Efficiency

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, seeing if a genome sequence is in a database, 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?

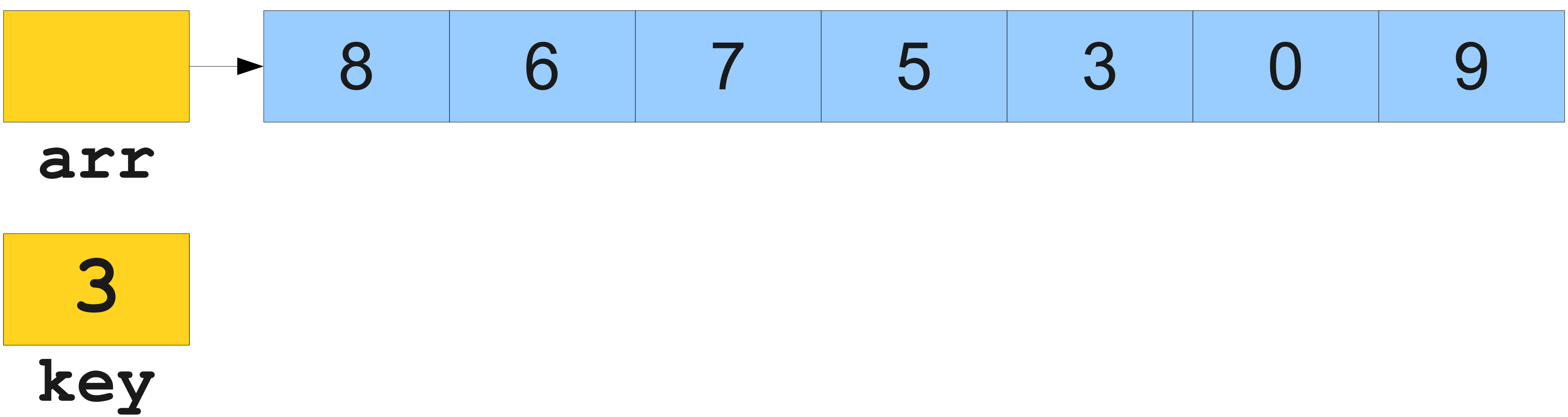
Linear Search

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;
}
```

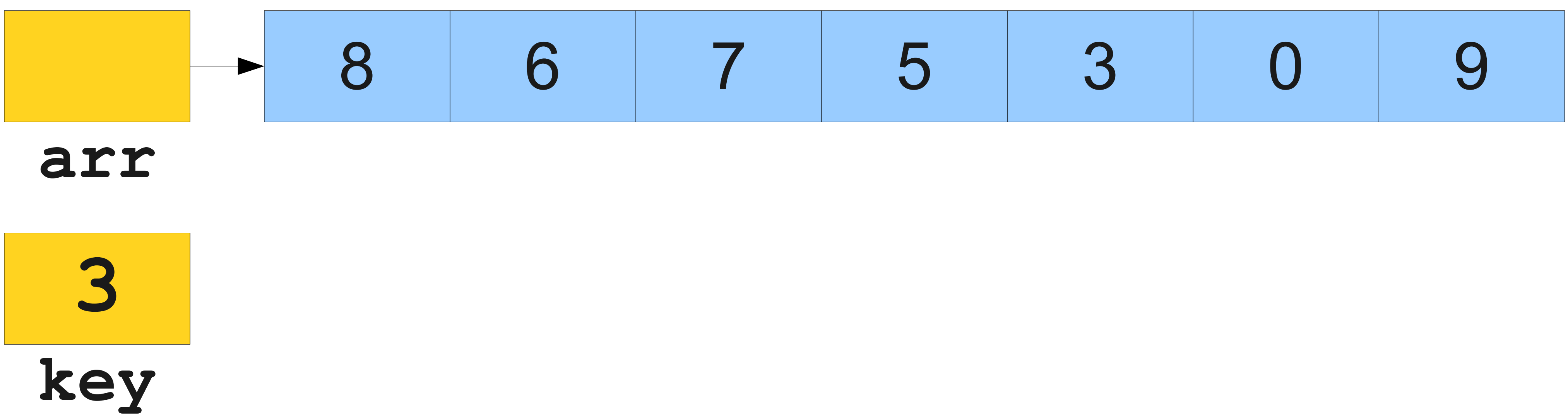
Linear Search

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

    return -1;
}
```



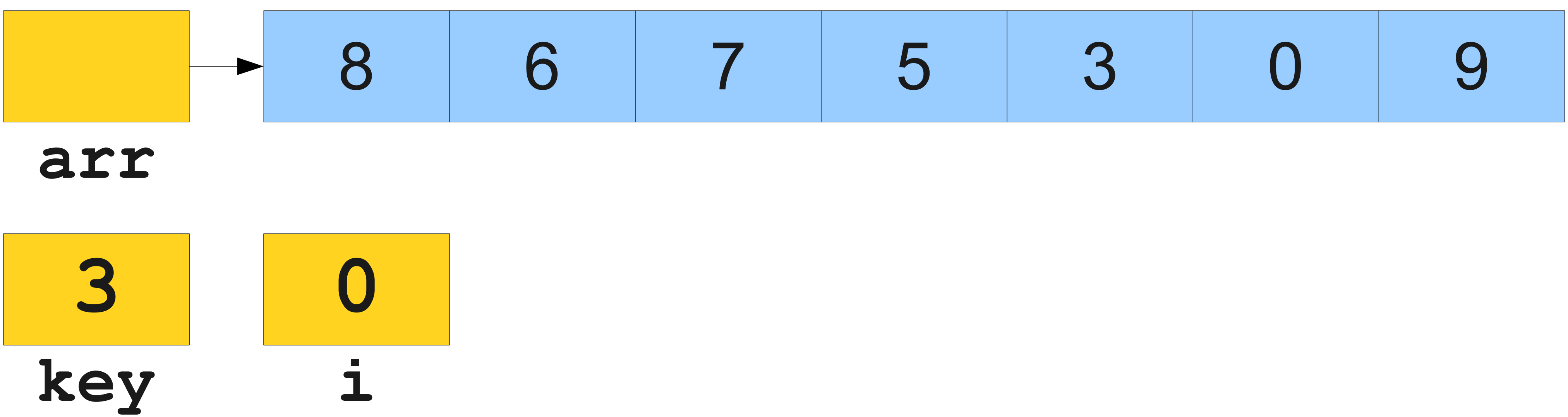
Linear Search

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

    return -1;
}
```



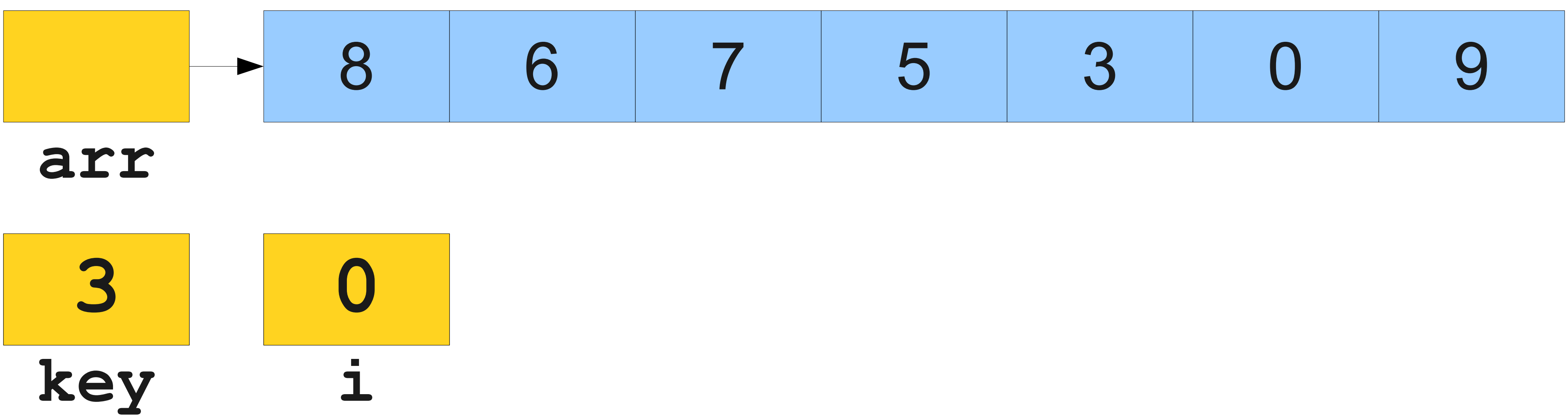
Linear Search

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

    return -1;
}
```



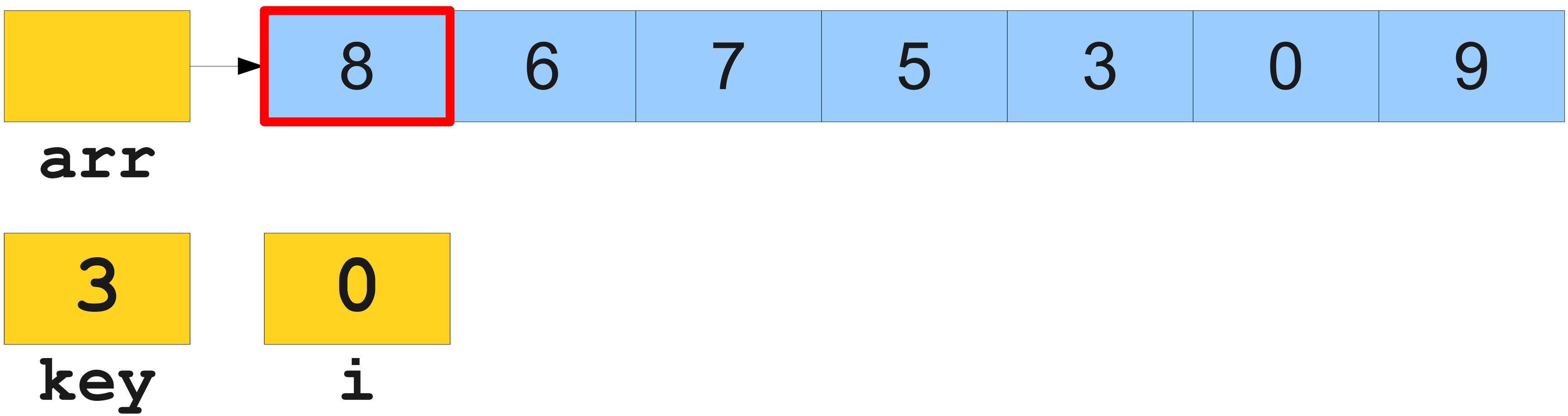
Linear Search

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

    return -1;
}
```



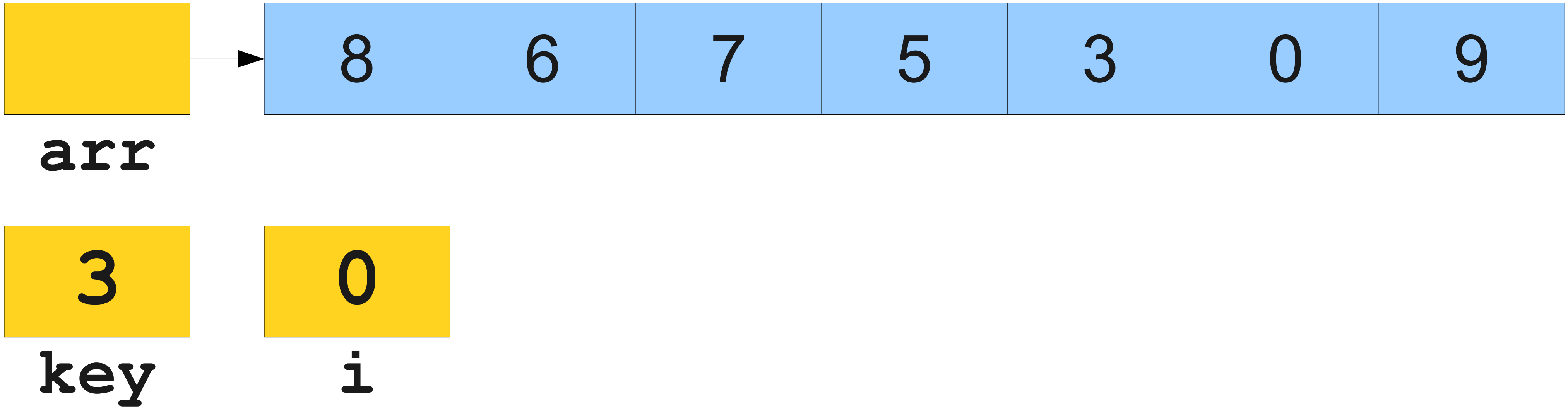
Linear Search

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

    return -1;
}
```



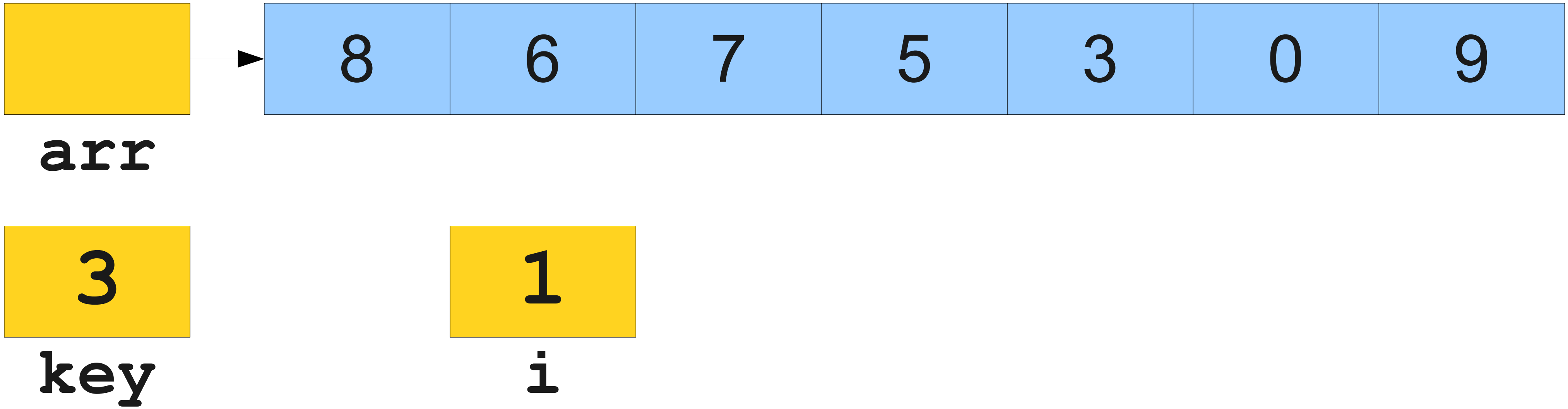
Linear Search

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

    return -1;
}
```



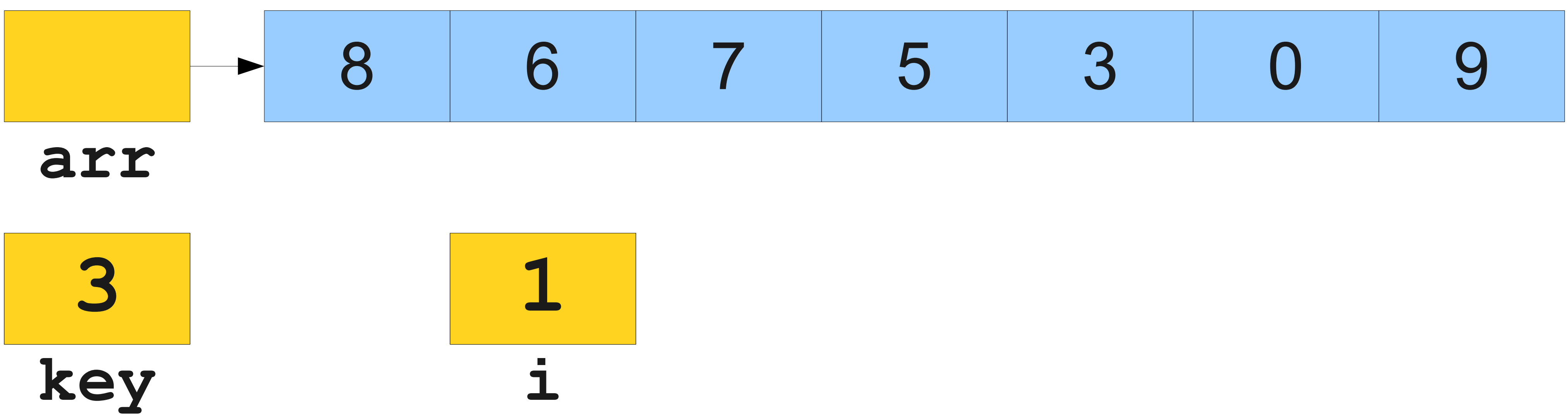
Linear Search

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

    return -1;
}
```



