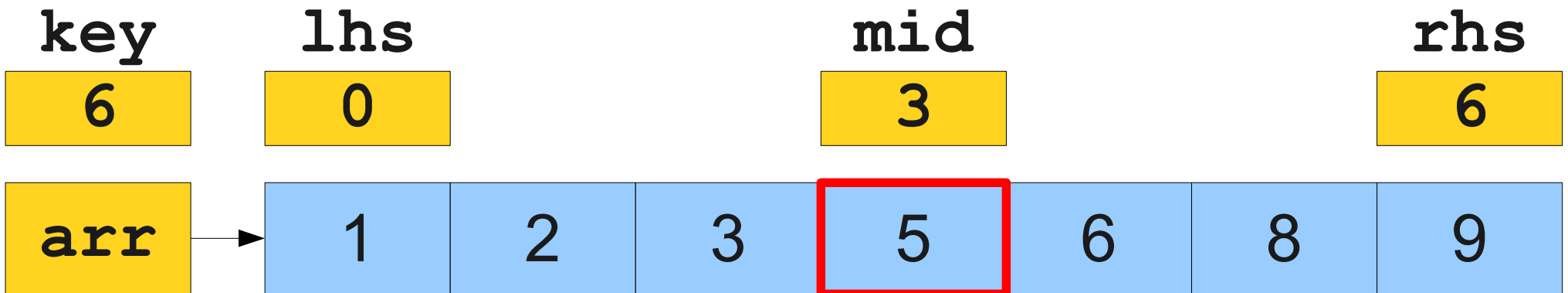
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



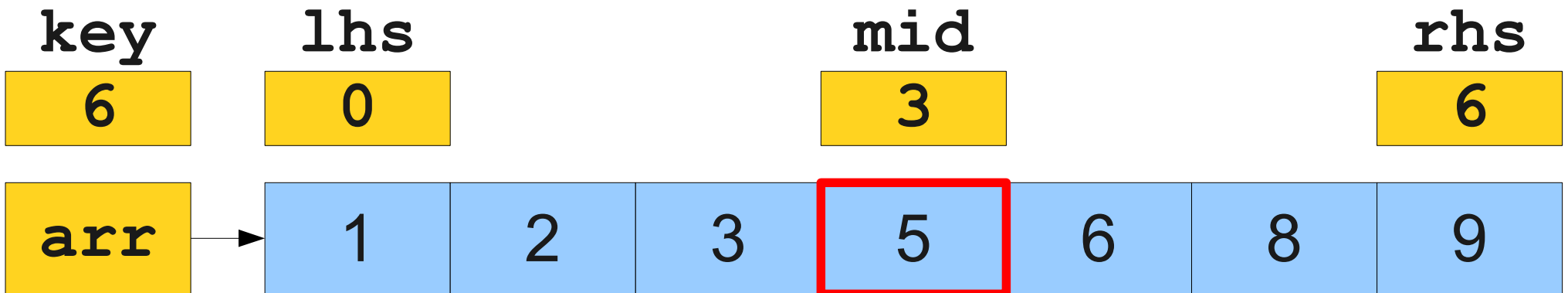
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



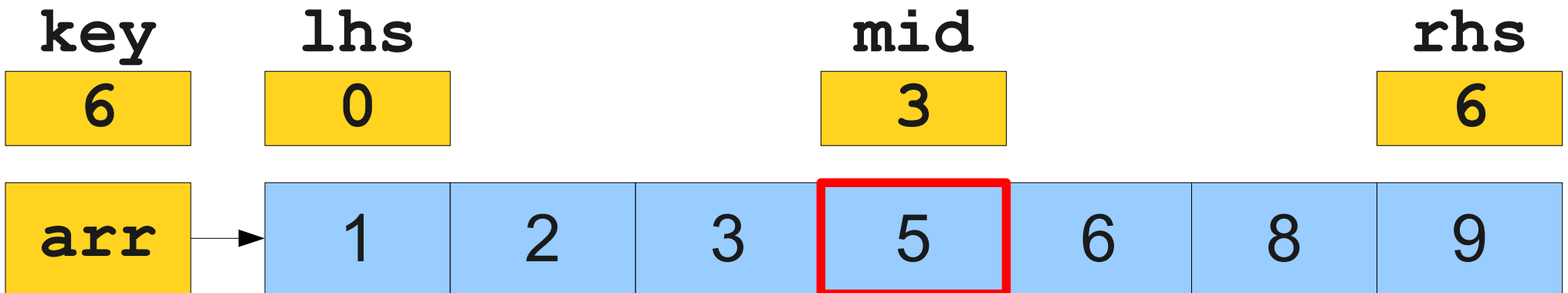
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



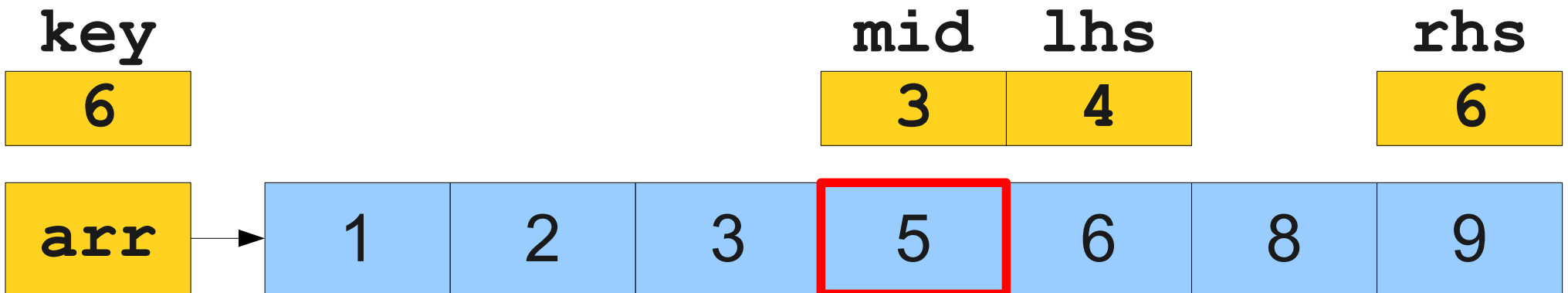
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



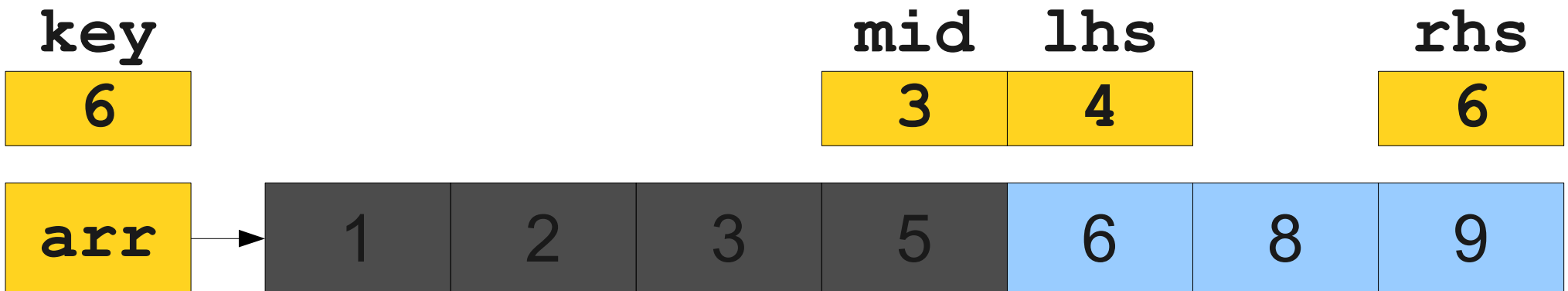
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



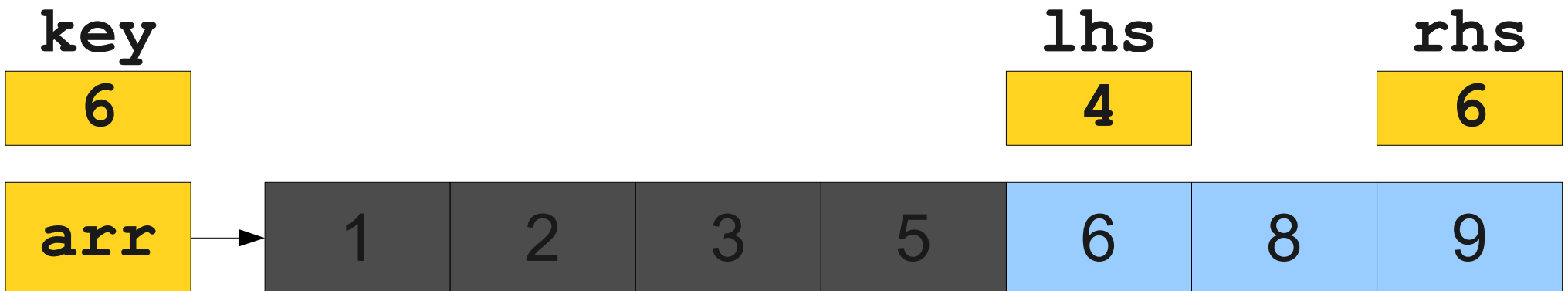
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



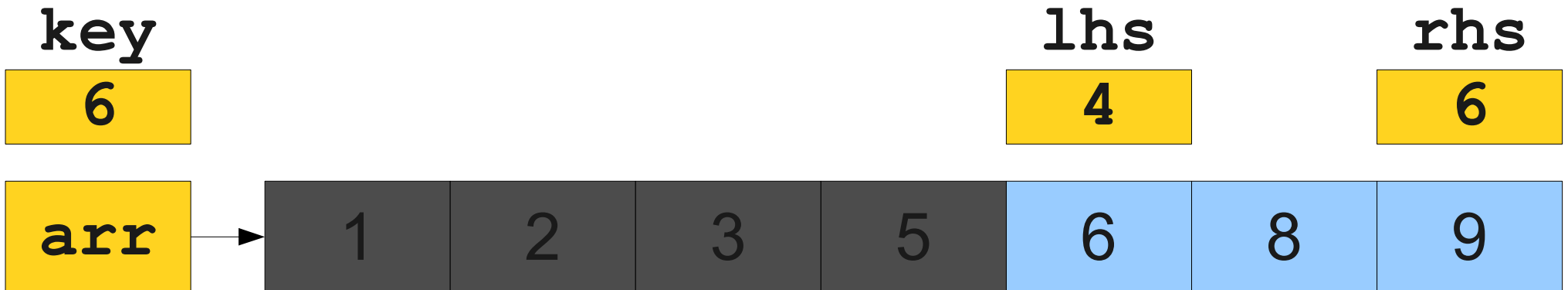
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



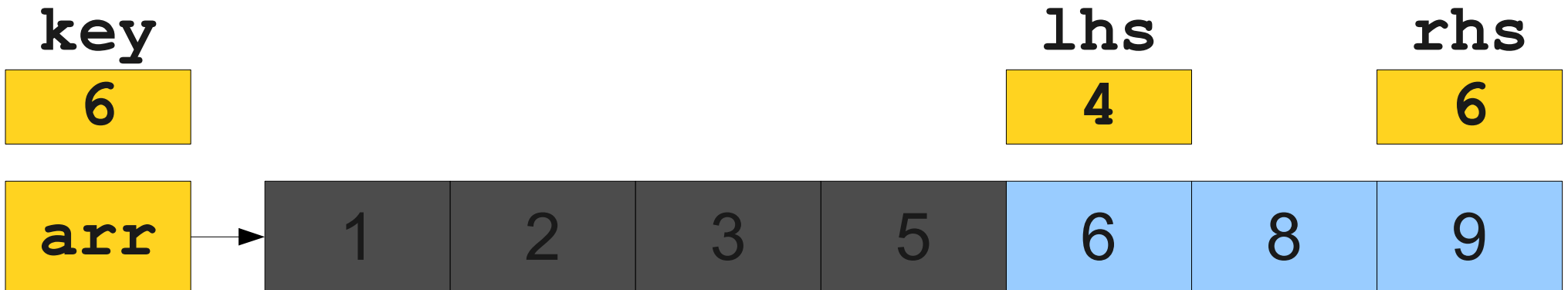
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs mid rhs

4

5

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

mid

rhs

4

5

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

4

mid

5

rhs

6

arr

1

2

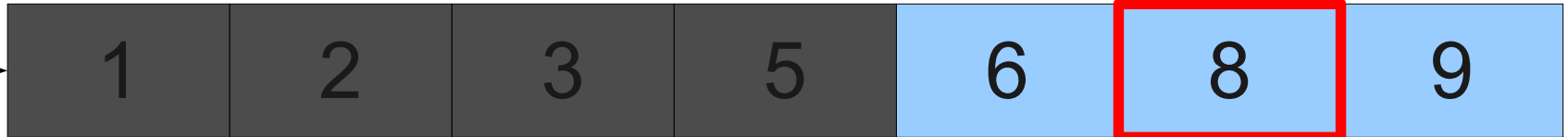
3

5

6

8

9



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

4

mid

5

rhs

6

arr

1

2

3

5

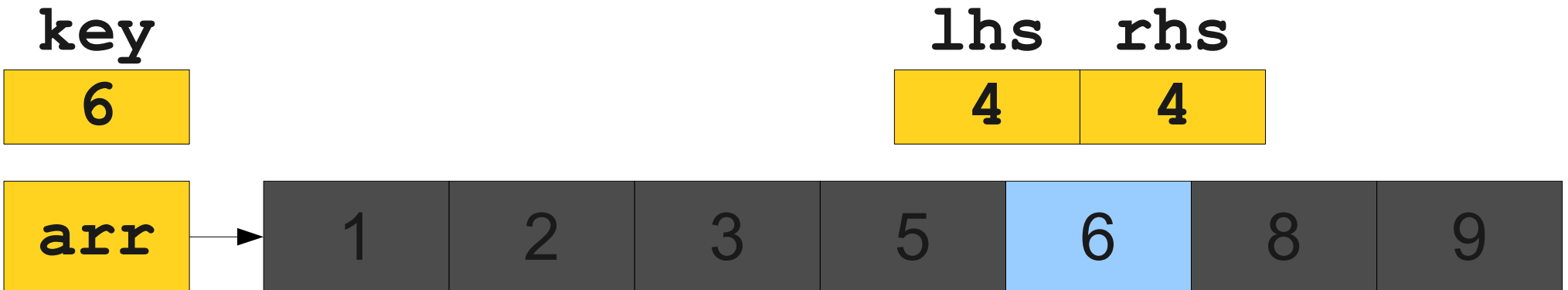
6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

lhs

rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
6

mid
4
lhs rhs
4 4

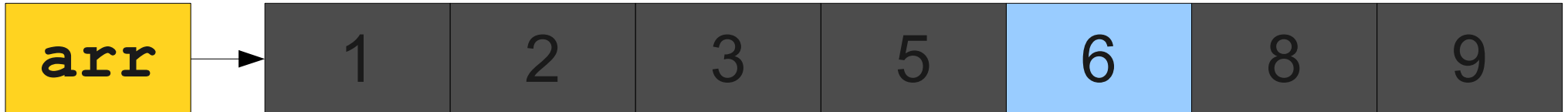


Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
6

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
6

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

6

mid

4

lhs rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
6

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;
```



Binary Search

```
private int binarySearch(int[] arr, int key) {
    int lhs = 0;
    int rhs = arr.length - 1;

    while (lhs <= rhs) {
        int mid = (lhs + rhs) / 2;