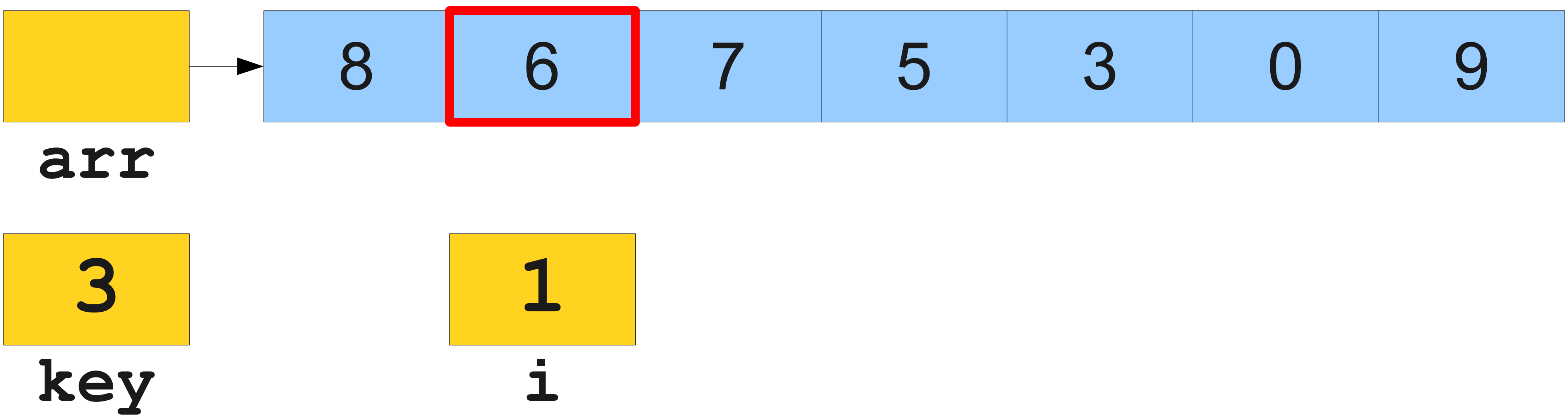
Linear Search

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

    return -1;
}
```



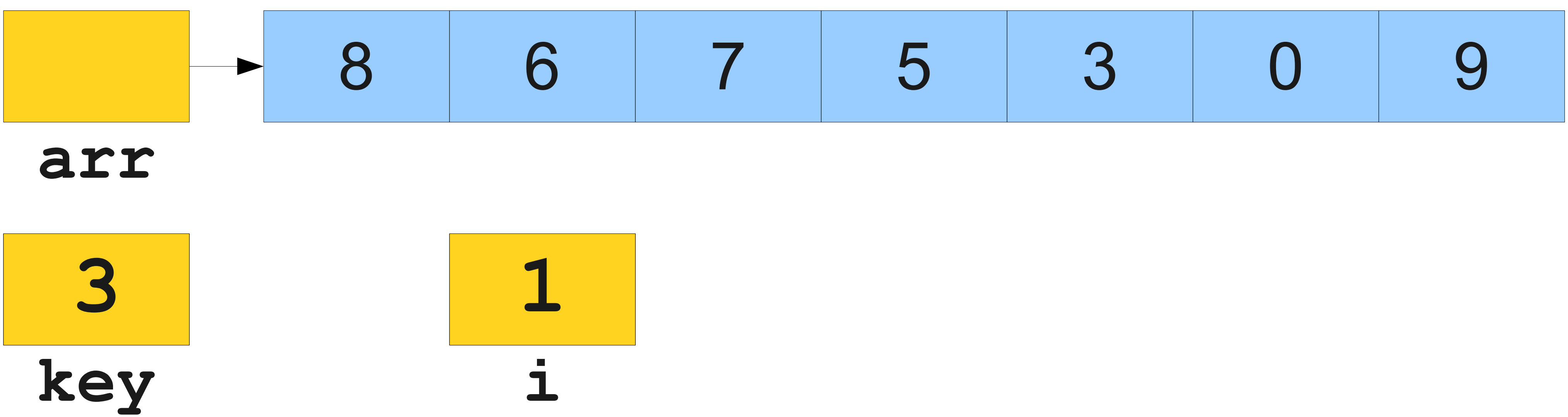
Linear Search

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

    return -1;
}
```



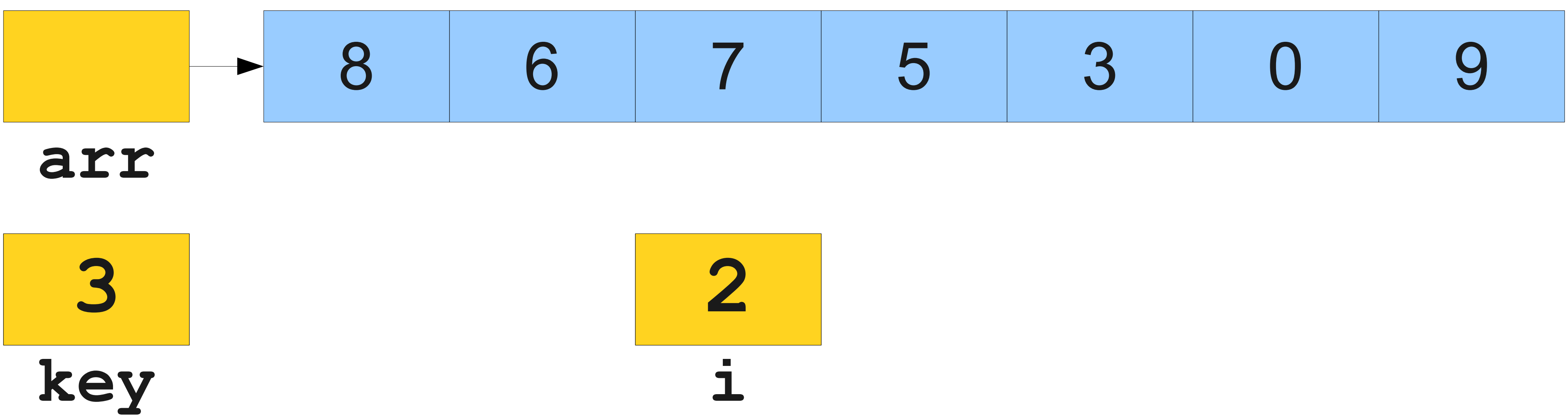
Linear Search

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

    return -1;
}
```



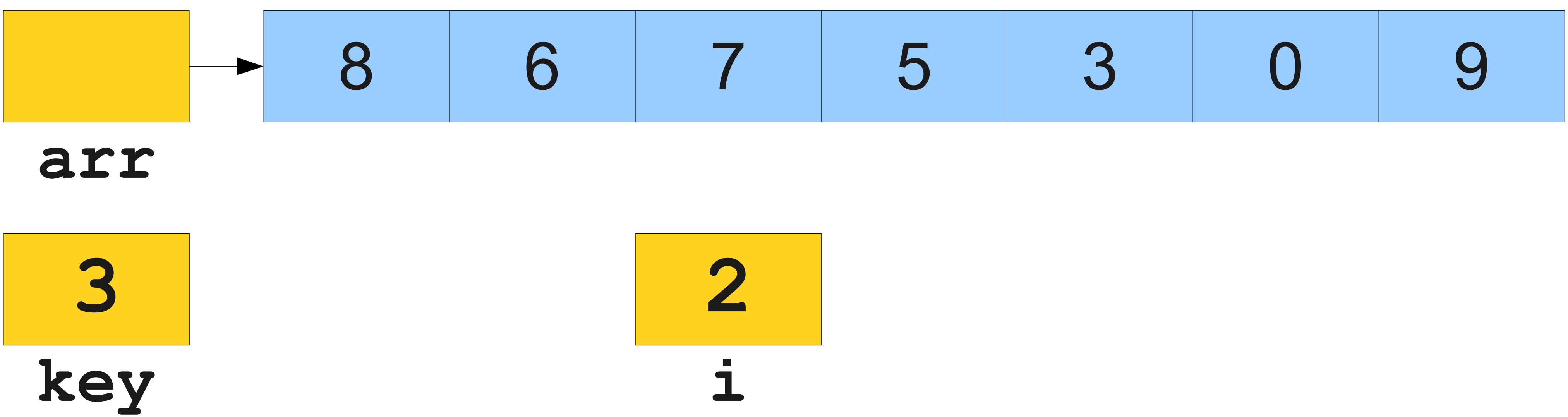
Linear Search

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

    return -1;
}
```



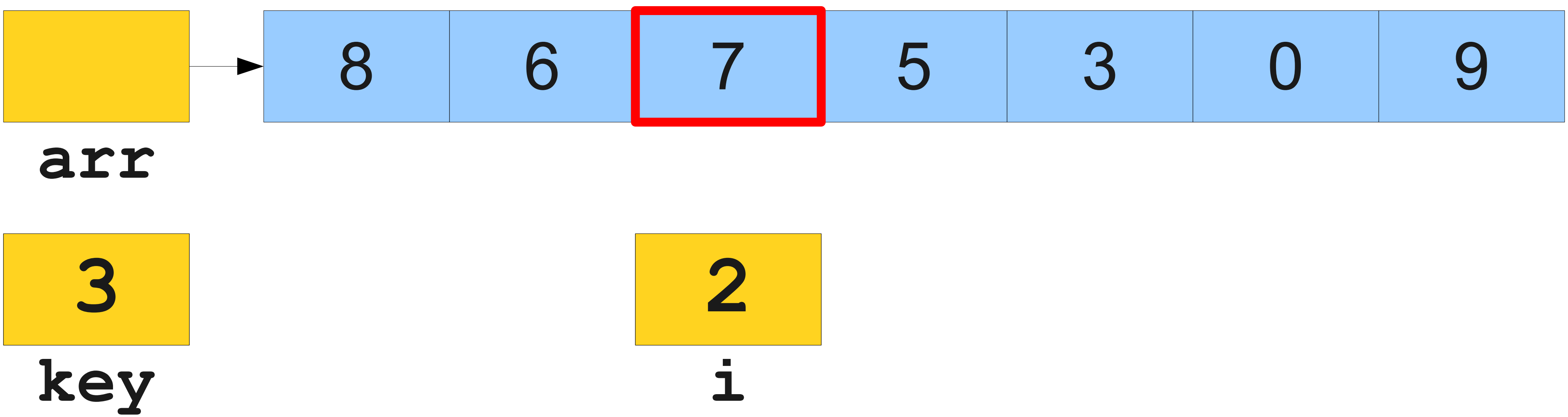
Linear Search

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

    return -1;
}
```



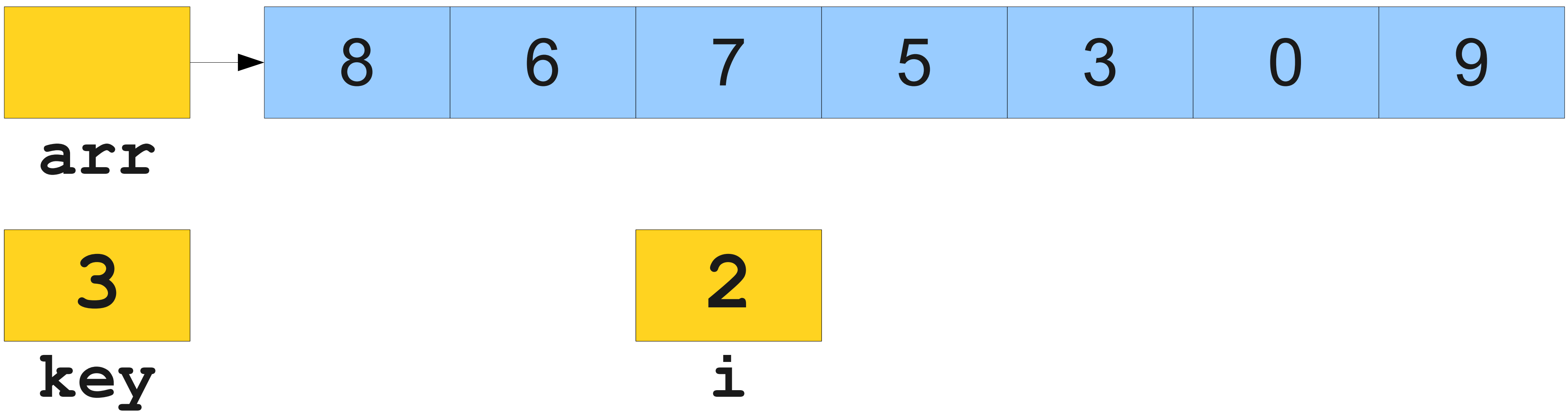
Linear Search

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

    return -1;
}
```



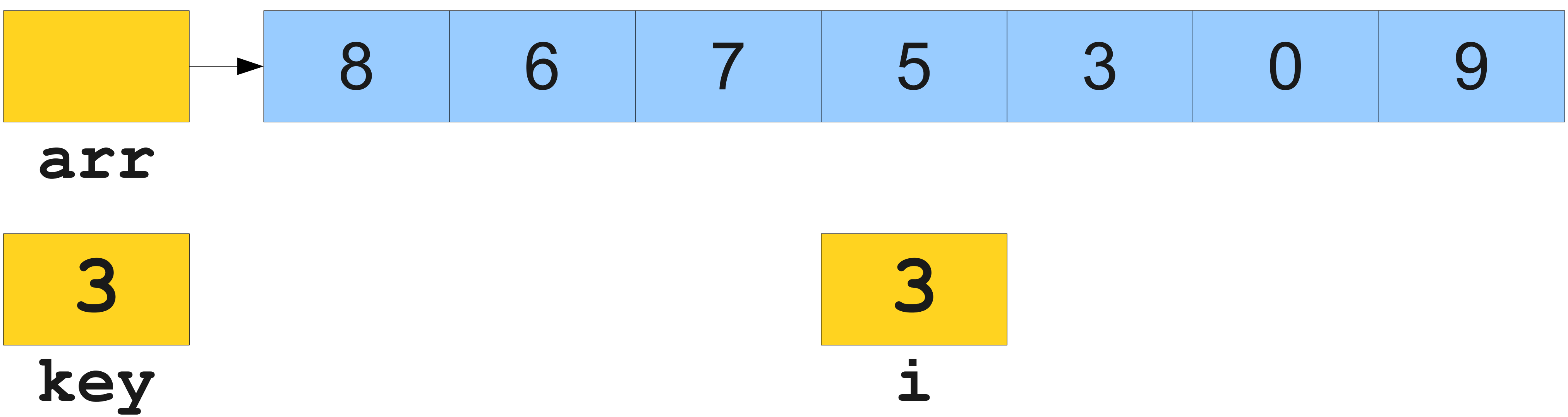
Linear Search

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

    return -1;
}
```



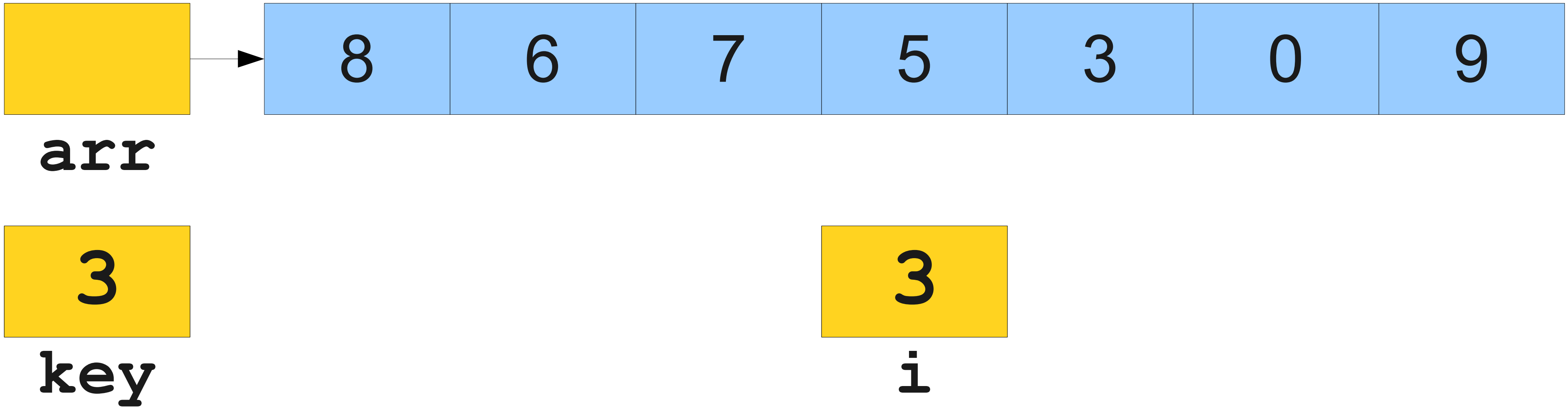
Linear Search

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

    return -1;
}
```