        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            lhs = mid + 1;
        else
            rhs = mid - 1;
    }
    return -1;
}
```

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

arr →

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

arr

1

2

3

5

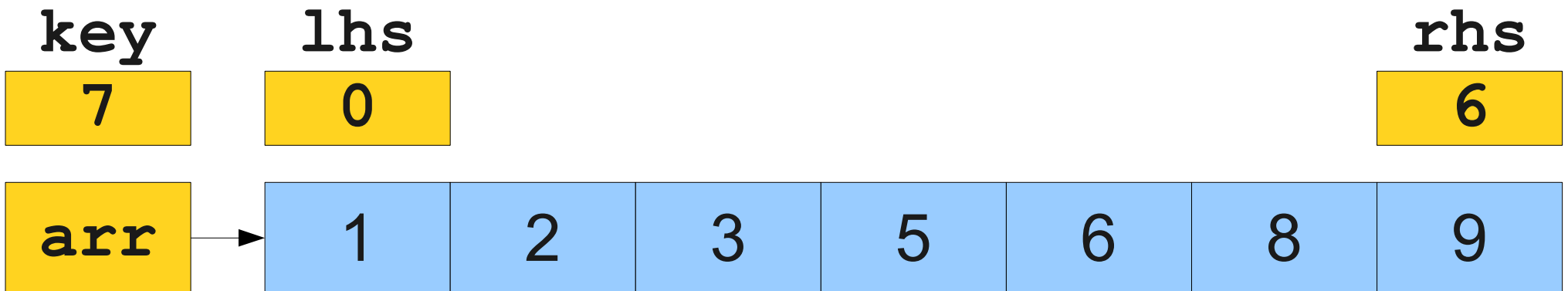
6

8

9

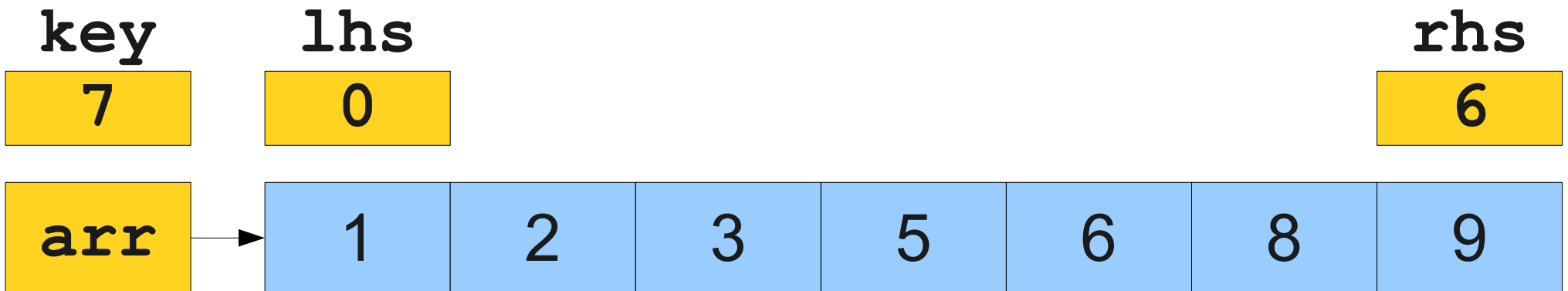
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



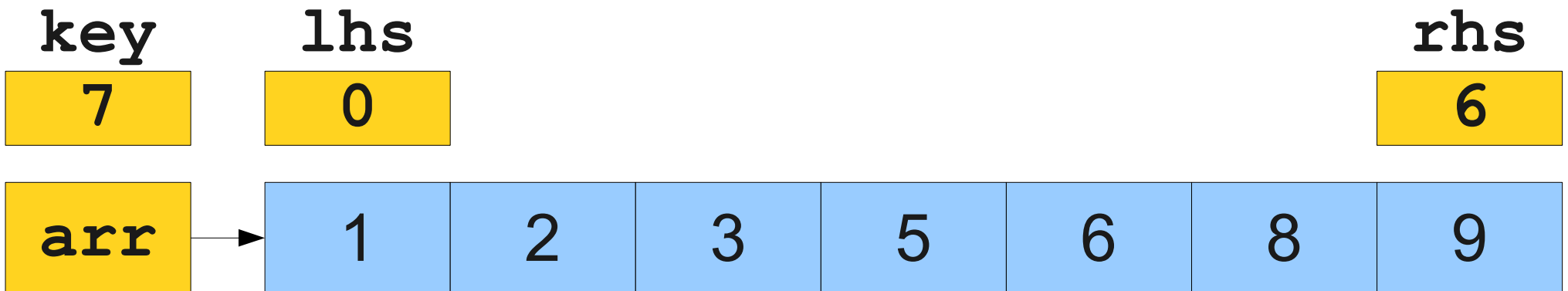
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



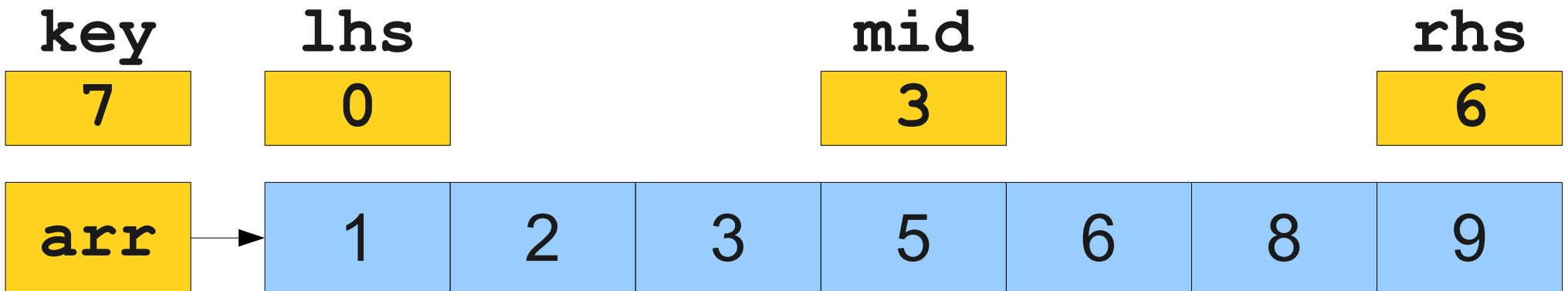
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



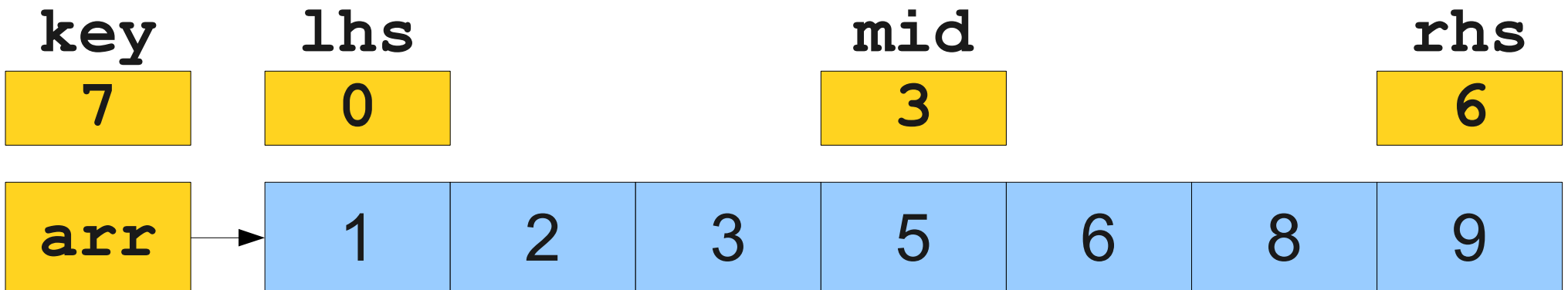
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



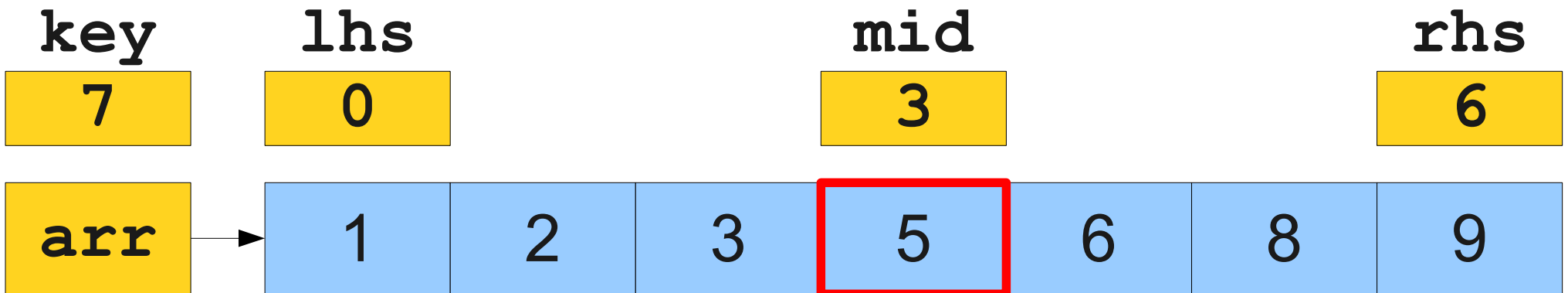
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



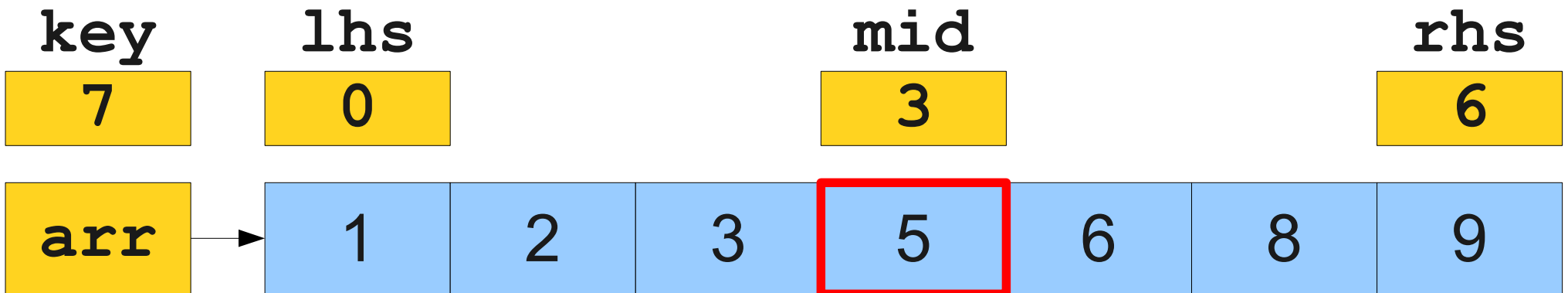
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



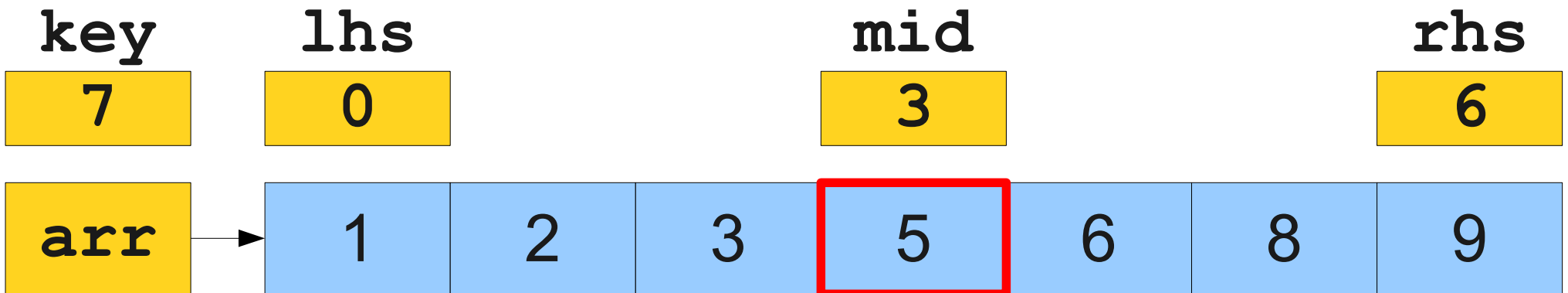
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



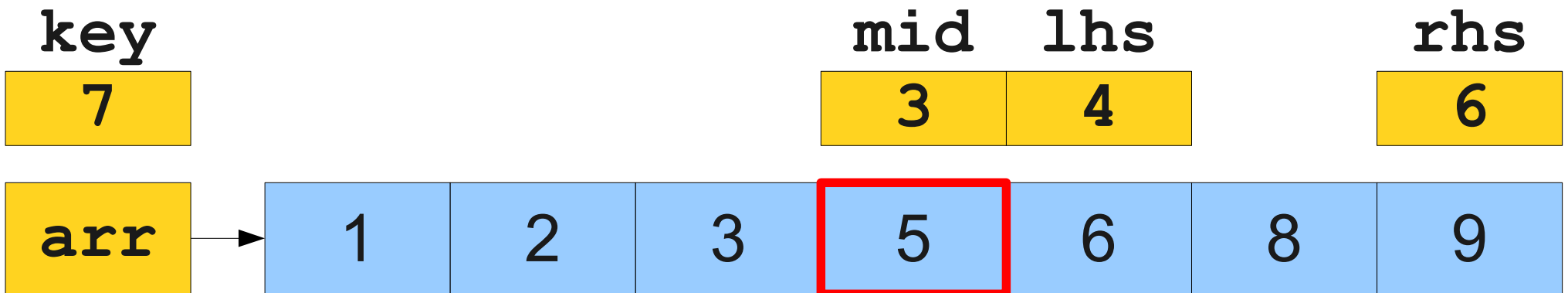
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



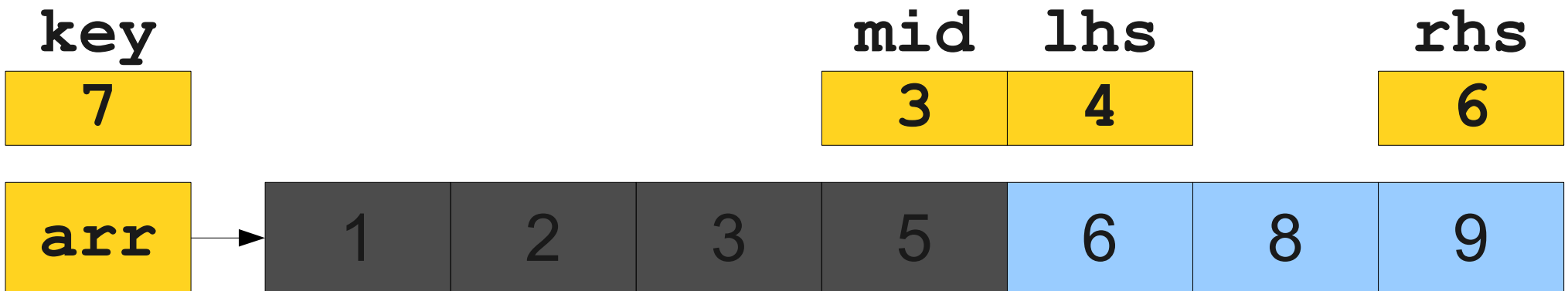
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



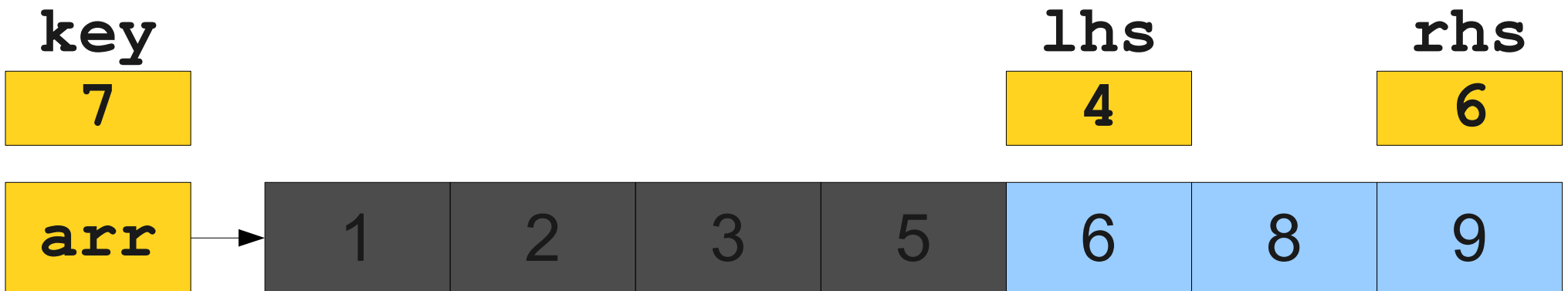
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



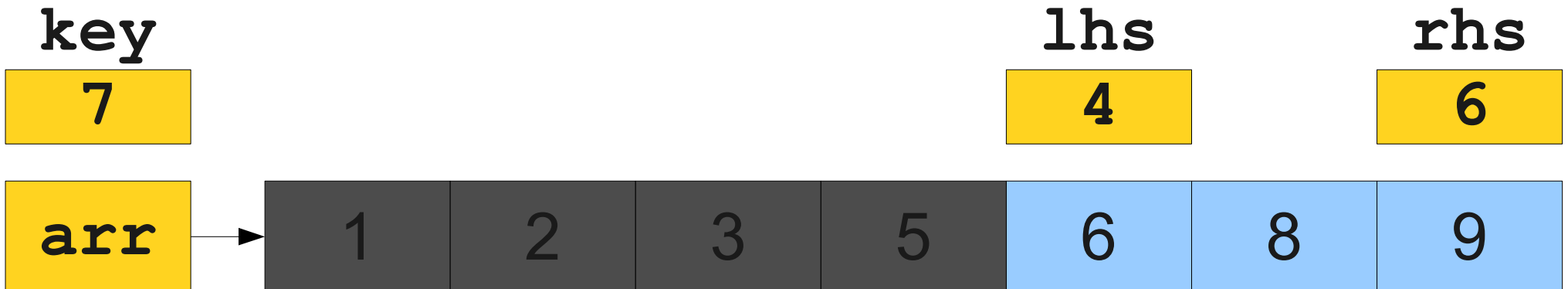
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



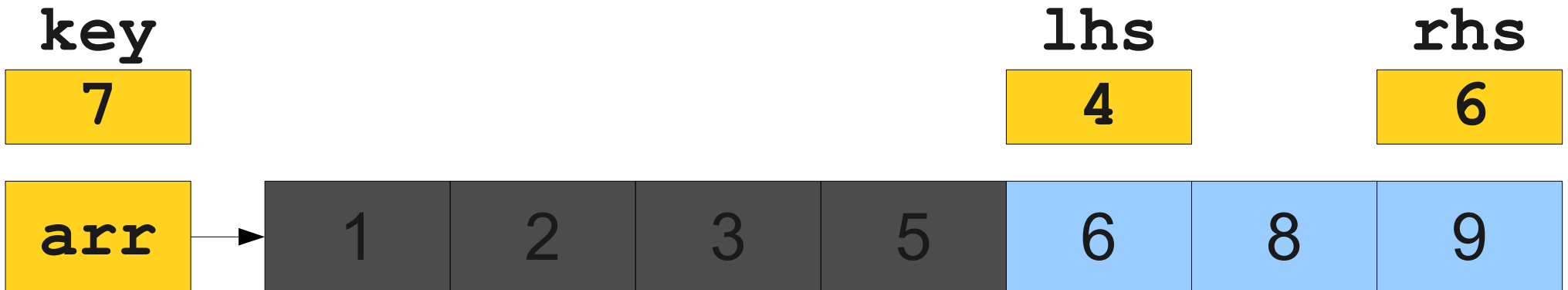
Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs mid rhs

4

5

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

4

mid

5

rhs

6

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
7

lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

lhs

rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
7

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
7

mid
4
lhs rhs
4 4

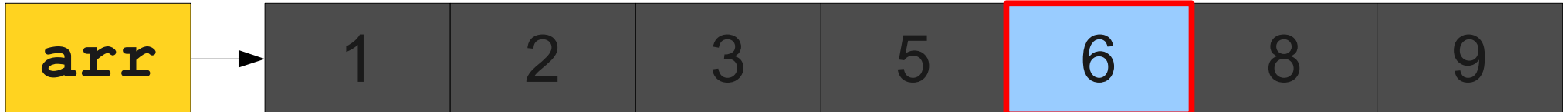


Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
7

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key
7

mid
4
lhs rhs
4 4



Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

mid

4

lhs rhs

4

4

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

rhs

4

lhs

5

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

rhs

lhs

4

5

arr

1

2

3

5

6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key)  
            return mid;  
        else if (arr[mid] < key)  
            lhs = mid + 1;  
        else  
            rhs = mid - 1;  
    }  
    return -1;  
}
```

key

7

rhs

lhs

4

5

arr

1

2

3

5

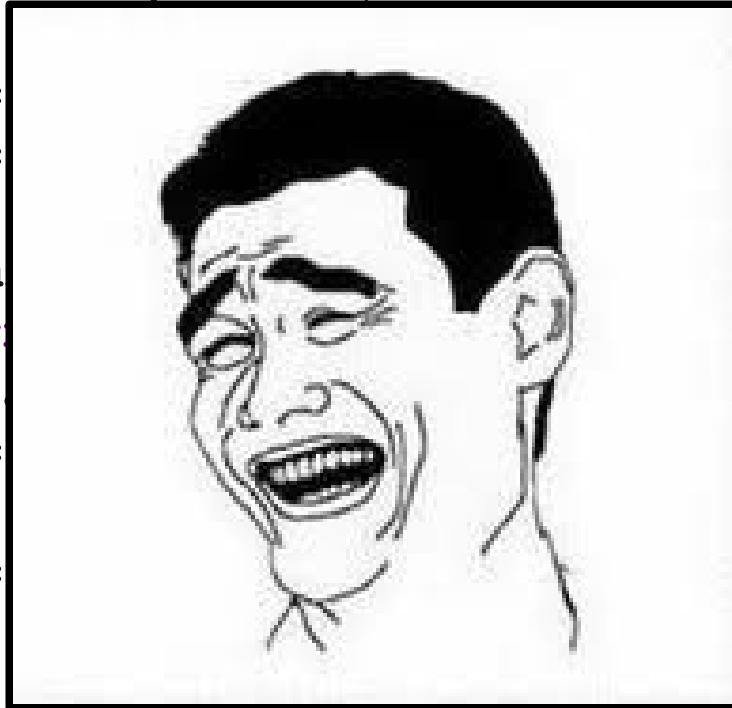
6

8

9

Binary Search

```
private int binarySearch(int[] arr, int key) {  
    int lhs = 0;  
    int rhs = arr.length - 1;  
  
    while (lhs <= rhs) {  
        int mid = (lhs + rhs) / 2;  
  
        if (arr[mid] == key) {  
            return mid;  
        } else if (arr[mid] < key) {  
            lhs = mid + 1;  
        } else {  
            rhs = mid - 1;  
        }  
    }  
    return -1;  
}
```



key
7

rhs lhs
4 5



Analyzing the Algorithms

For Comparison

```
private int linearSearch(int[] arr,
                        int key) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == key)
            return i;
    }

    return -1;
}
```

```
private int binarySearch(int[] arr,
                        int key) {
    int lhs = 0;
    int rhs = arr.length - 1;

    while (lhs <= rhs) {
        int mid = (lhs + rhs) / 2;

        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            lhs = mid + 1;
        else
            rhs = mid - 1;
    }

    return -1;
}
```

Analyzing Linear Search

- How many elements of the array do we have to look at to do a linear search?
- Let's suppose that there are N elements in the array.
- We may have to look at each of them once.
- Number of lookups: N .

Analyzing Binary Search

- How many elements of the array do we have to look at to do a binary search?
- Let's suppose that there are N elements in the array.
- Each lookup cuts the size of the array in half.
- How many times can we cut the array in half before we run out of elements?

Slicing and Dicing

- After zero lookups: N
- After one lookup: $N / 2$
- After two lookups: $N / 4$
- After three lookups: $N / 8$
- ...
- After k lookups: $N / 2^k$

Cutting in Half

- After doing k lookups, there are $N / 2^k$ elements left.
- The algorithm stops when there is just one element left.

- Solving for the number of iterations:

$$N / 2^k = 1$$

$$N = 2^k$$

$$\log_2 N = k$$

- So binary search stops after **$\log_2 N$** lookups.

For Comparison

N	$\log_2 N$
10	3
100	7
1000	10
1,000,000	20
1,000,000,000	30

Binary search can check whether a value exists in an array of one billion elements in just 30 array accesses!

A Feel for $\log_2 N$

- It is conjectured that the number of atoms in the universe is 10^{100} .
- $\log_2 10^{100}$ is roughly 300.
- If you (somehow) listed all the atoms in the universe in sorted order, you would need to look at 300 before you found the one you were looking for.

Sorting

Bubble Sort

- Until the array is sorted:
 - Look at each adjacent pair of elements.
 - If they are out of order, swap them.

Should we use bubble sort?

Should we use bubble sort?

Let's ask the President of the United States!



A Presidential Decree



- Bubble sort is the wrong way to go.
- What other algorithms could we use instead?

Selection Sort

- Find the smallest number and swap it to the front of the array.
- Find the second-smallest number and swap it to the second position.
- Find the third-smallest number and swap it to the third position.
- ...

```
private void selectionSort(int[] array) {
    for (int index = 0; index < array.length; index++) {
        int smallestIndex = findSmallest(array, index);
        swap(array, index, smallestIndex);
    }
}

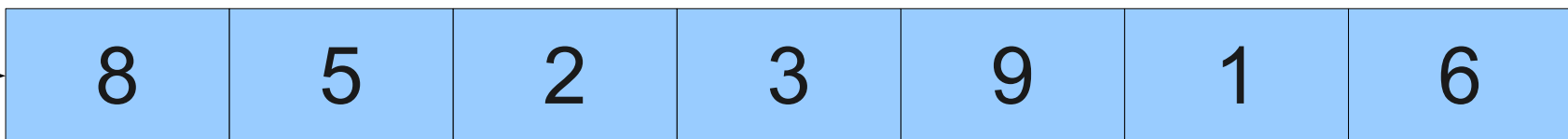
private int findSmallest(int[] array, int startPoint) {
    int smallestIndex = startPoint;
    for (int i = startPoint + 1; i < array.length; i++) {
        if (array[i] < array[smallestIndex])
            smallestIndex = i;
    }
    return smallestIndex;
}

private void swap(int[] array, int i, int j) {
    int temp = array[i];
    array[i] = array[j];
    array[j] = temp;
}
```


Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

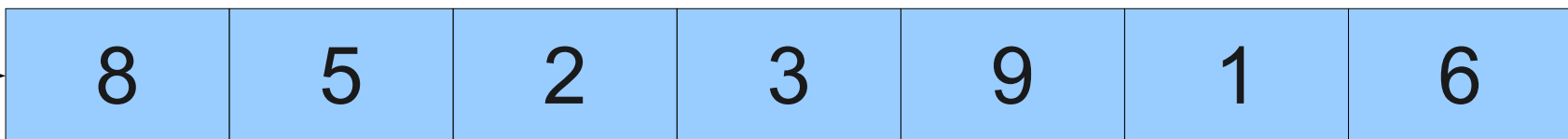
array



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

0

8

5

2

3

9

1

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

0

8

5

2

3

9

1

6

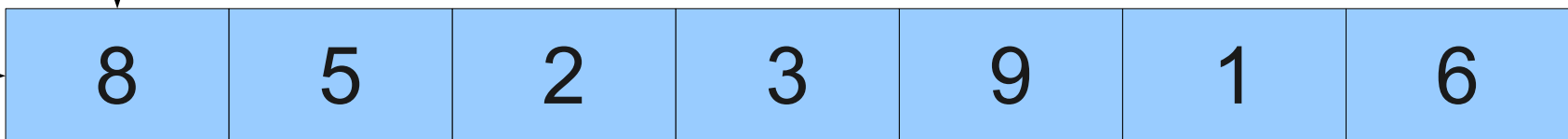
Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



startPoint



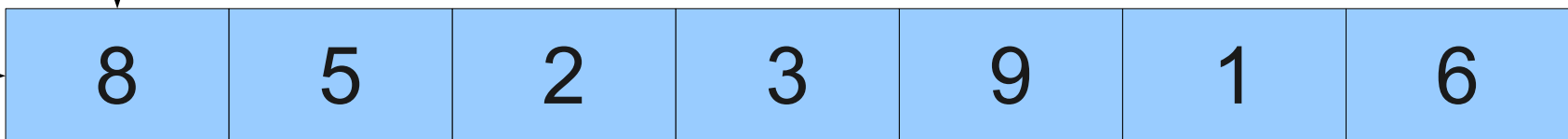
Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



startPoint



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

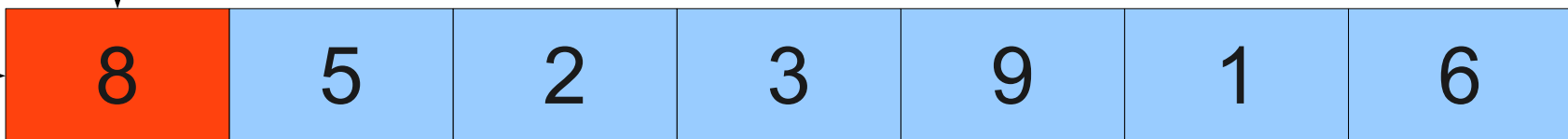
array



startPoint



smallestIndex



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

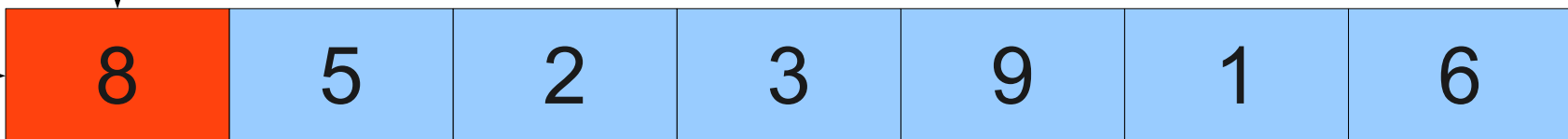
array



startPoint



smallestIndex



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



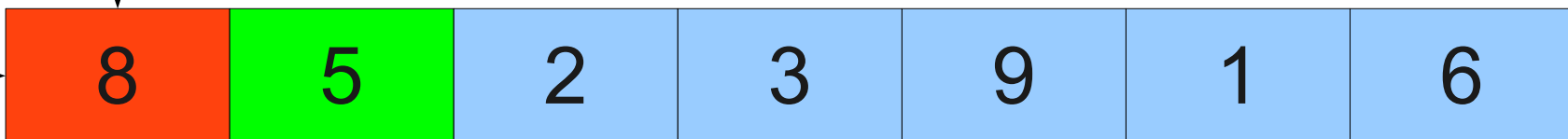
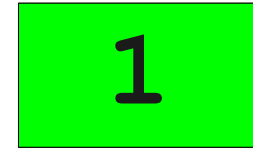
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