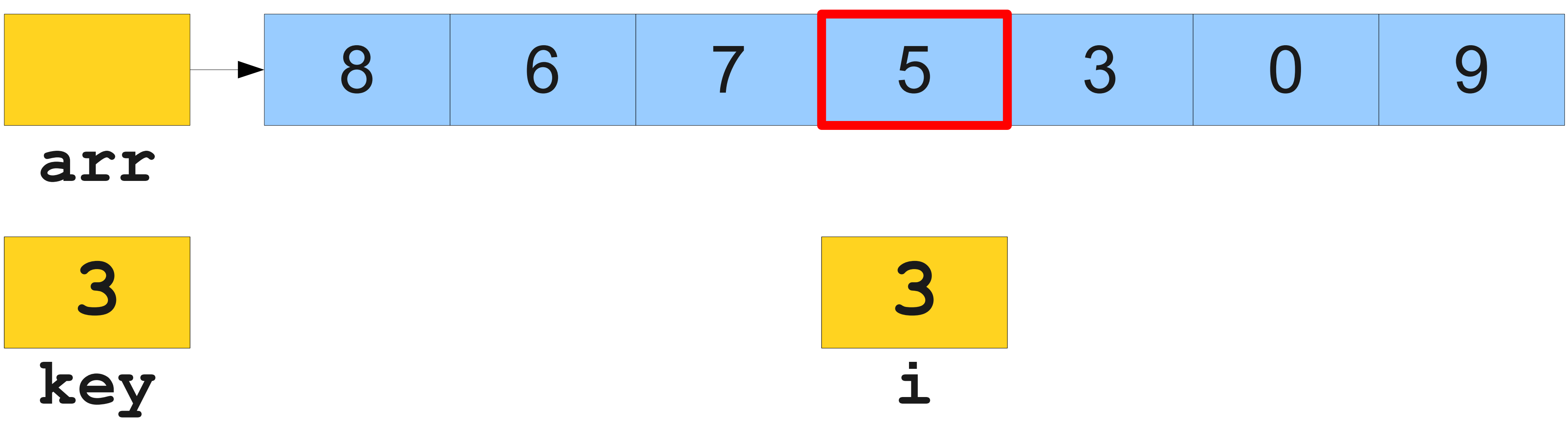
Linear Search

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

    return -1;
}
```



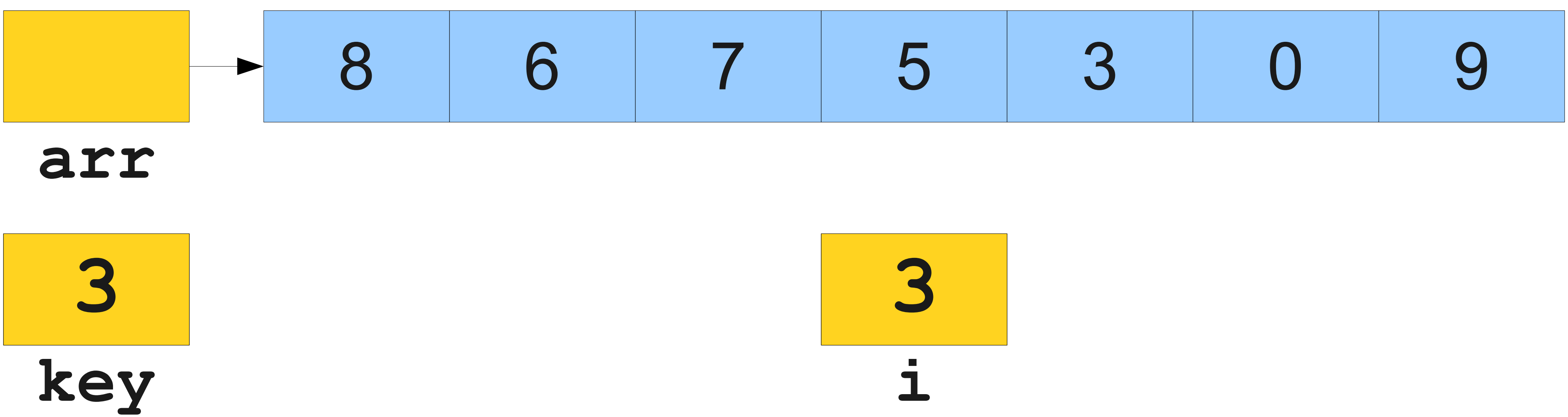
Linear Search

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

    return -1;
}
```



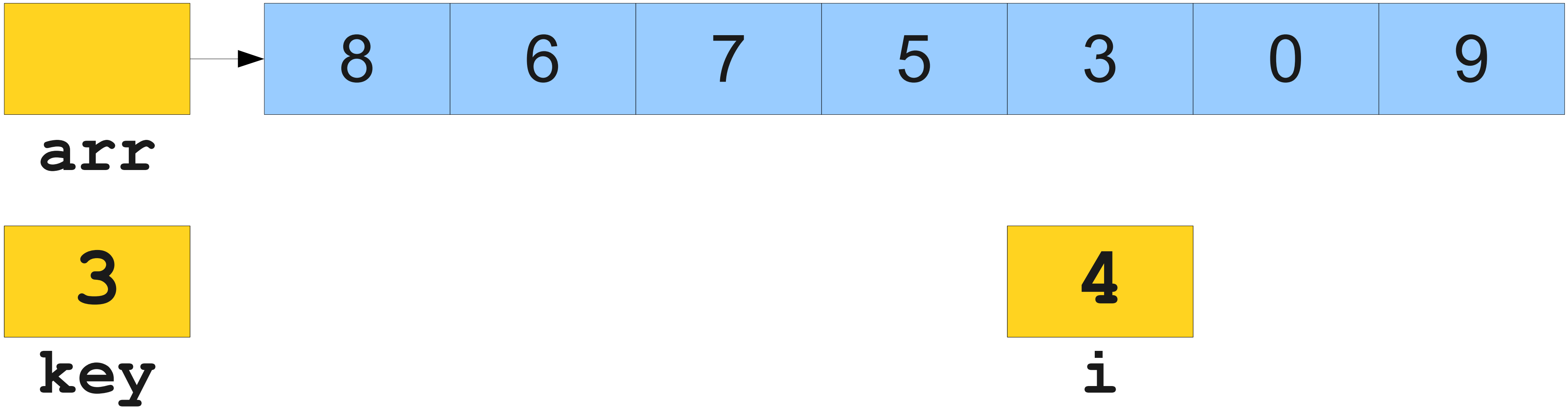
Linear Search

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

    return -1;
}
```



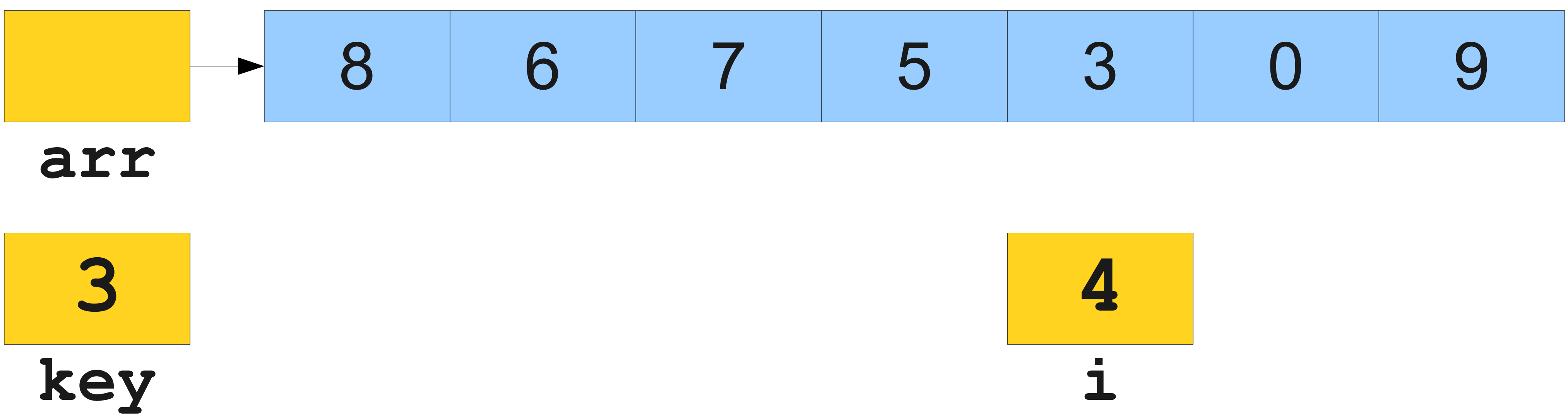
Linear Search

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

    return -1;
}
```



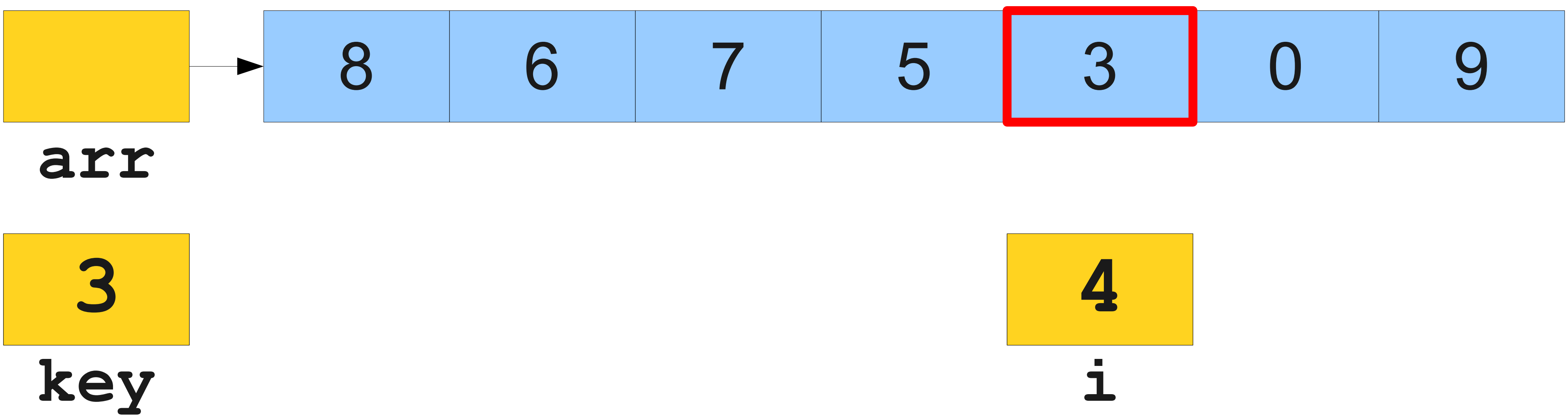
Linear Search

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

    return -1;
}
```



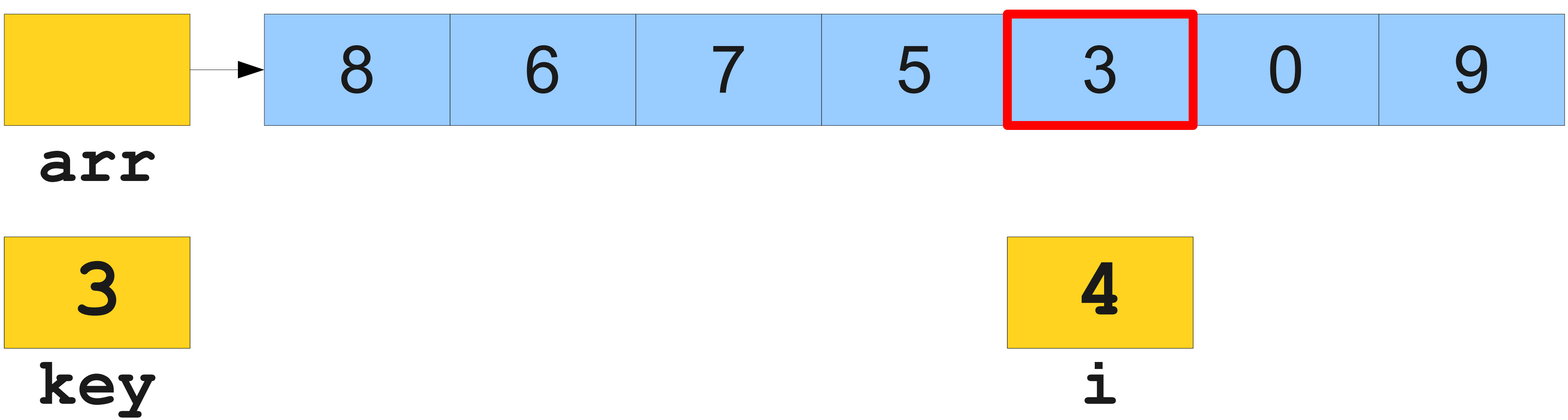
Linear Search

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

    return -1;
}
```



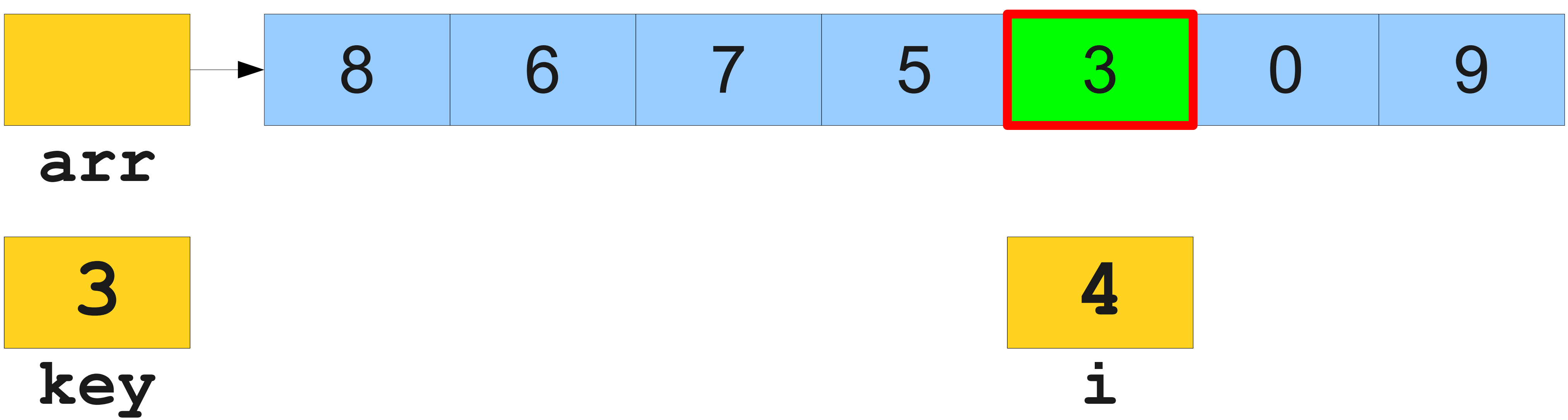
Linear Search

```
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;
    }
}
```

5	3	0	9

arr

3

key

4

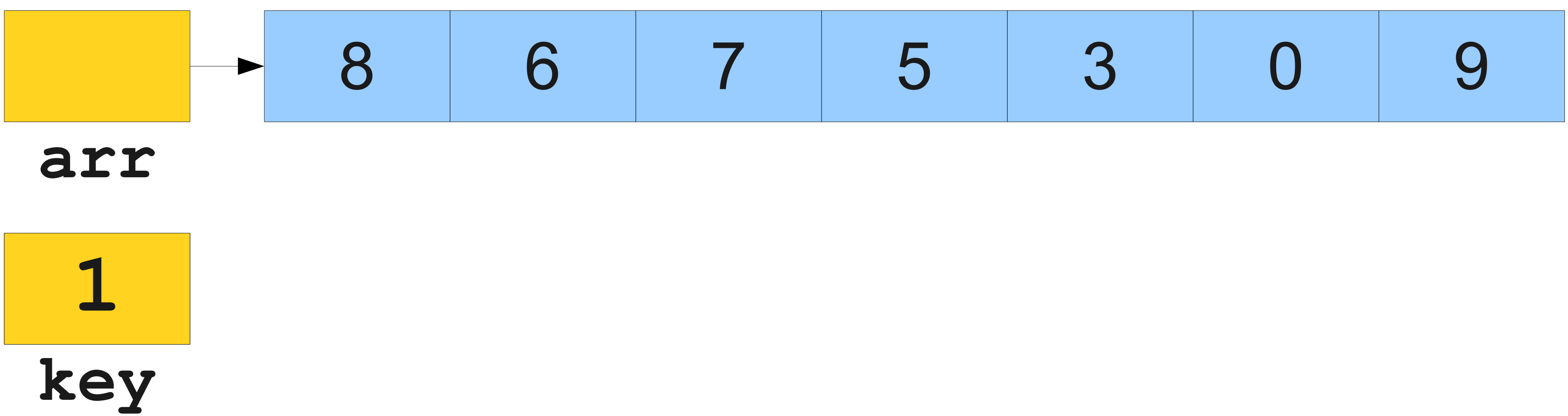
i

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;
}
```

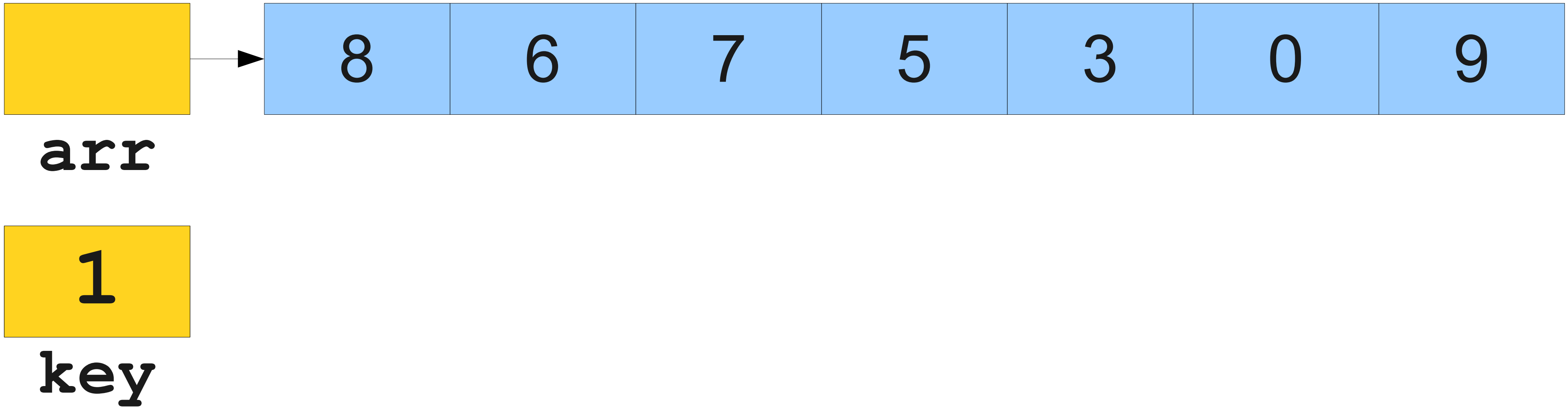
Linear Search

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

    return -1;
}
```



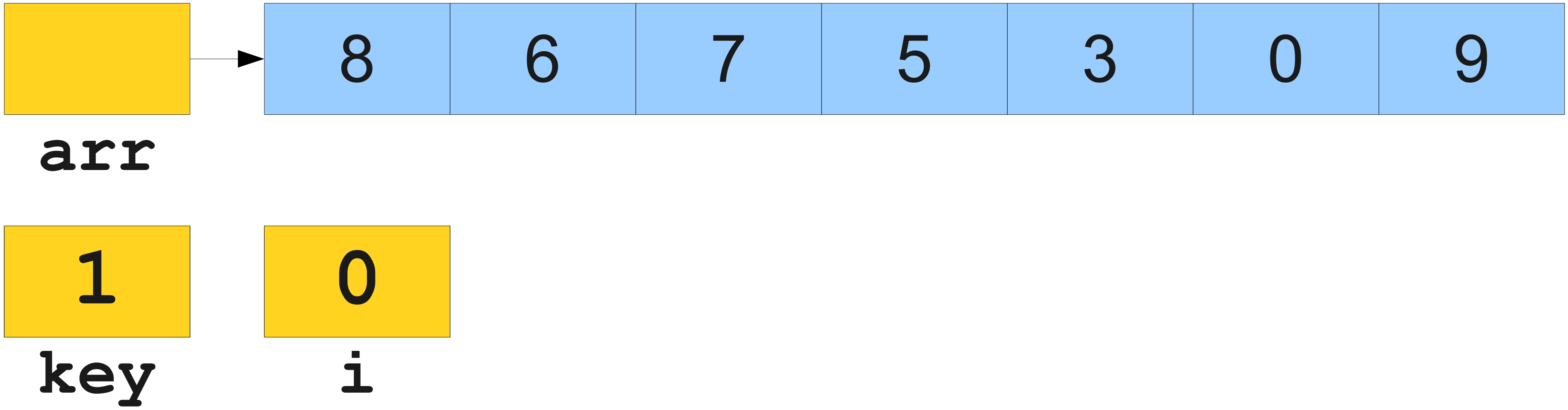
Linear Search

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

    return -1;
}
```



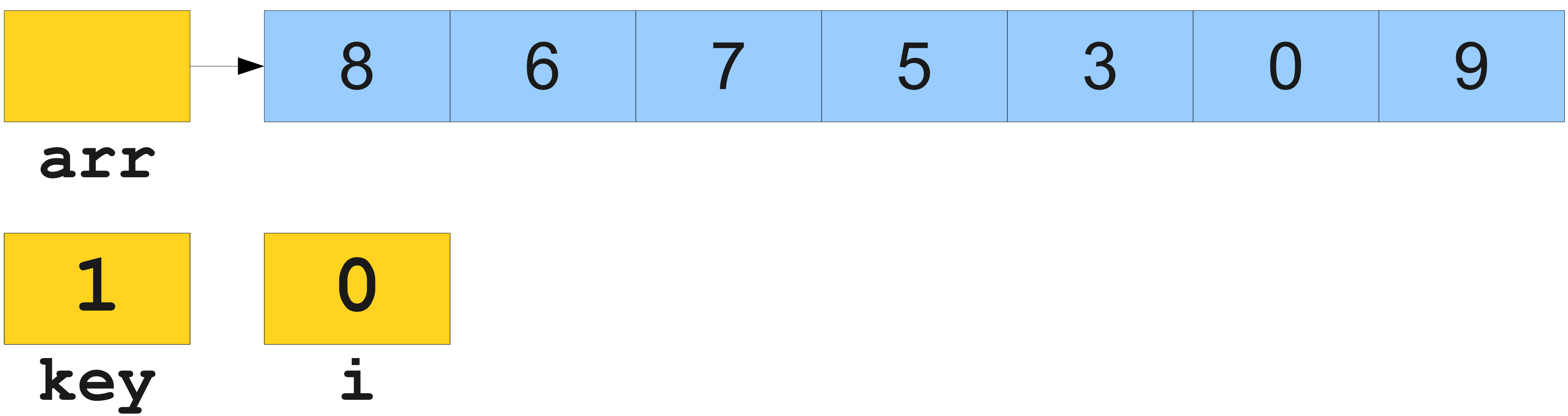
Linear Search

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

    return -1;
}
```



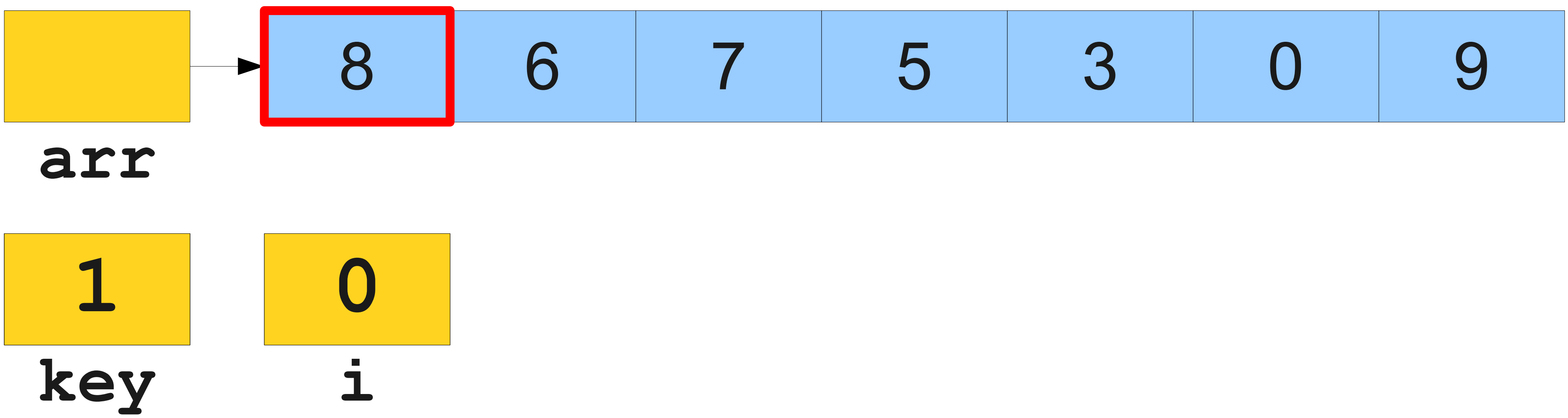
Linear Search

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

    return -1;
}
```



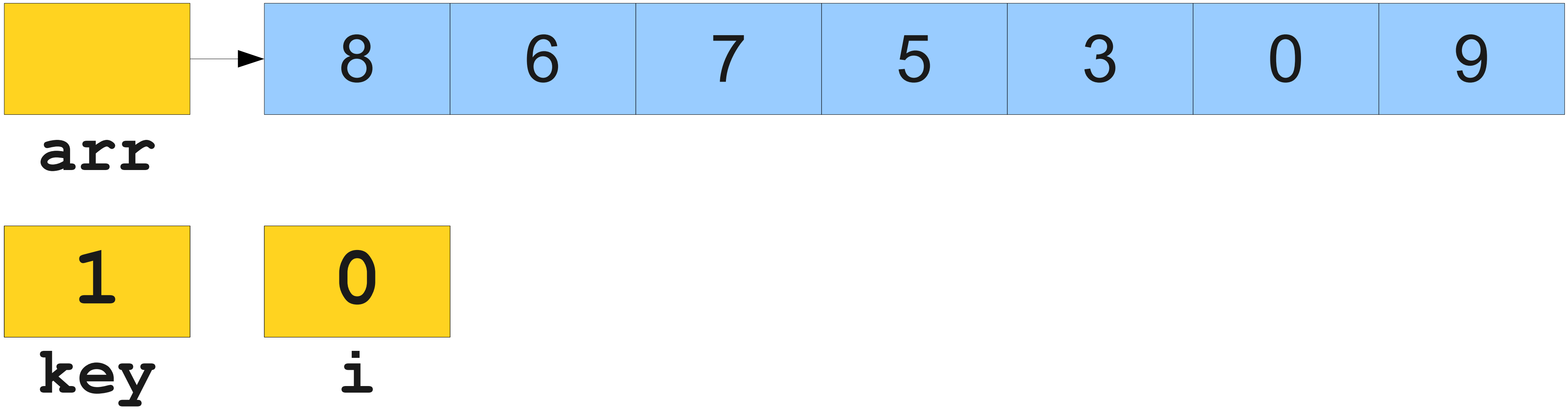
Linear Search

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

    return -1;
}
```



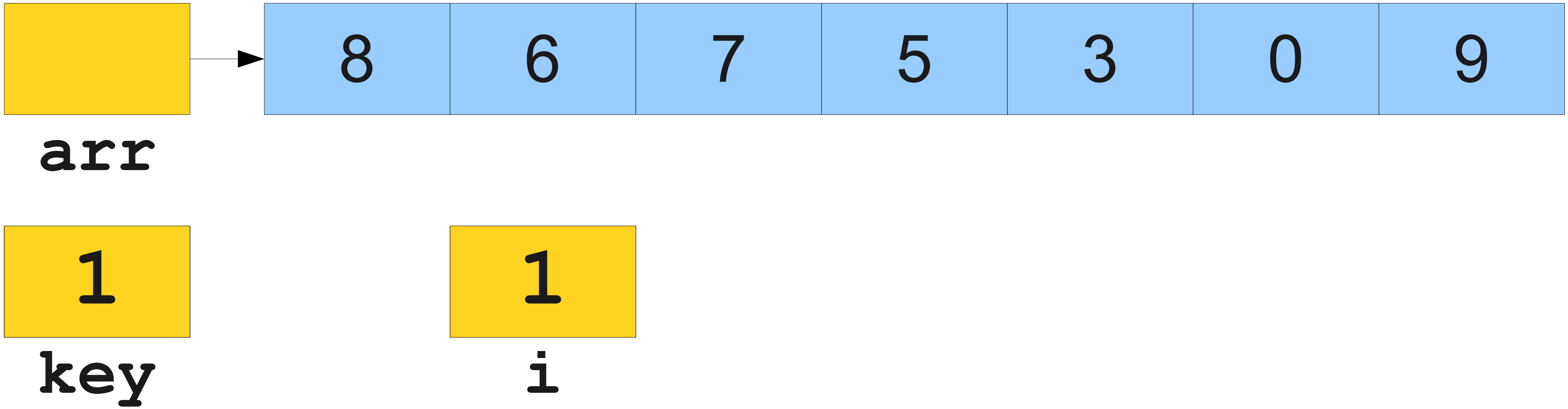
Linear Search

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

    return -1;
}
```



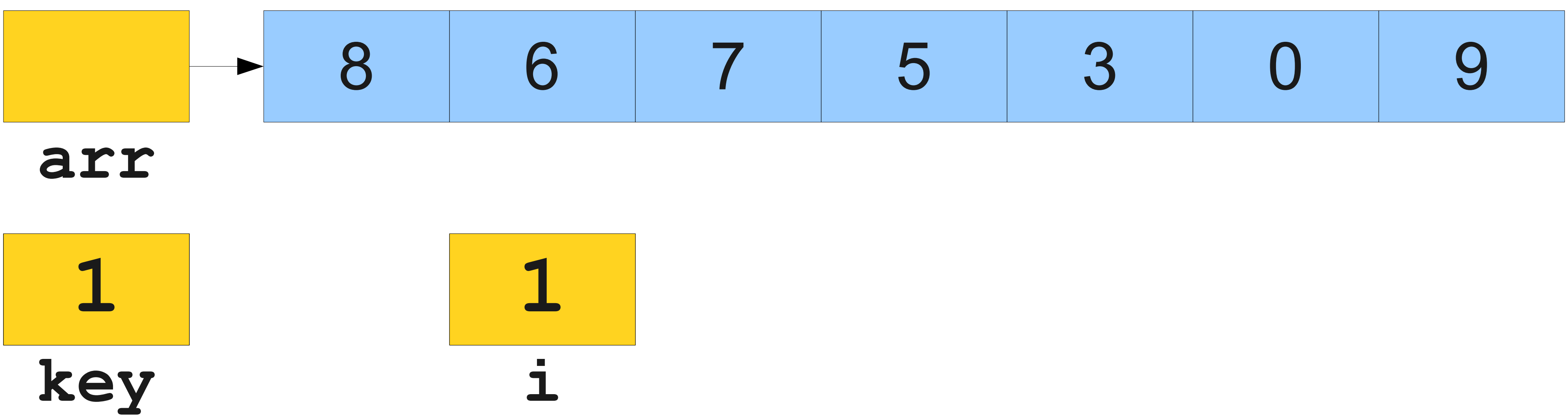
Linear Search

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

    return -1;
}
```



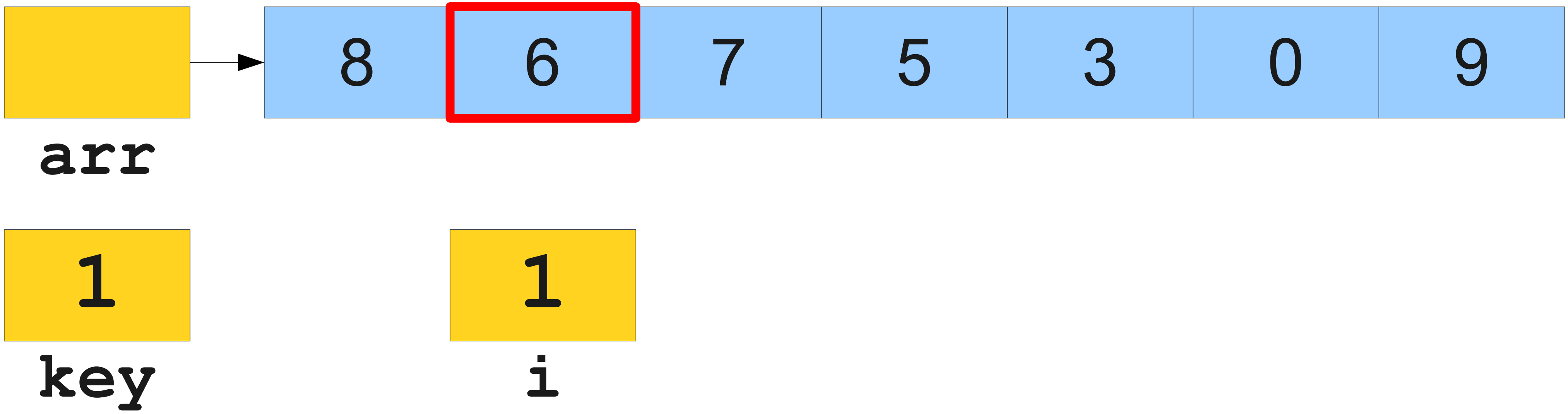
Linear Search

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

    return -1;
}
```



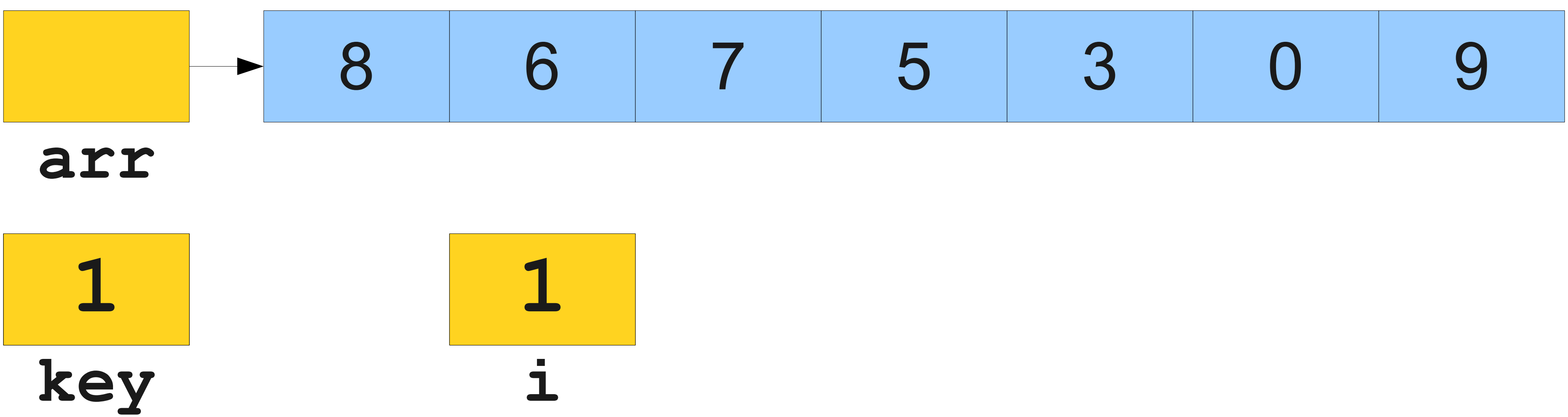
Linear Search

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

    return -1;
}
```