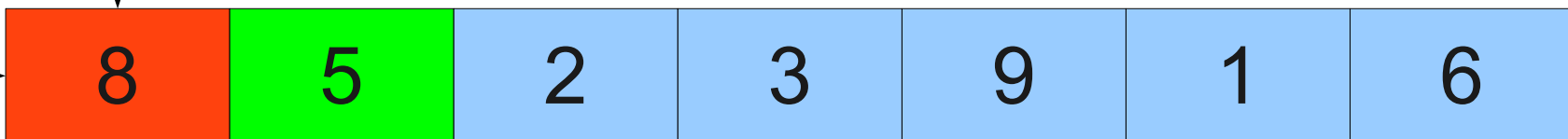
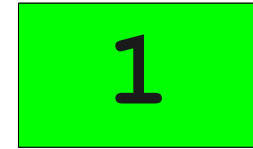
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



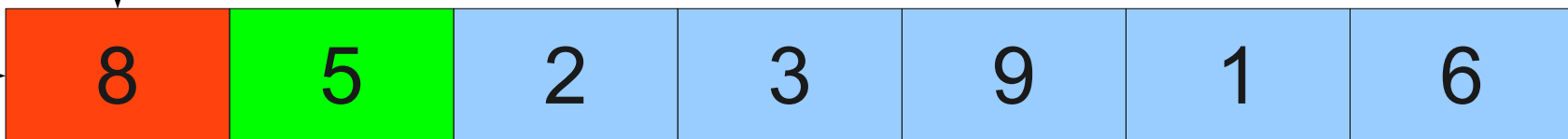
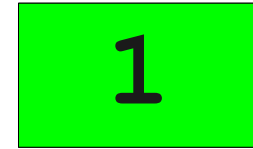
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



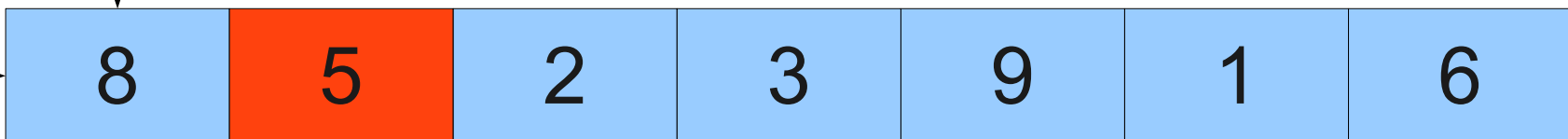
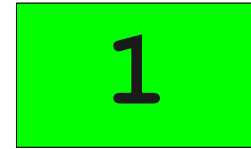
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



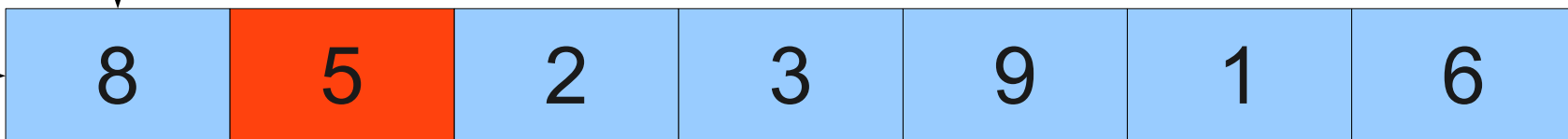
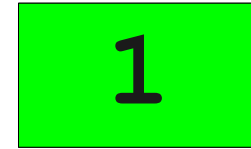
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



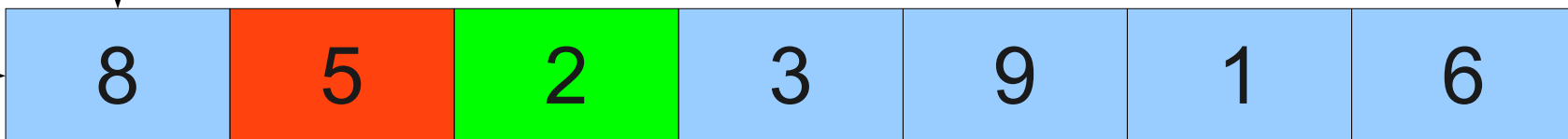
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



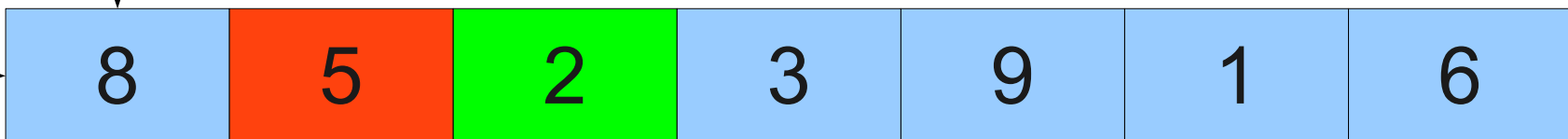
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



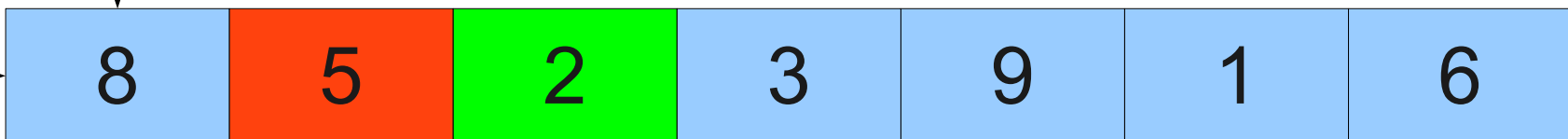
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



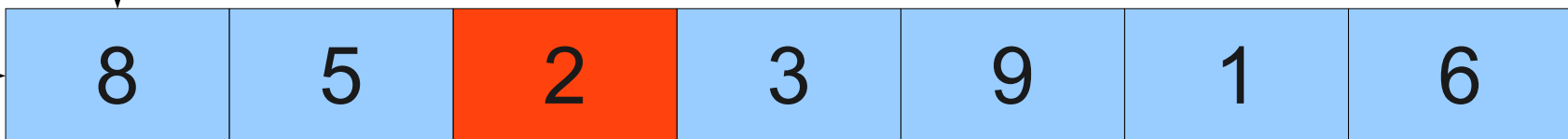
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



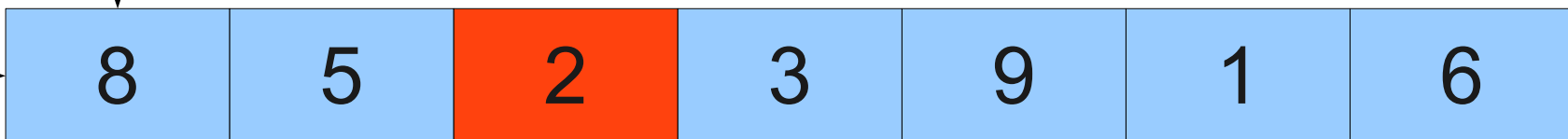
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



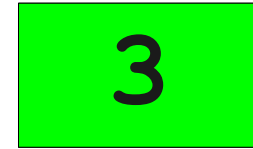
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



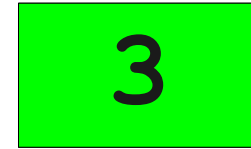
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



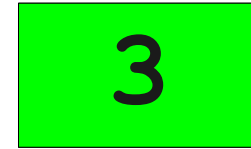
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



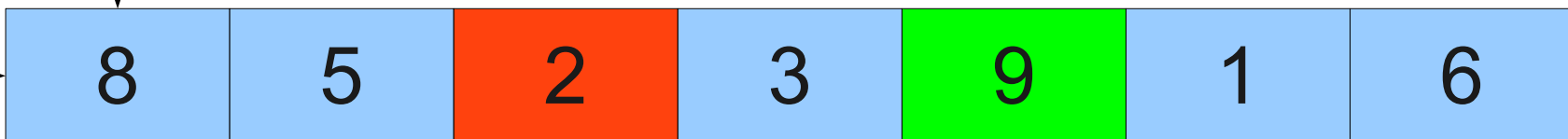
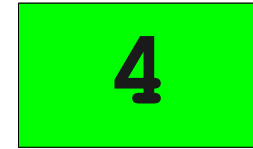
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



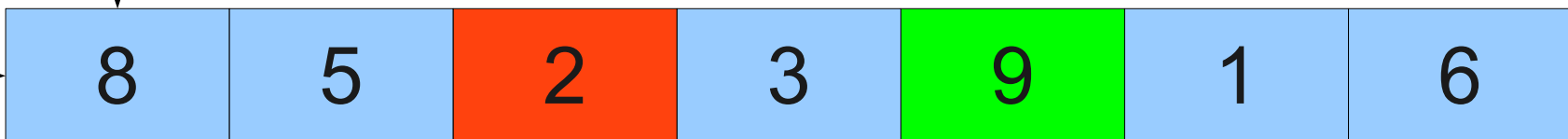
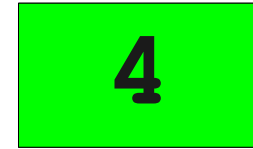
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



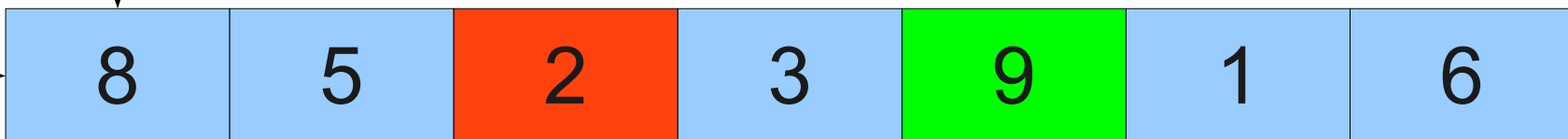
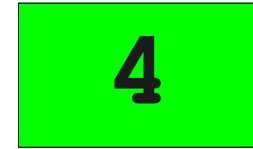
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



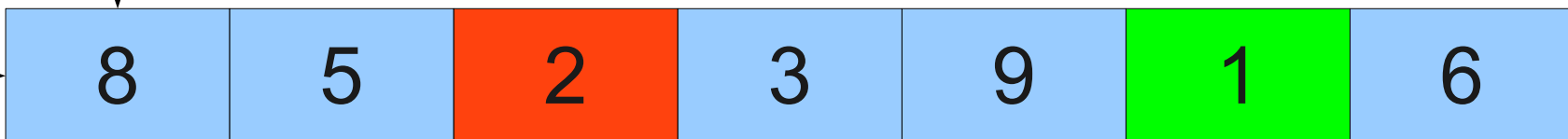
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



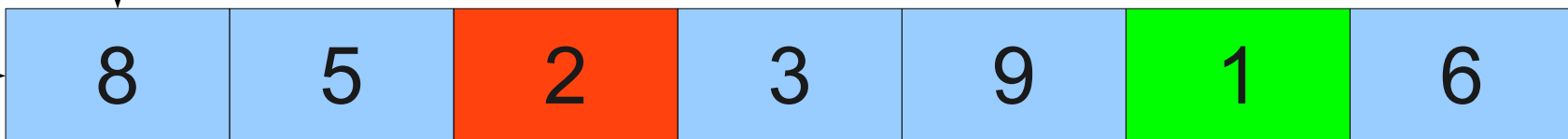
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



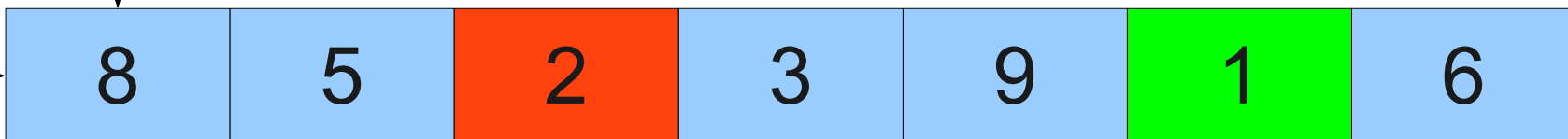
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



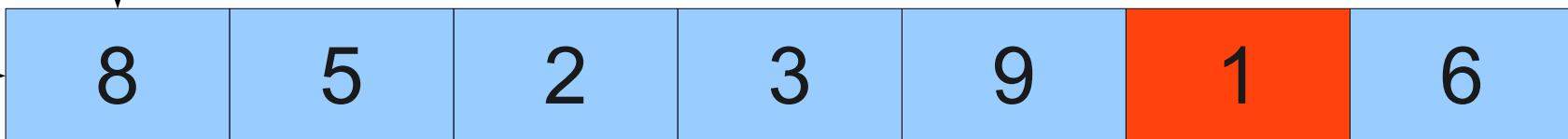
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



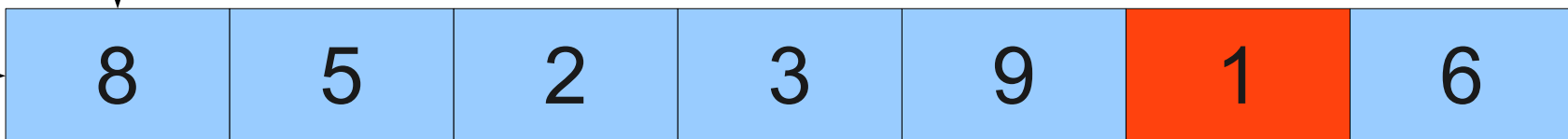
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



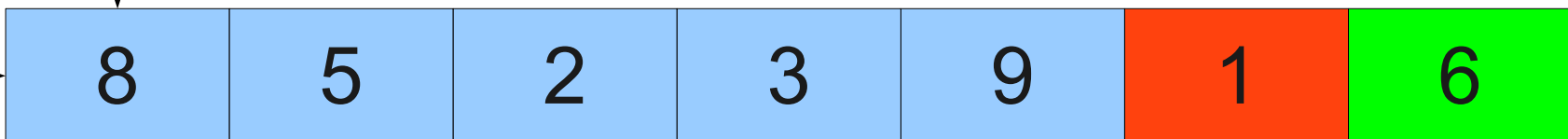
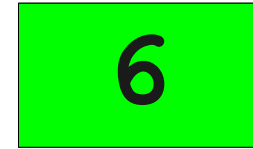
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



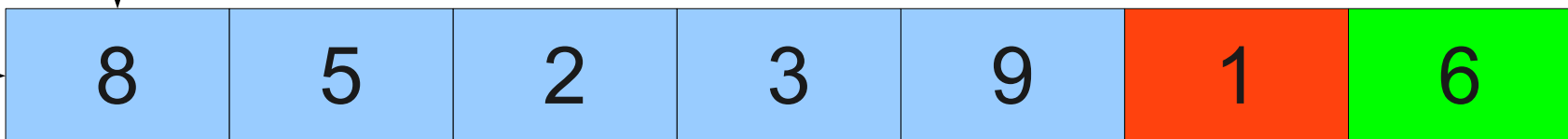
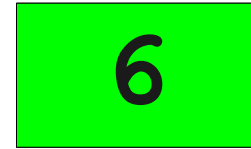
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



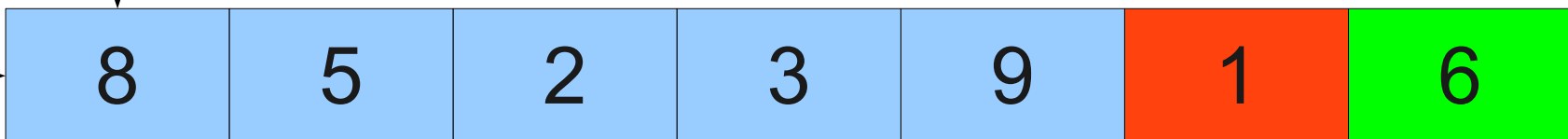
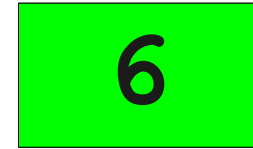
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



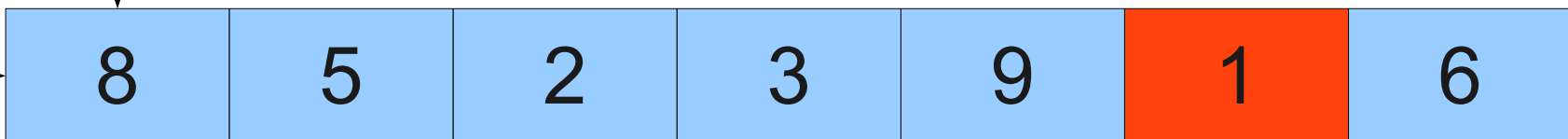
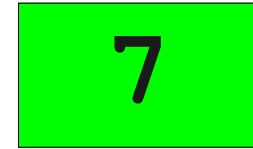
startPoint



smallestIndex



i



Selection Sort in Action

```
private int findSmallest(int[] array, int startPoint) {  
    int smallestIndex = startPoint;  
    for (int i = startPoint + 1; i < array.length; i++) {  
        if (array[i] < array[smallestIndex])  
            smallestIndex = i;  
    }  
    return smallestIndex;  
}
```

array



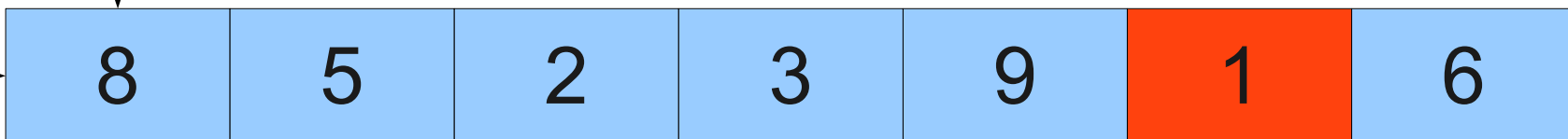
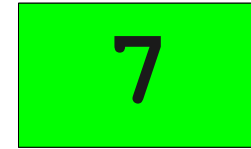
startPoint



smallestIndex



i



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

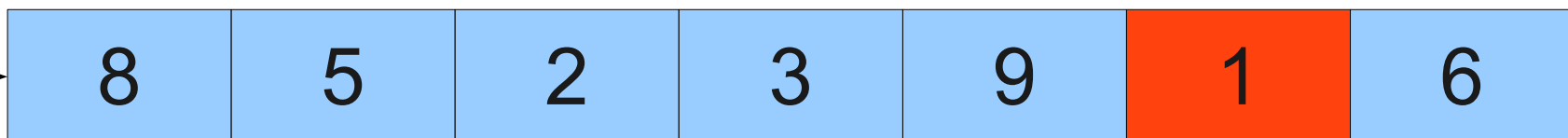
array	index	smallestIndex
	0	5

8	5	2	3	9	1	6
---	---	---	---	---	---	---

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

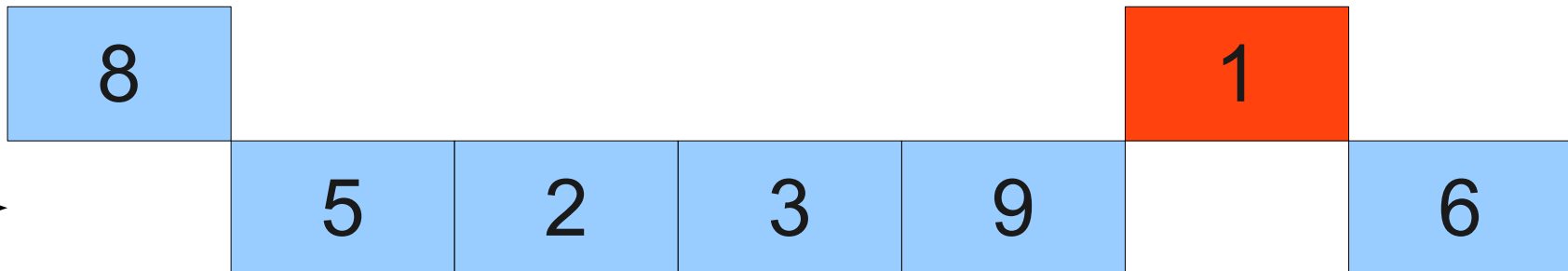
array	index	smallestIndex
	0	5



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

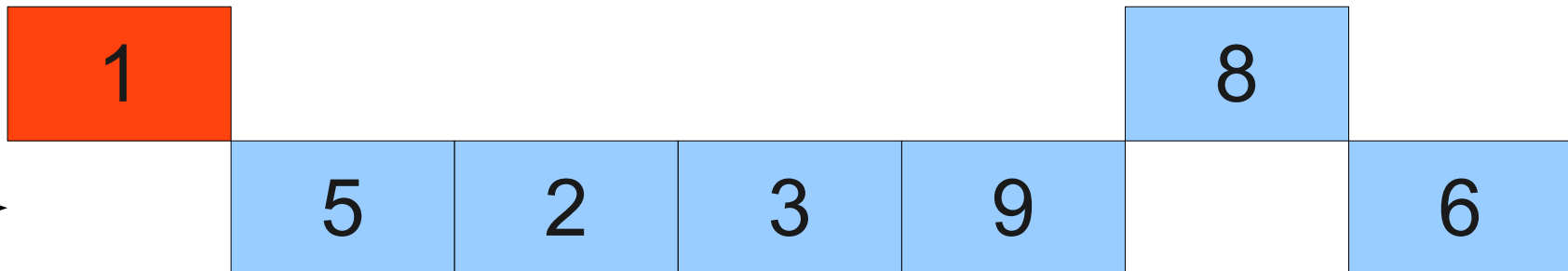
array	index	smallestIndex
	0	5



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

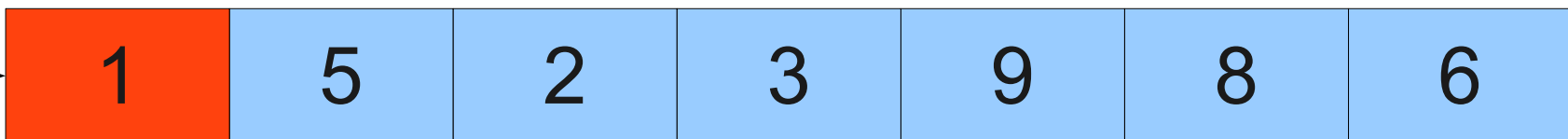
array	index	smallestIndex
	0	5



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array	index	smallestIndex
	0	5



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

0

1

5

2

3

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

0

1

5

2

3

9

8

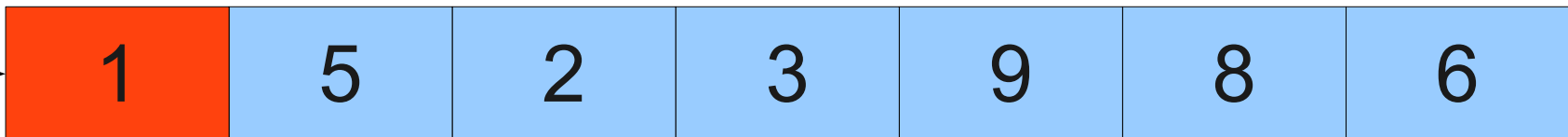
6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