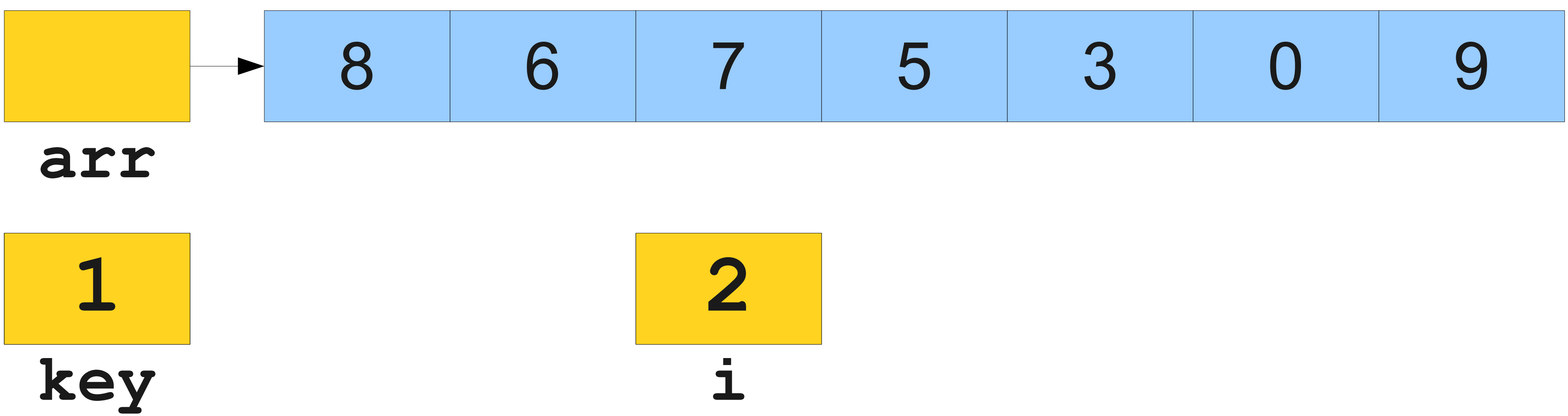
Linear Search

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

    return -1;
}
```



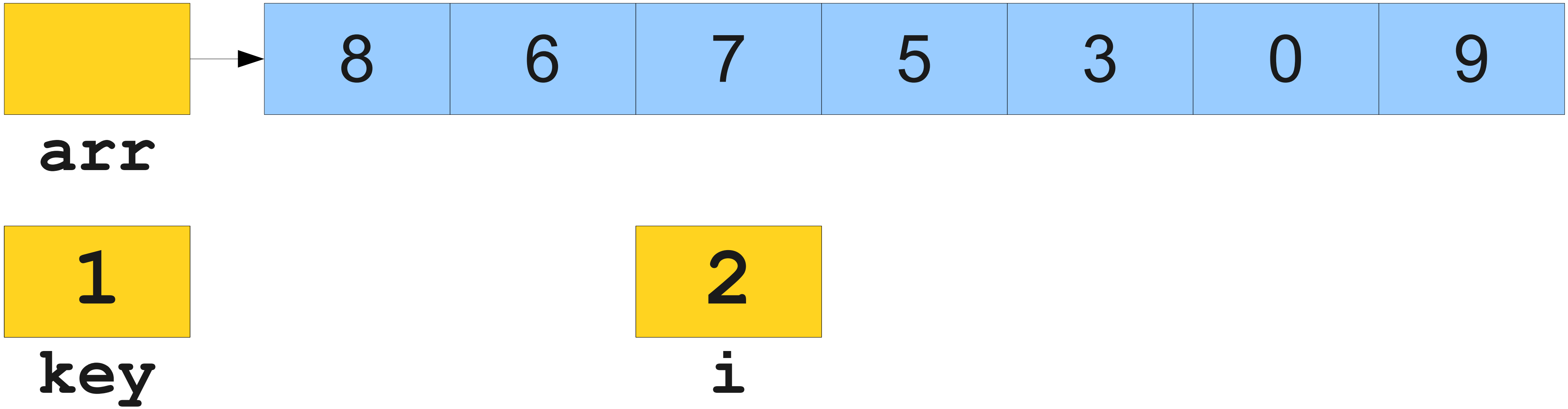
Linear Search

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

    return -1;
}
```



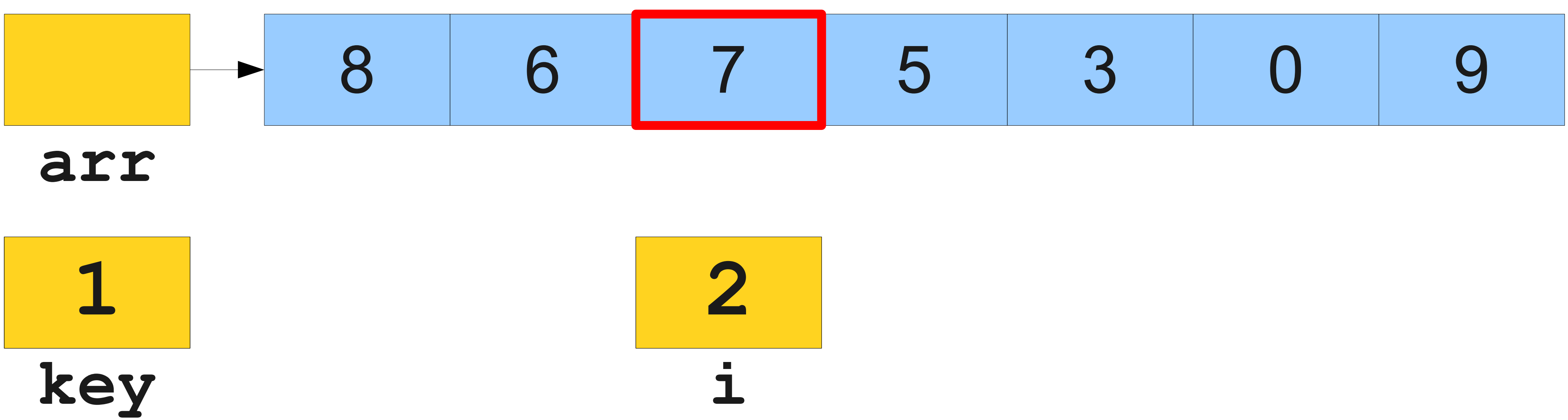
Linear Search

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

    return -1;
}
```



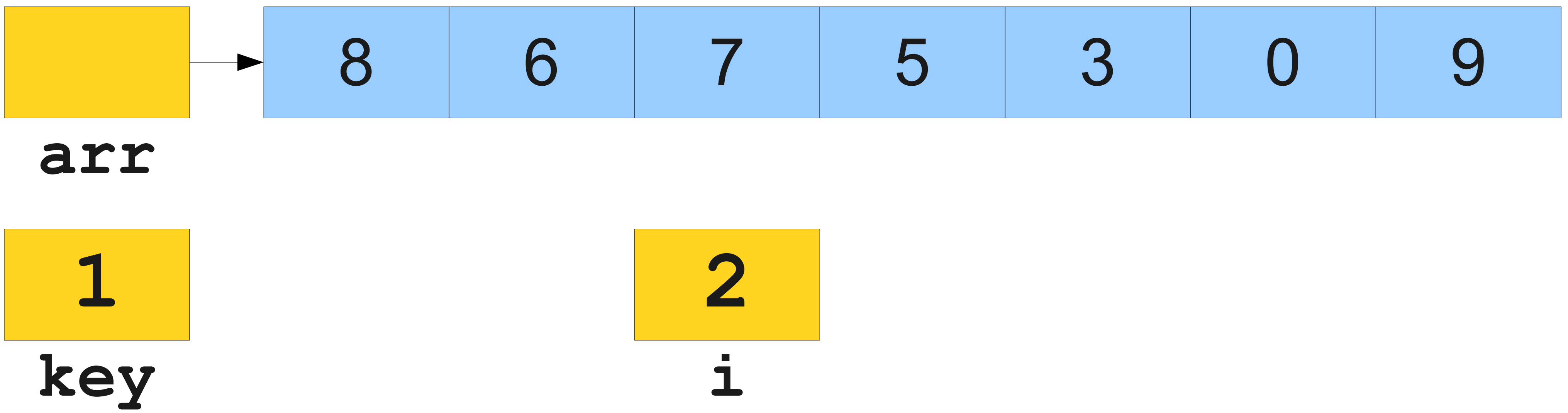
Linear Search

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

    return -1;
}
```



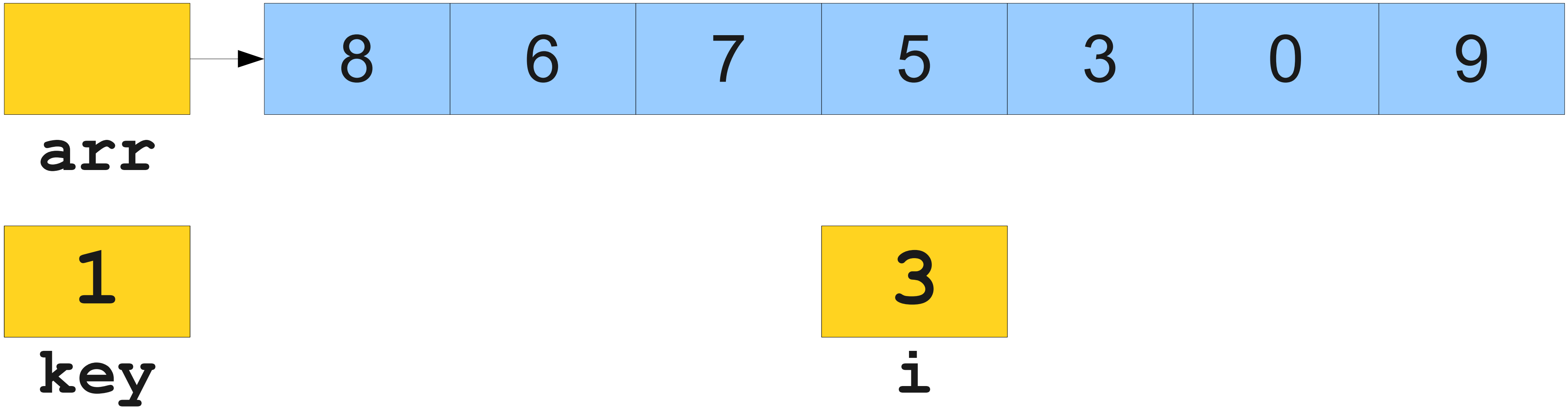
Linear Search

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

    return -1;
}
```



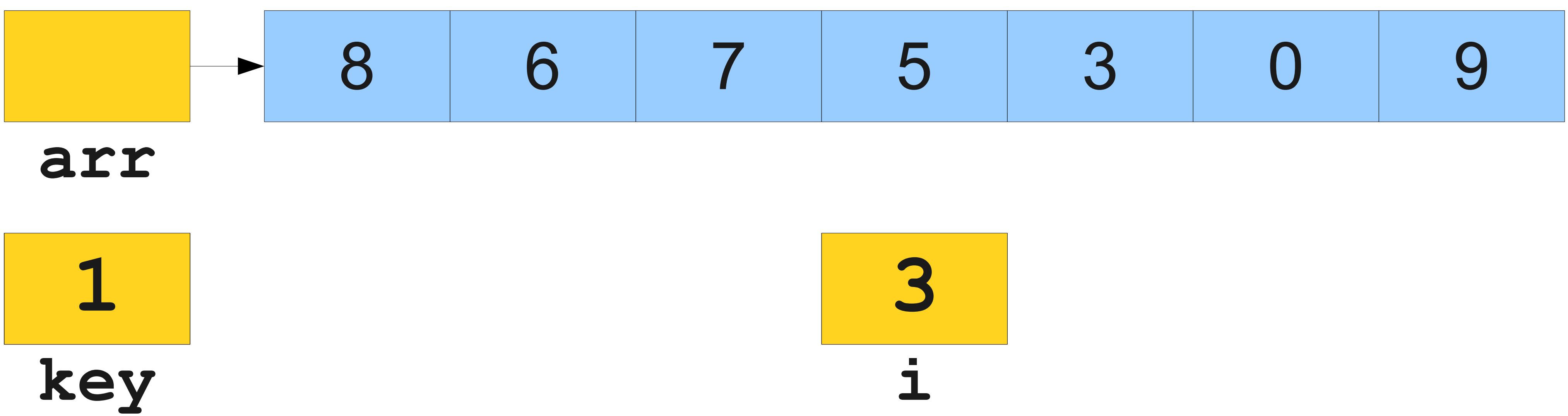
Linear Search

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

    return -1;
}
```



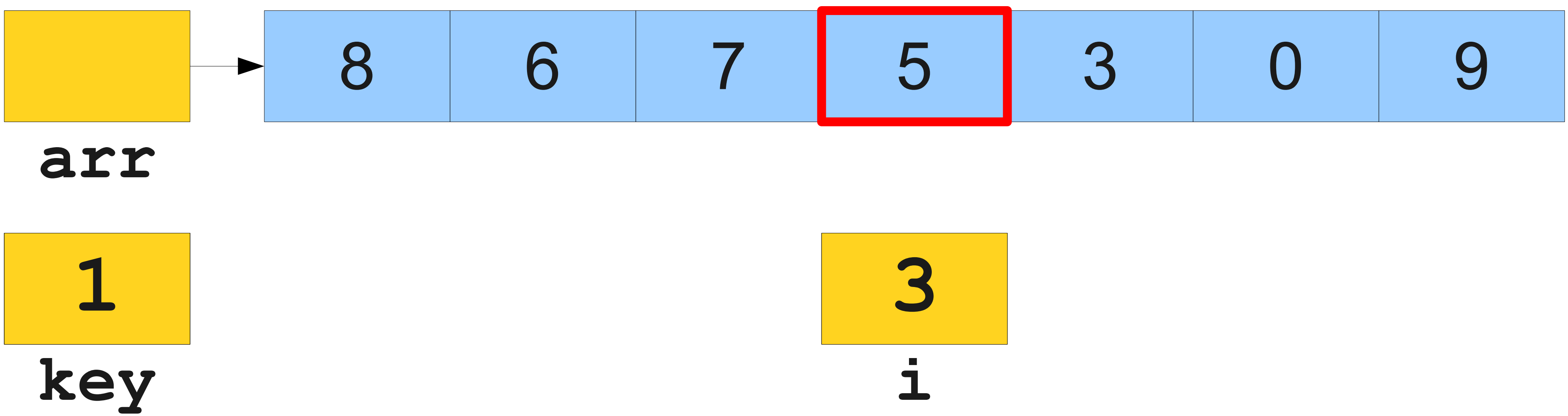
Linear Search

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

    return -1;
}
```



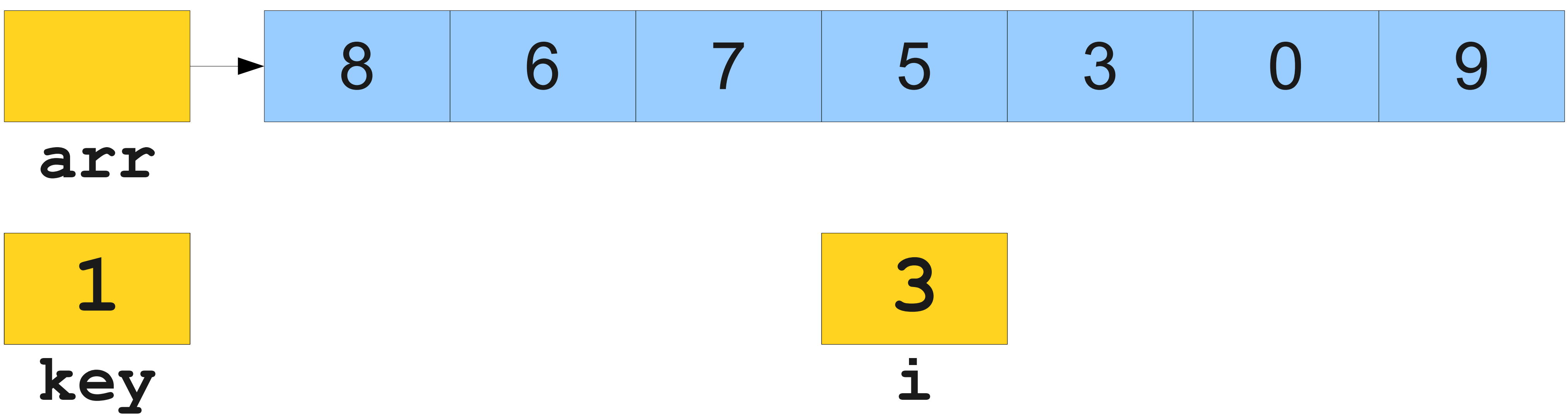
Linear Search

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

    return -1;
}
```



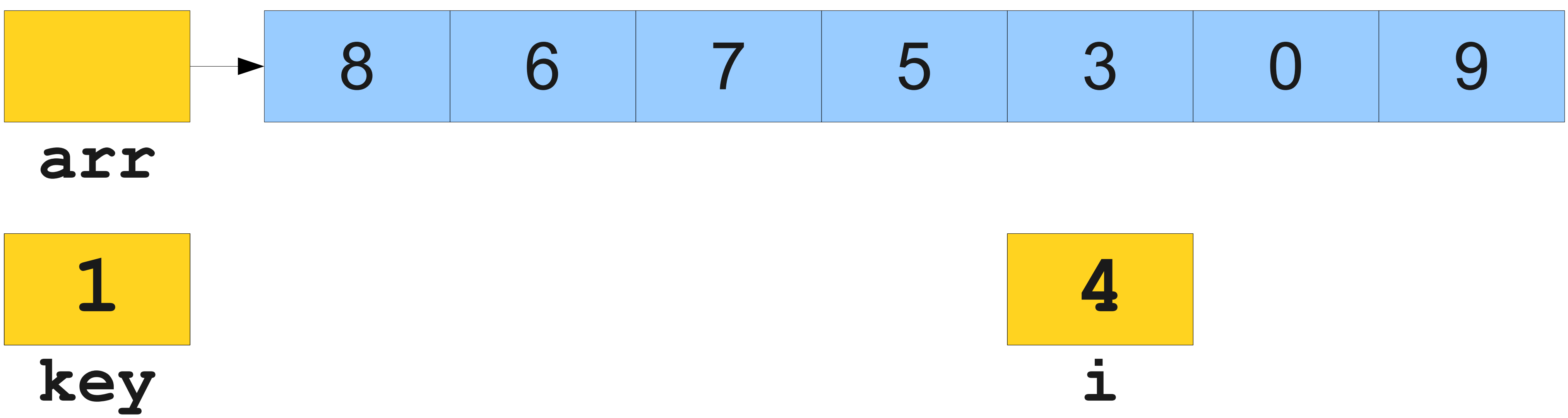
Linear Search

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

    return -1;
}
```



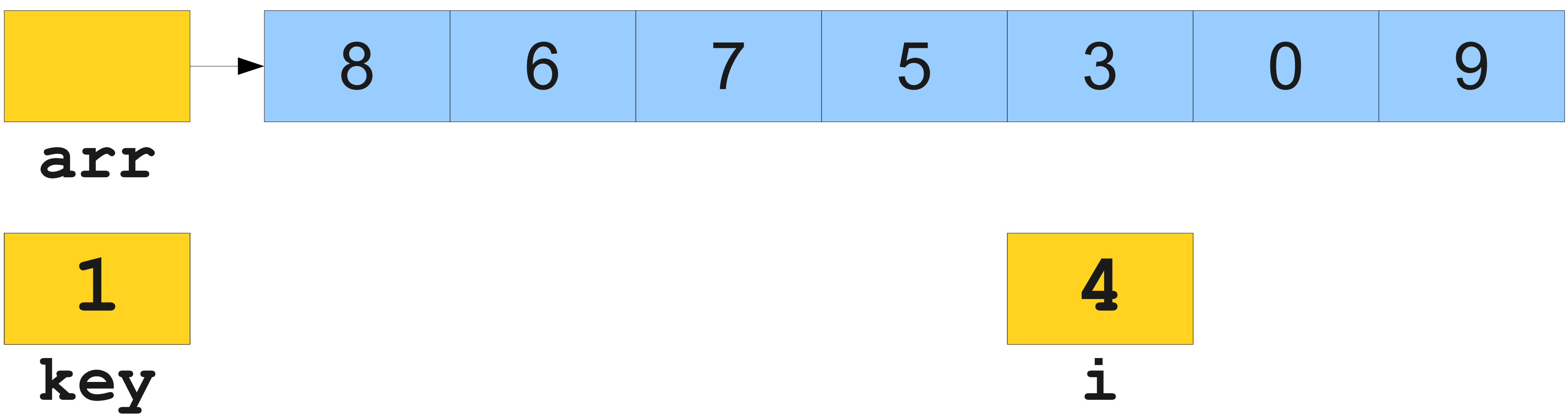
Linear Search

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

    return -1;
}
```



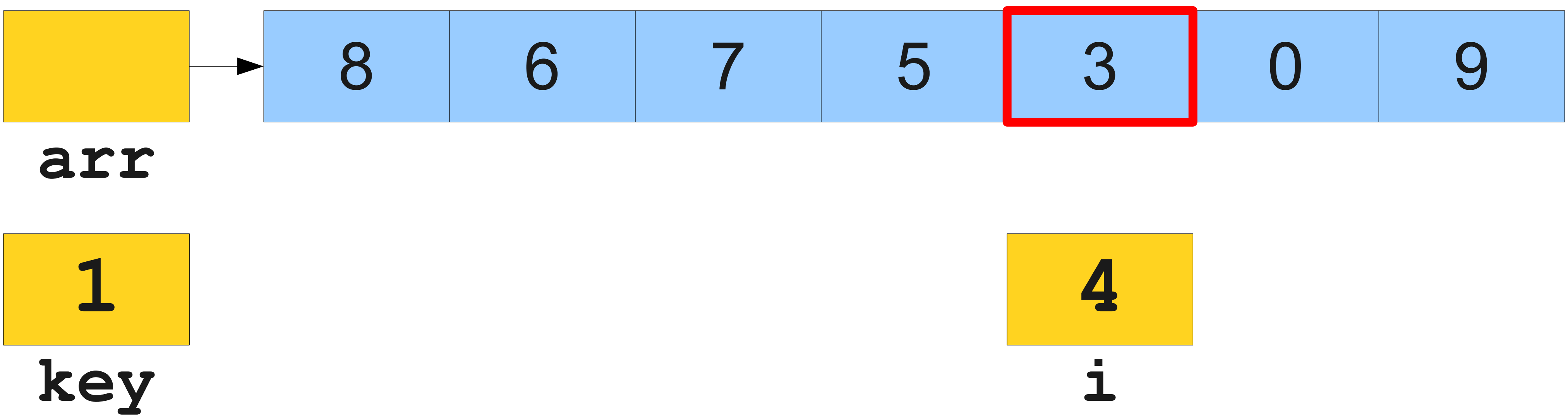
Linear Search

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

    return -1;
}
```



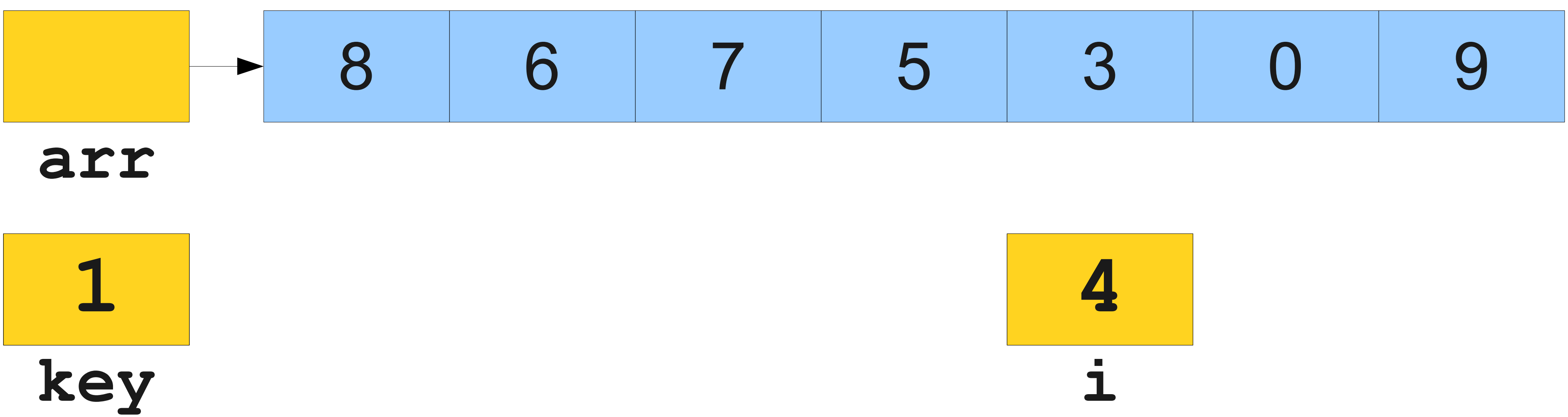
Linear Search

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

    return -1;
}
```



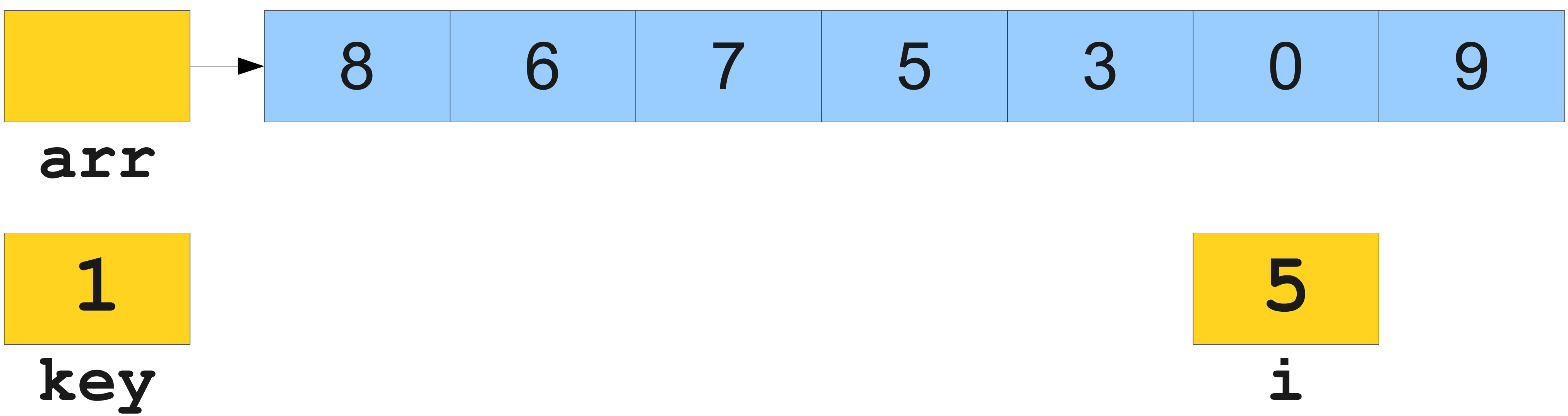
Linear Search

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

    return -1;
}
```



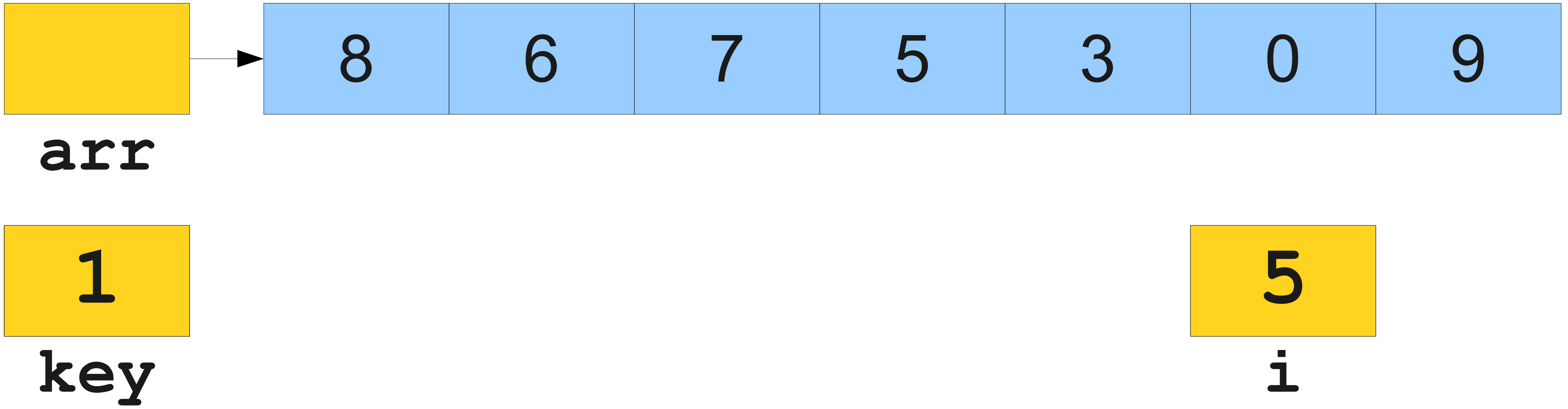
Linear Search

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

    return -1;
}
```



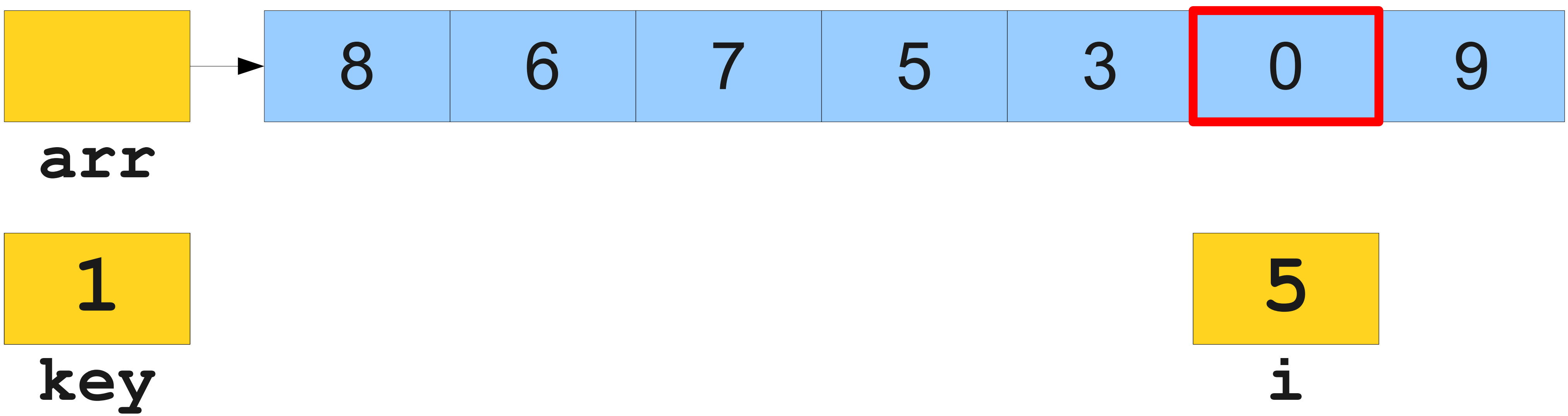
Linear Search

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

    return -1;
}
```



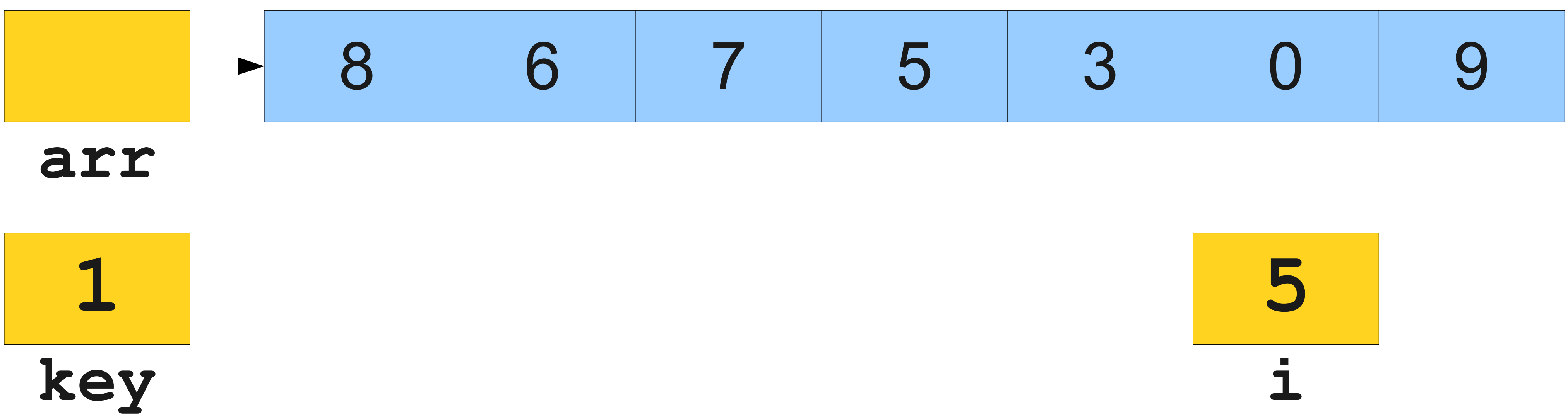
Linear Search

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

    return -1;
}
```



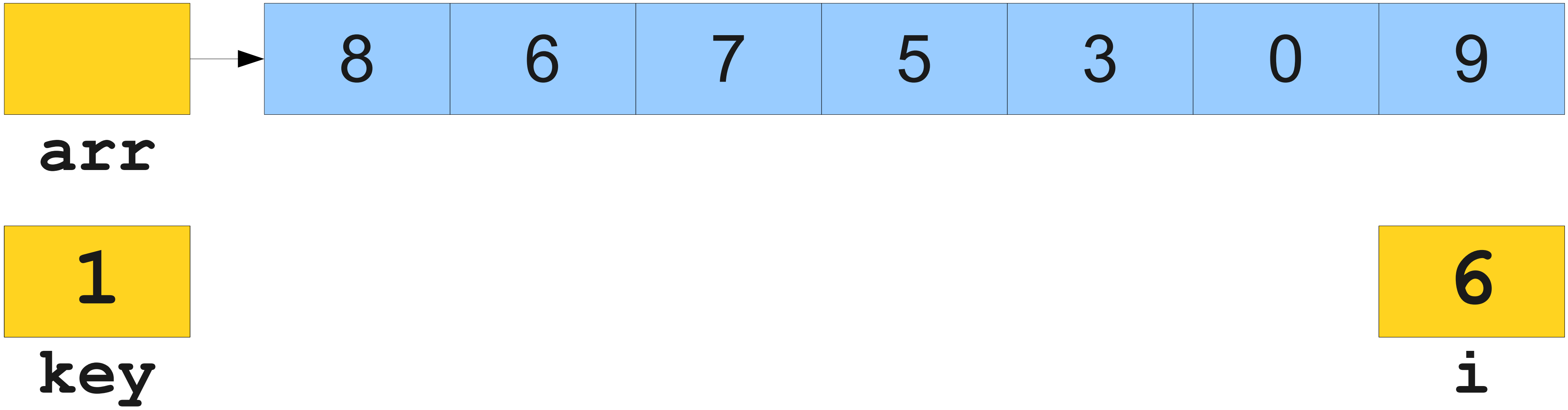
Linear Search

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

    return -1;
}
```



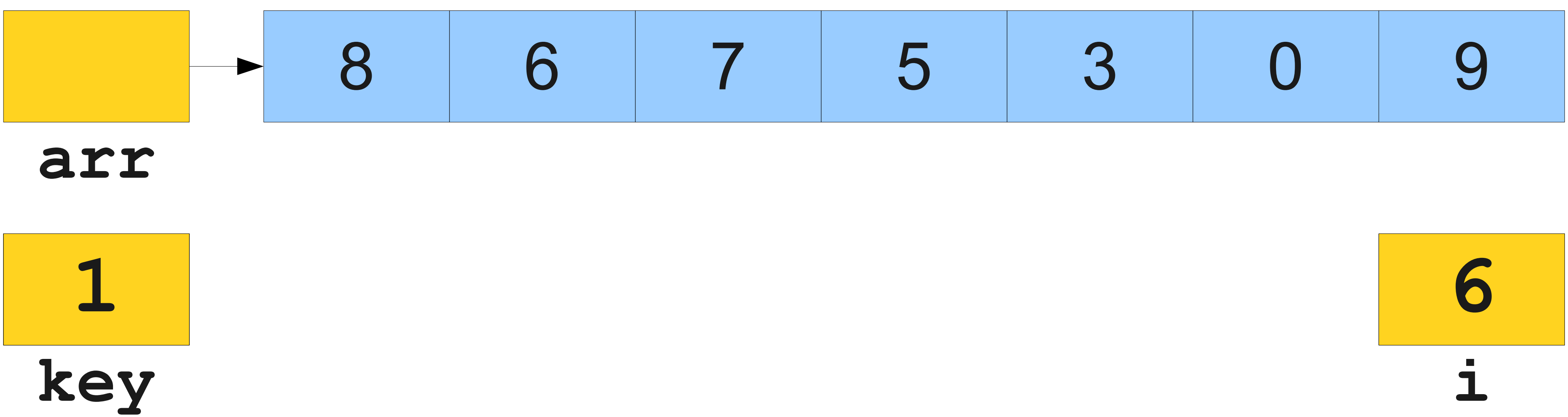
Linear Search

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

    return -1;
}
```