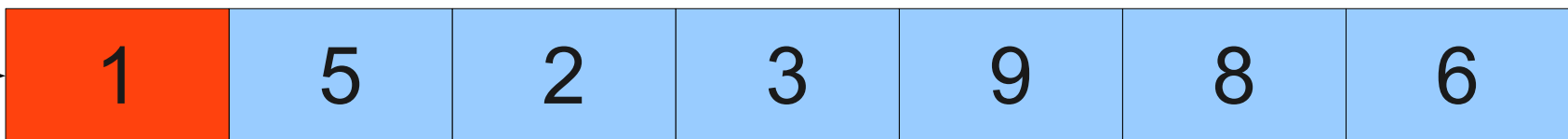


Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

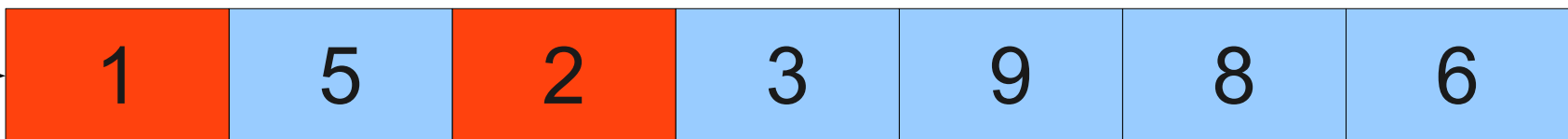


Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

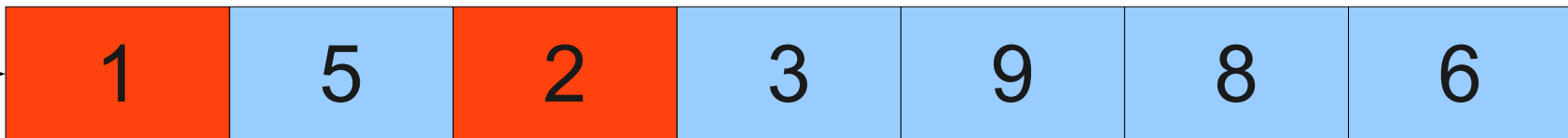
index



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

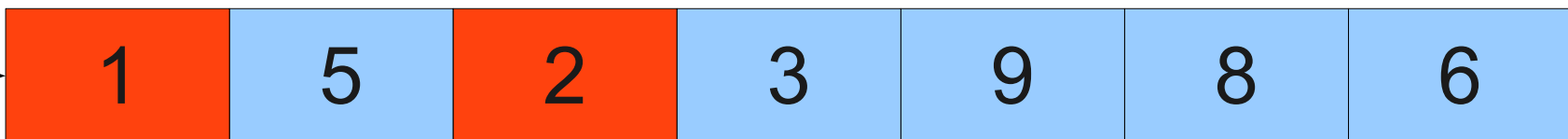
array	index	smallestIndex
	1	2



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array	index	smallestIndex
	1	2



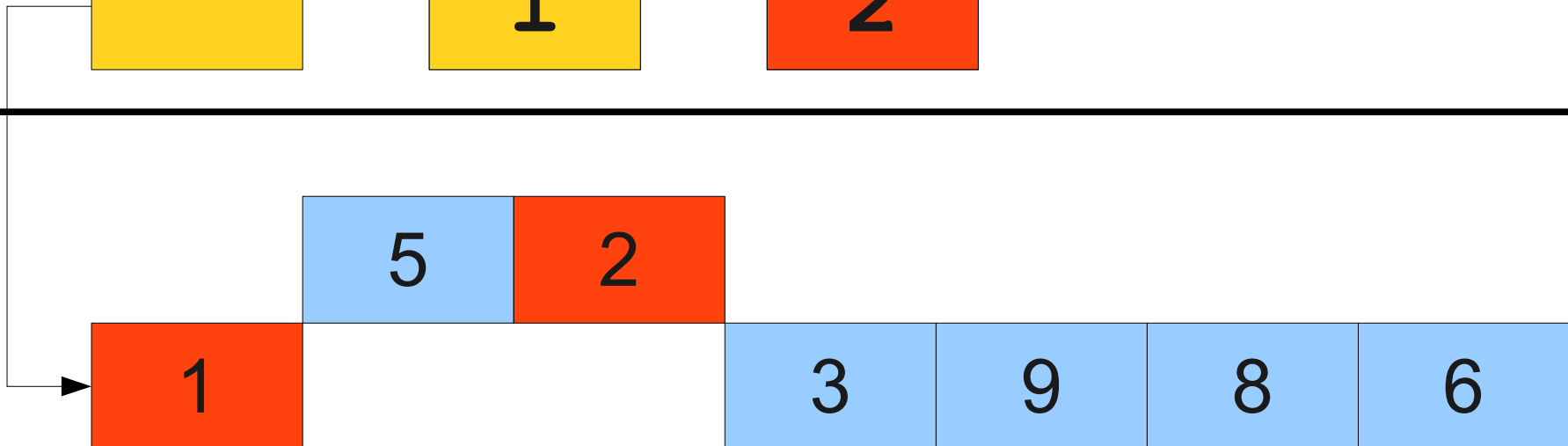
Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array index smallestIndex



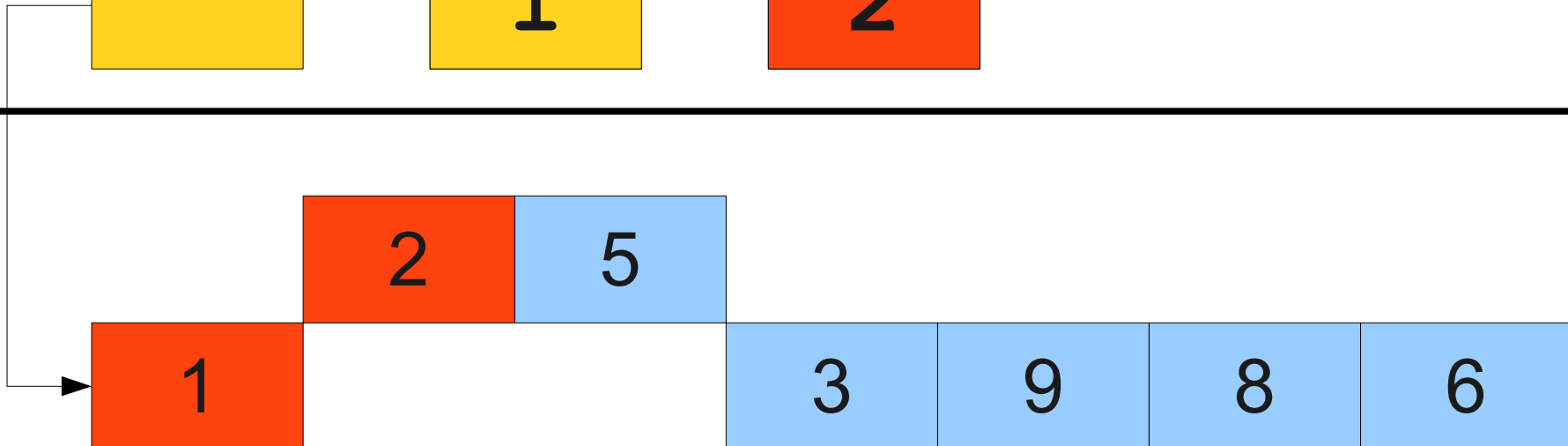
A diagram showing three variables: 'array' in a yellow box, 'index' in a yellow box containing the number '1', and 'smallestIndex' in a red box containing the number '2'.



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

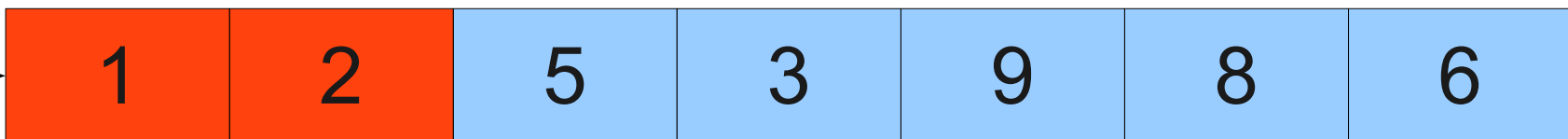
array	index	smallestIndex
	1	2



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array	index	smallestIndex
	1	2



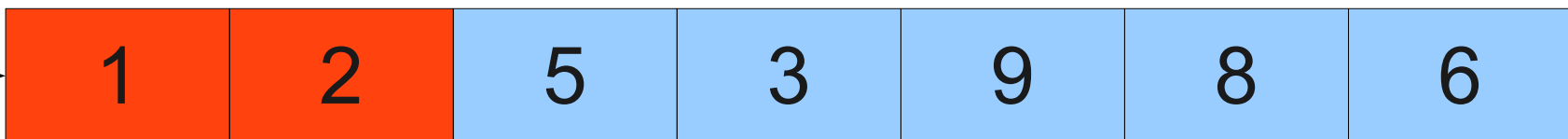
Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

1

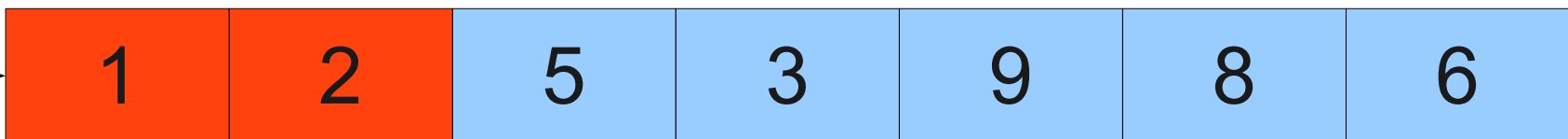


Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

2

1

2

5

3

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

2

1

2

5

3

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

2

1

2

5

3

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array	index	smallestIndex
	2	3