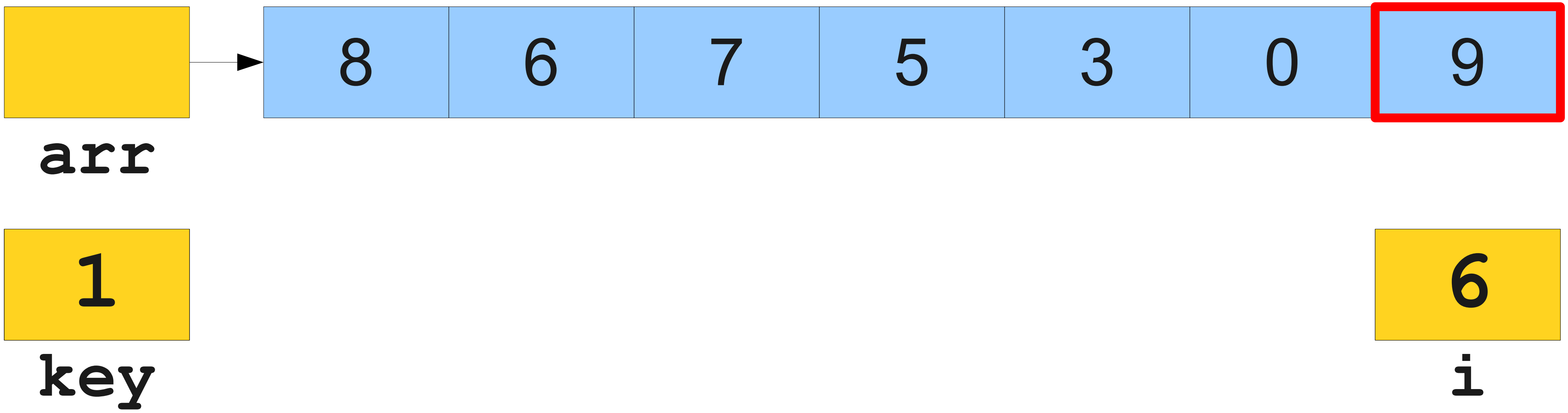
Linear Search

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

    return -1;
}
```



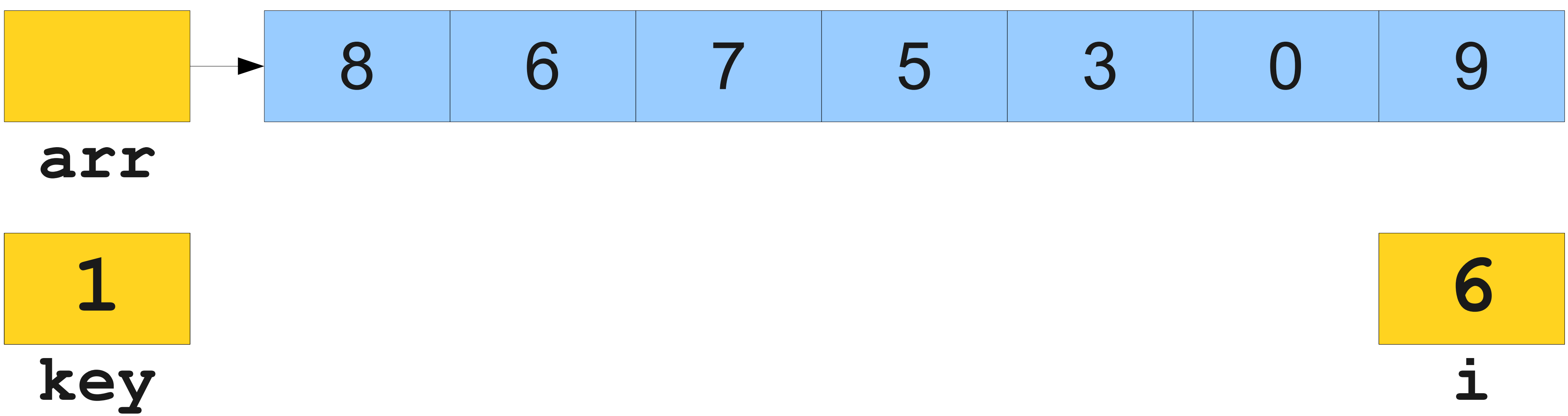
Linear Search

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

    return -1;
}
```



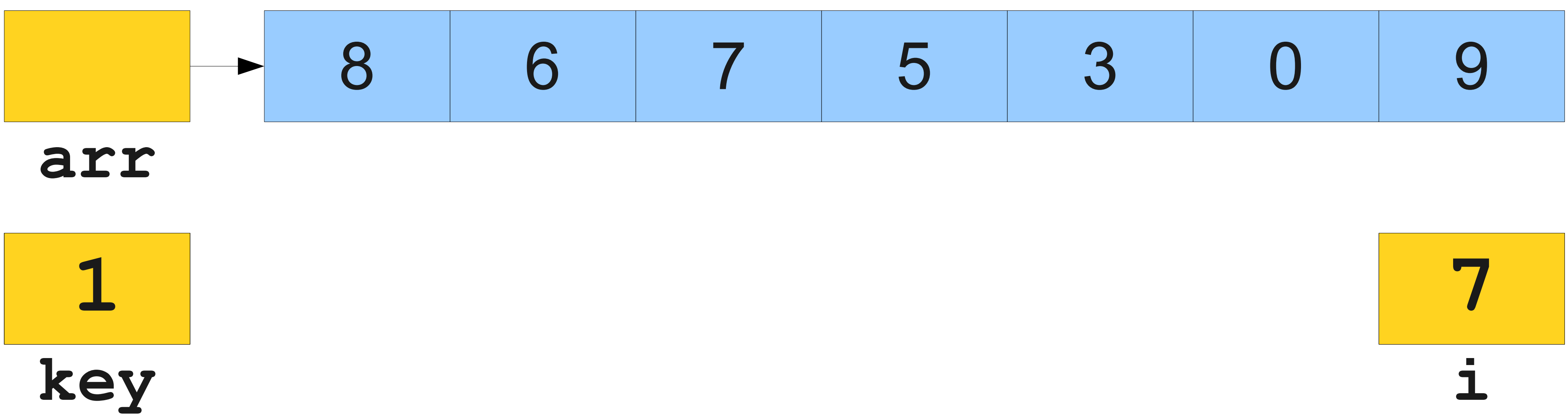
Linear Search

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

    return -1;
}
```



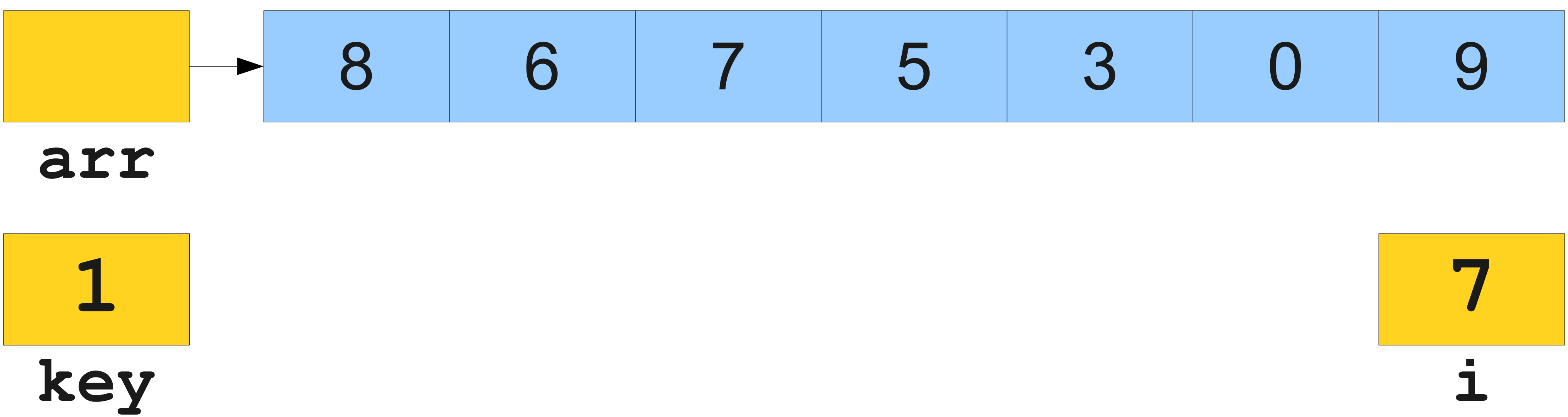
Linear Search

```
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;
}
```

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

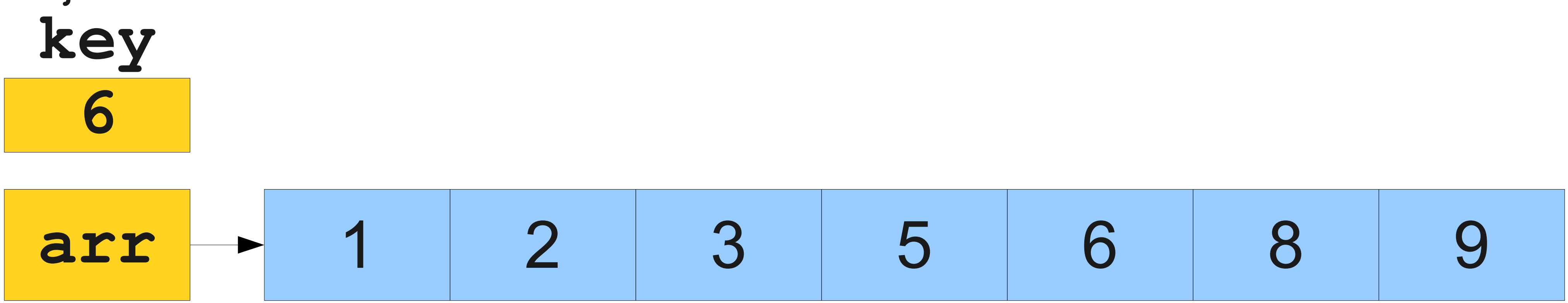
2	3	5	7	11	13	17	19	23	29	31	37	41
43	47	53	59	61	67	71	73	79	83	89	97	101
103	107	109	113	127	131	137	139	149	151	157	163	167

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;
}
```

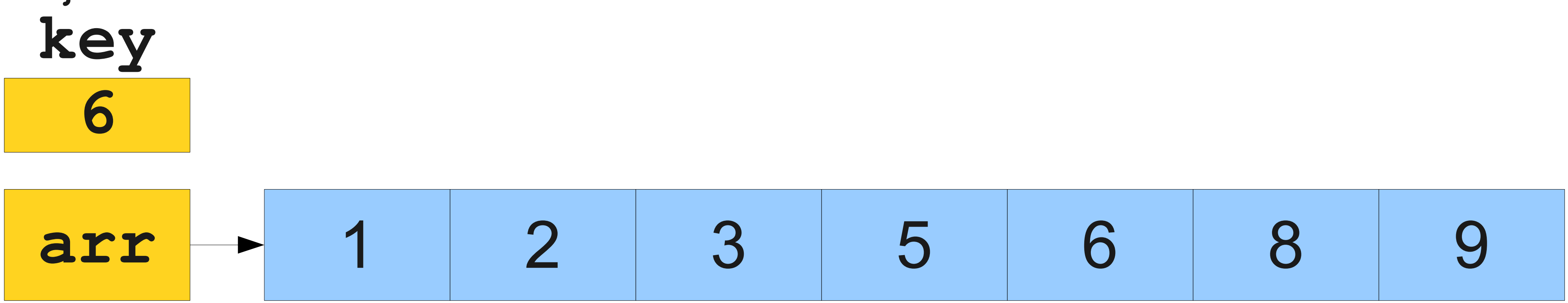
Binary Search

```
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;
}
```



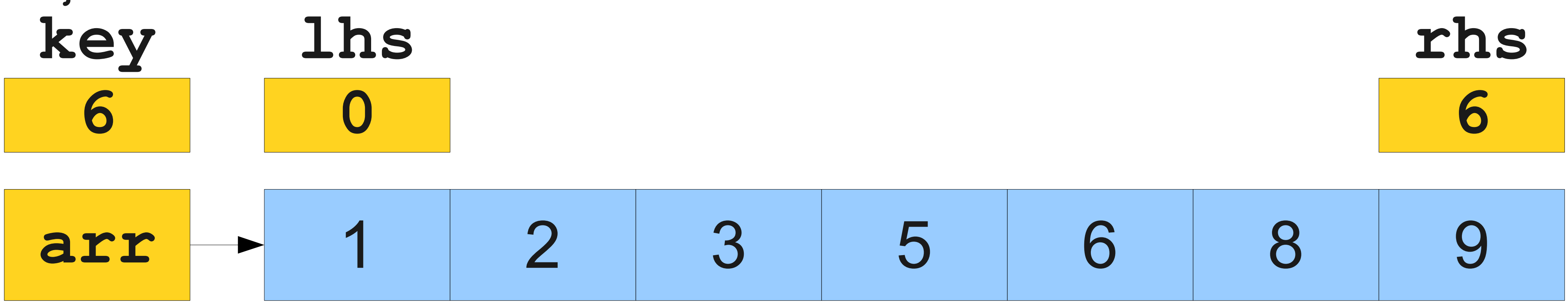
Binary Search

```java
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;
}
```



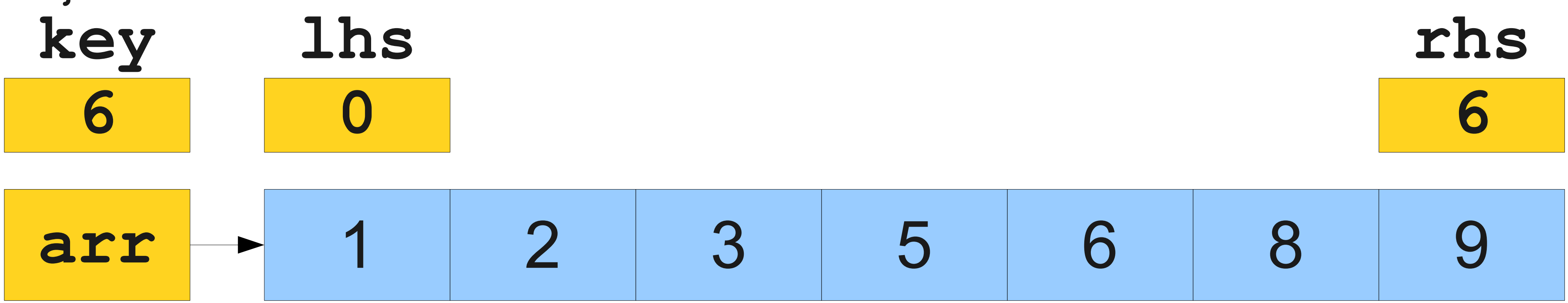
Binary Search

```
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;
}
```



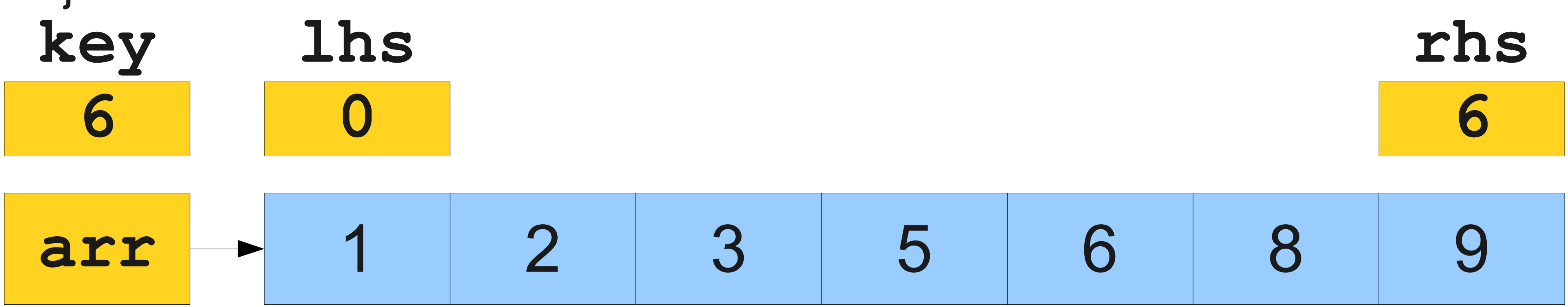
Binary Search

```
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;
}
```