1	2	5	3	9	8	6
---	---	---	---	---	---	---

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array index smallestIndex

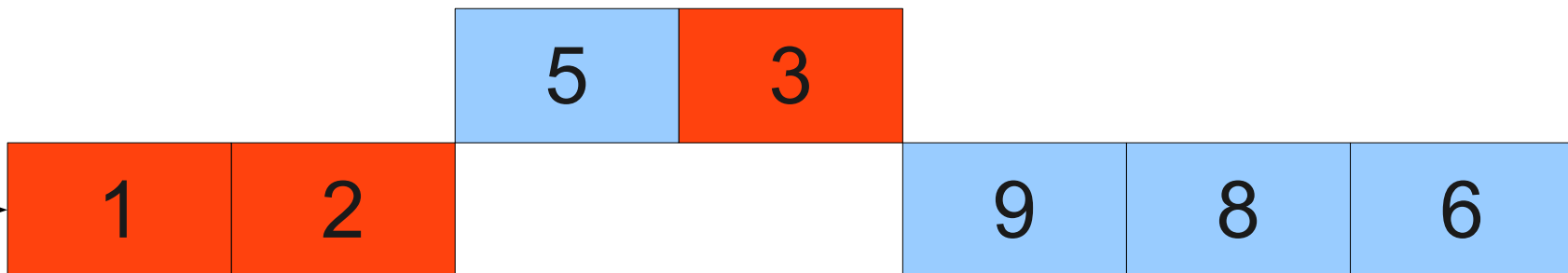


1 2 5 3 9 8 6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

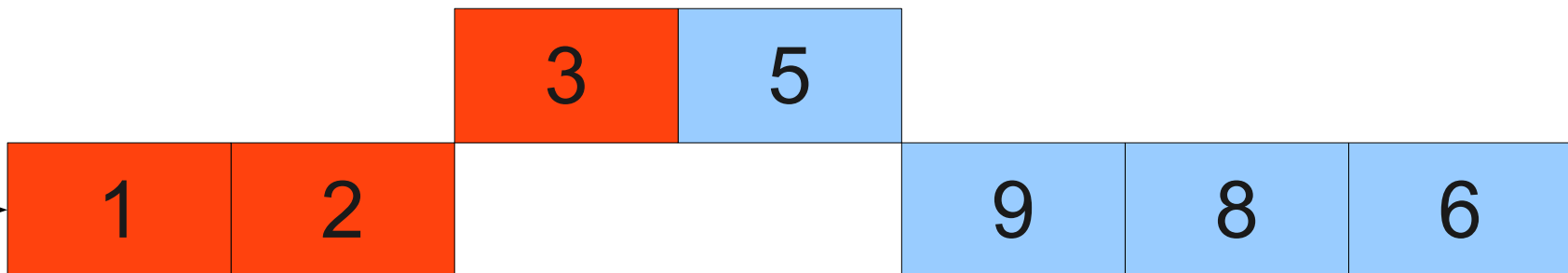
array index smallestIndex



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

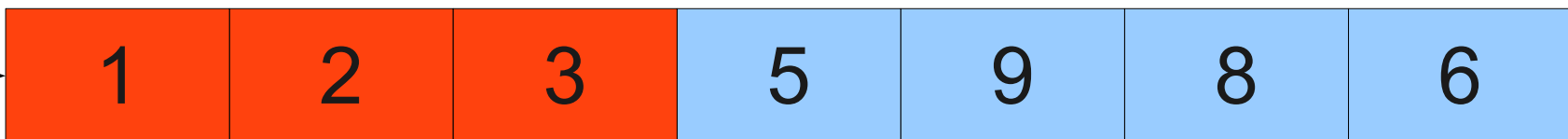
array index smallestIndex



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array	index	smallestIndex
	2	3



Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

3

1

2

3

5

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

3

1

2

3

5

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

4

1

2

3

5

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

4

1

2

3

5

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

4

1

2

3

5

9

8

6

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

4

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

4

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

5

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

5

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

6

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

6

1

2

3

5

6

8

9

Selection Sort in Action

```
private void selectionSort(int[] array) {  
    for (int index = 0; index < array.length; index++) {  
        int smallestIndex = findSmallest(array, index);  
        swap(array, index, smallestIndex);  
    }  
}
```

array

index

7

1

2

3

5

6

8

9

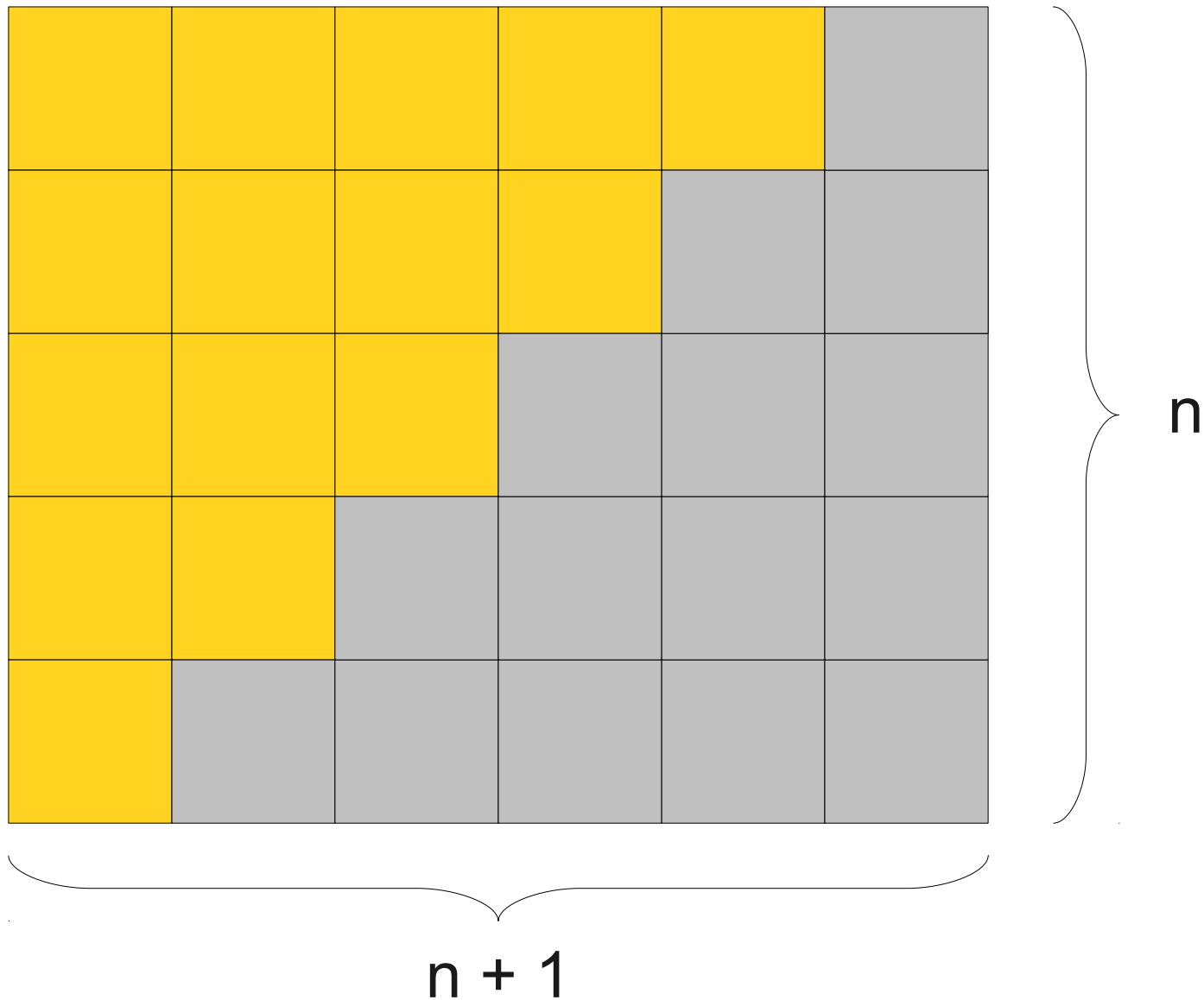
Selection Sort in Action



Analyzing Selection Sort

- How much work do we do for selection sort?
- To find the smallest value, we need to look at all N array elements.
- To find the second-smallest value, we need to look at $N - 1$ array elements.
- To find the third-smallest value, we need to look at $N - 2$ array elements.
- Work is $N + (N - 1) + (N - 2) + \dots + 1$.

$$1 + 2 + \dots + (n - 1) + n = n(n+1) / 2$$



An Interesting Observation

- Selection sort does roughly $N^2 / 2$ array lookups.
- Suppose we double the number of elements in the array we want to sort.
- How much longer will it take to sort the new array?

$$\begin{aligned} & \textit{newTime} / \textit{oldTime} \\ &= ((2N)^2 / 2) / (N^2 / 2) \\ &= (2N)^2 / N^2 \\ &= 4N^2 / N^2 \\ &= 4 \end{aligned}$$

An Interesting Observation

- Selection sort does roughly $N^2 / 2$ array lookups.
- Suppose we double the number of elements in the array we want to sort.
- How much longer will it take to sort the new array?

$$\begin{aligned} & \textit{newTime} / \textit{oldTime} \\ &= ((2N)^2 / 2) / (N^2 / 2) \\ &= (2N)^2 / N^2 \\ &= 4N^2 / N^2 \\ &= 4 \end{aligned}$$

- So we should expect it to take about four times longer.

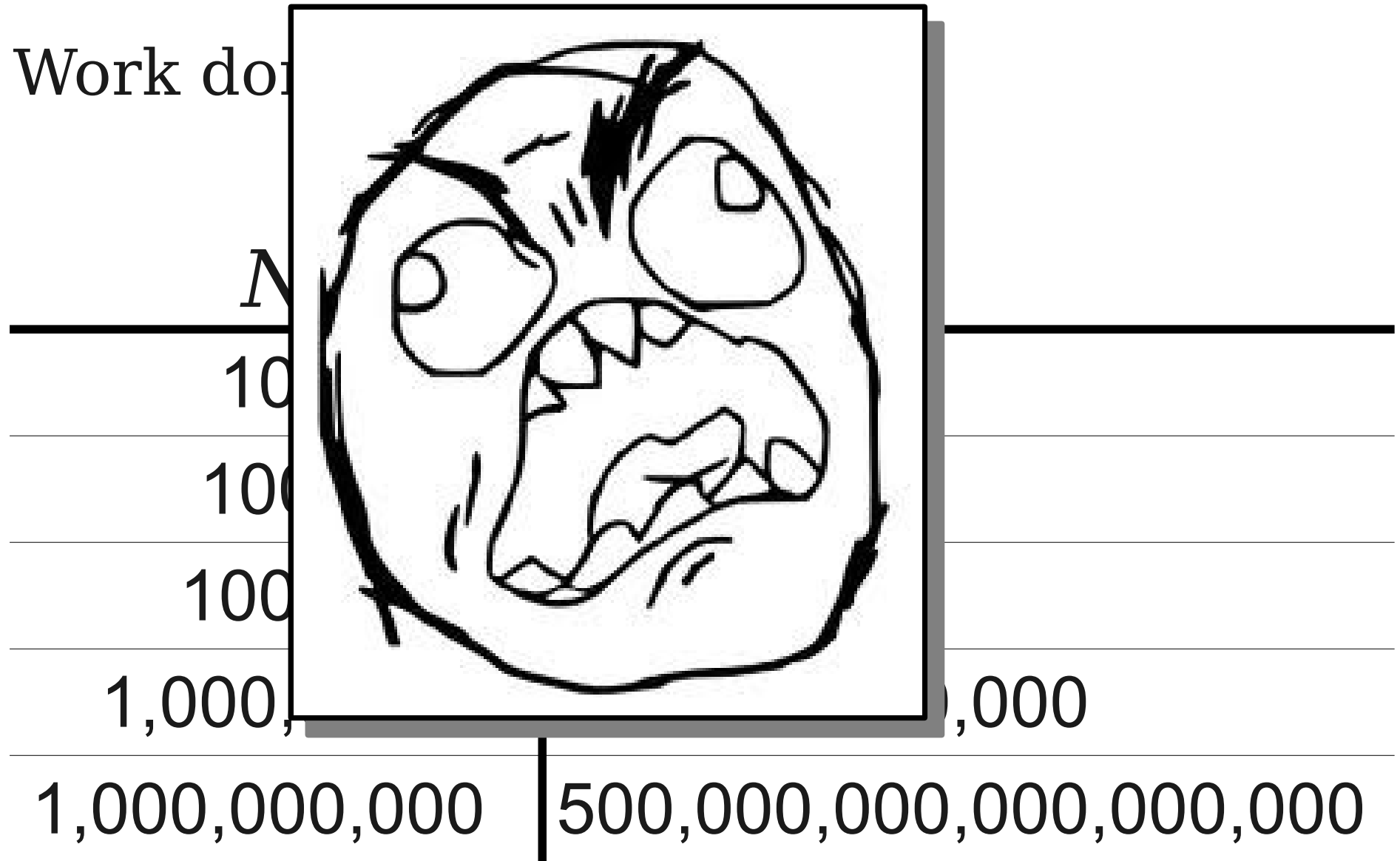
Analyzing Selection Sort

- Work done is roughly $N^2 / 2$.

N	$N^2 / 2$
10	50
100	5,000
1000	500,000
1,000,000	500,000,000,000
1,000,000,000	500,000,000,000,000,000

Analyzing Selection Sort

- Work done



A Few Other Sorting Algorithms