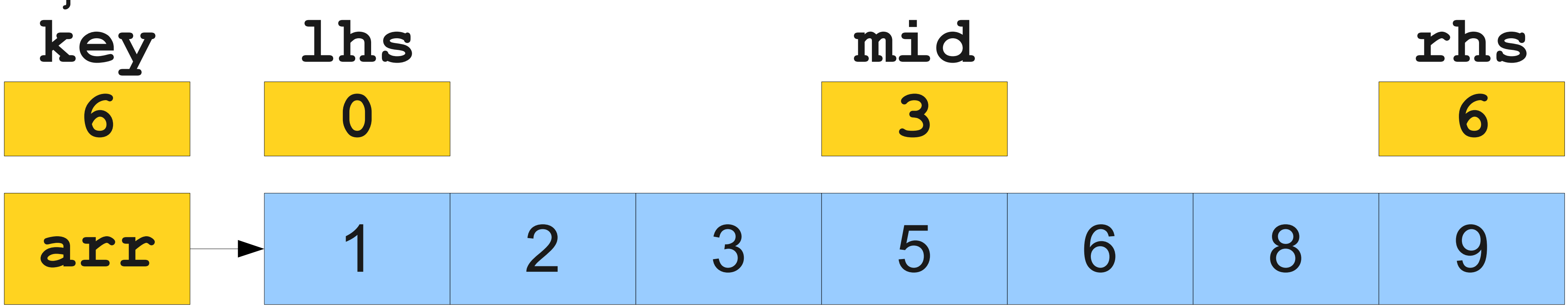
Binary Search

```
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;
}
```



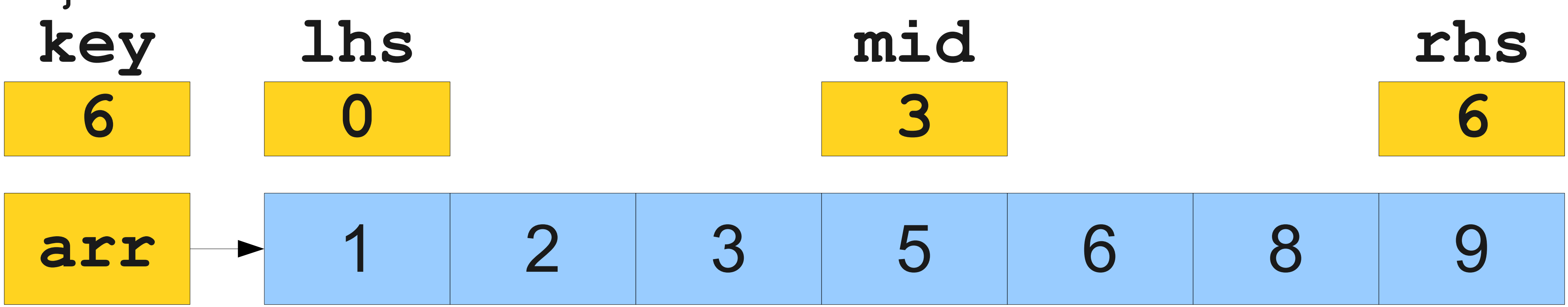
Binary Search

```
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;
}
```



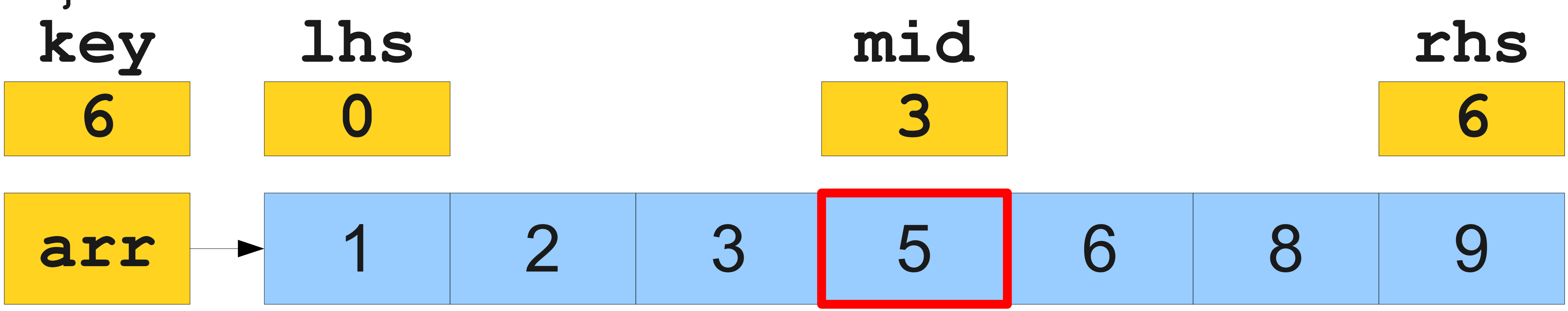
Binary Search

```
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;
}
```



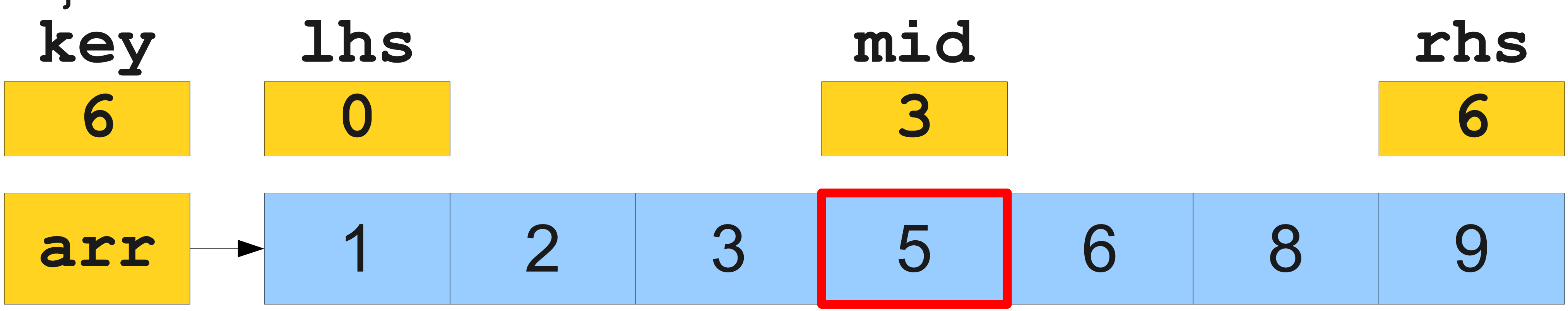
Binary Search

```
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;
}
```



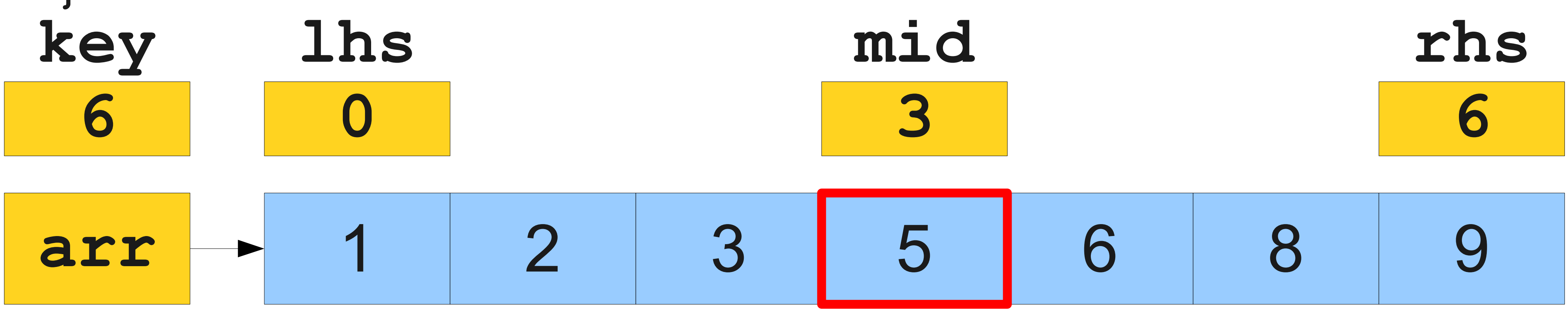
Binary Search

```
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;
}
```



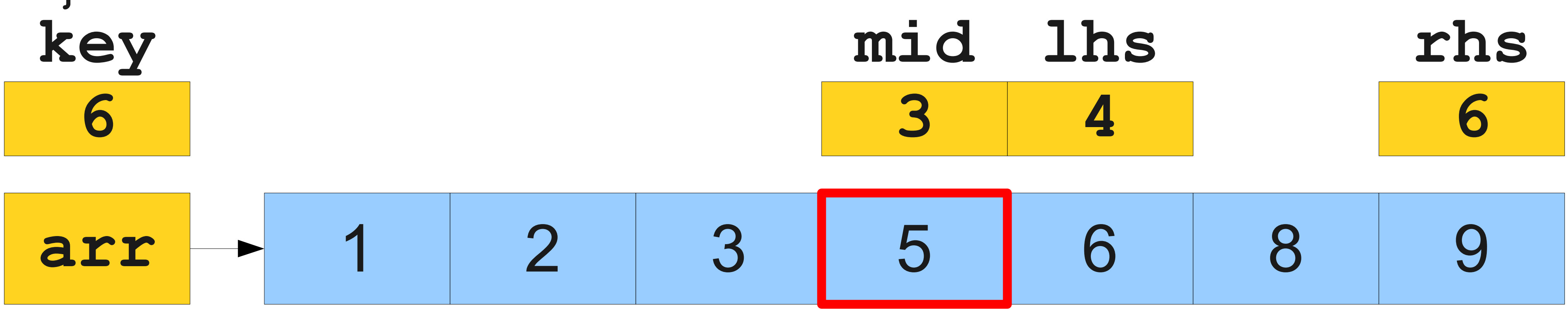
Binary Search

```
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;
}
```



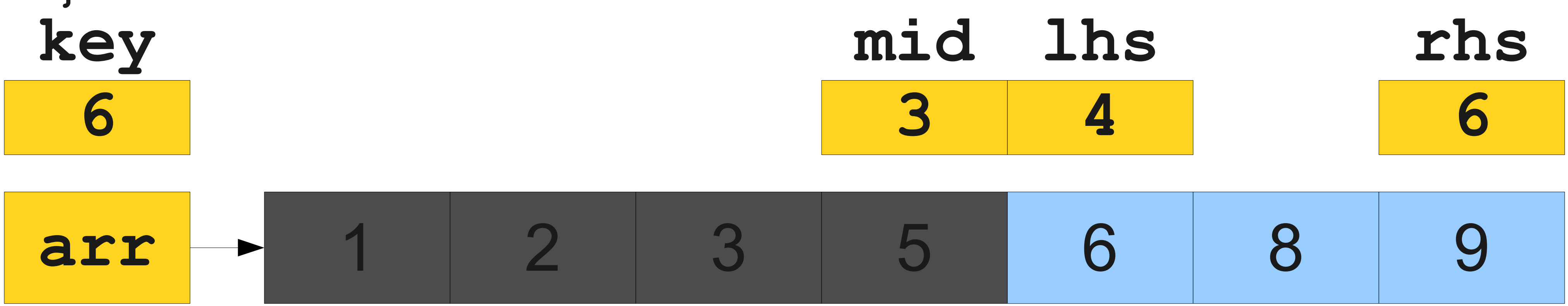
Binary Search

```
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;
}
```



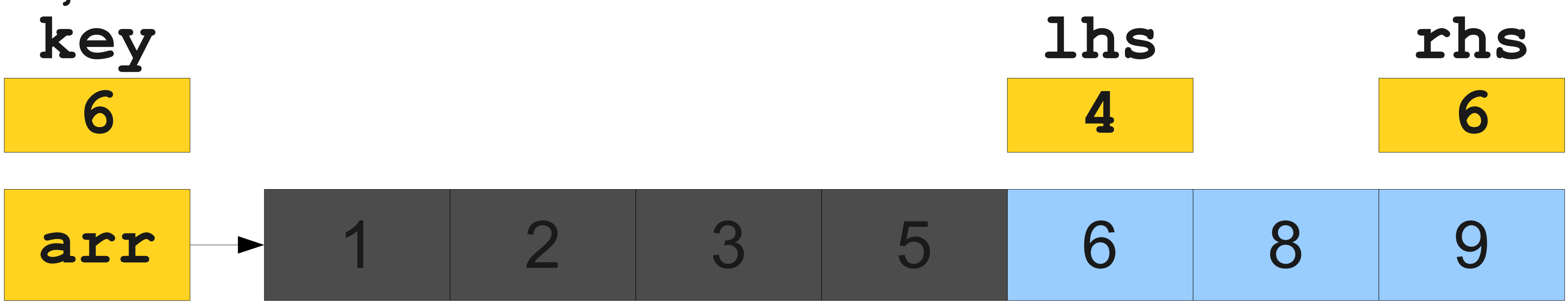
Binary Search

```
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;
}
```



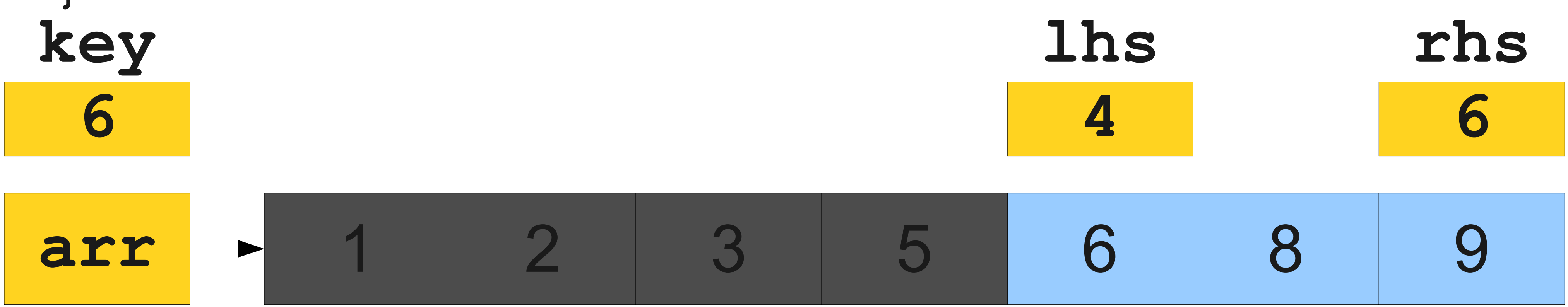
Binary Search

```
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;
}
```



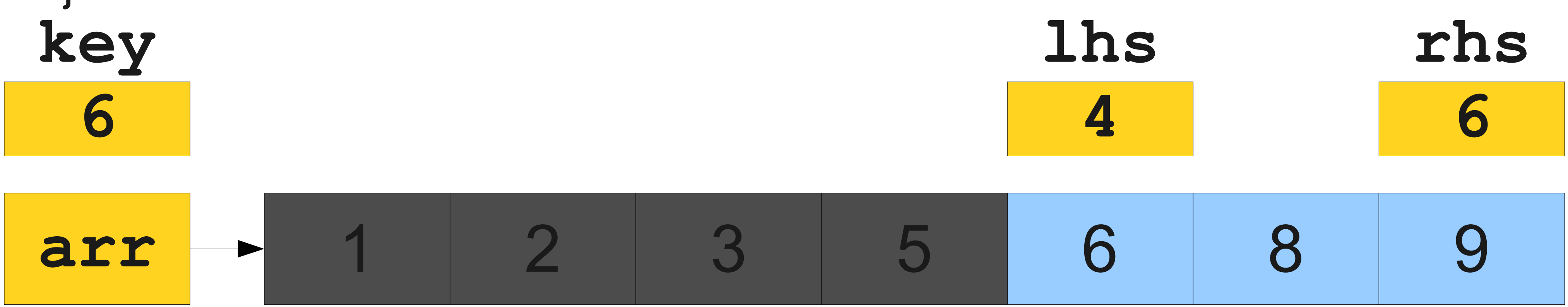
Binary Search

```
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;
}
```



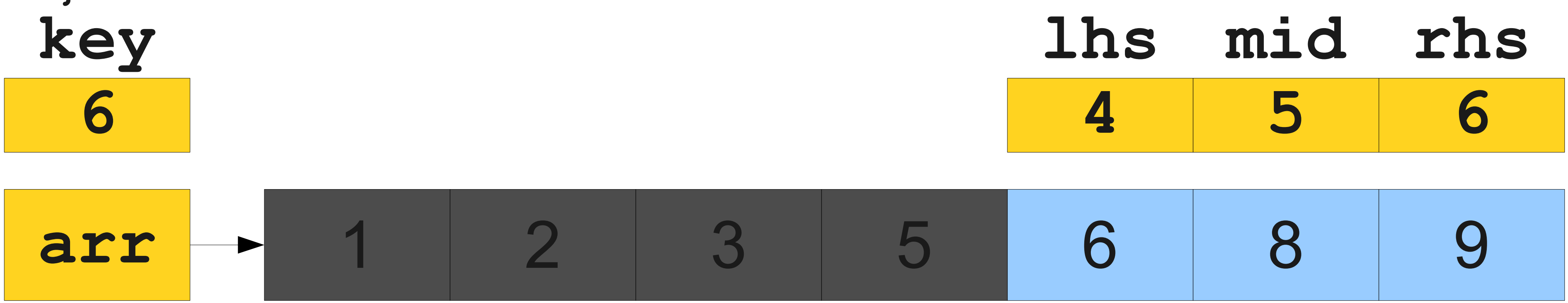
Binary Search

```
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;
}
```



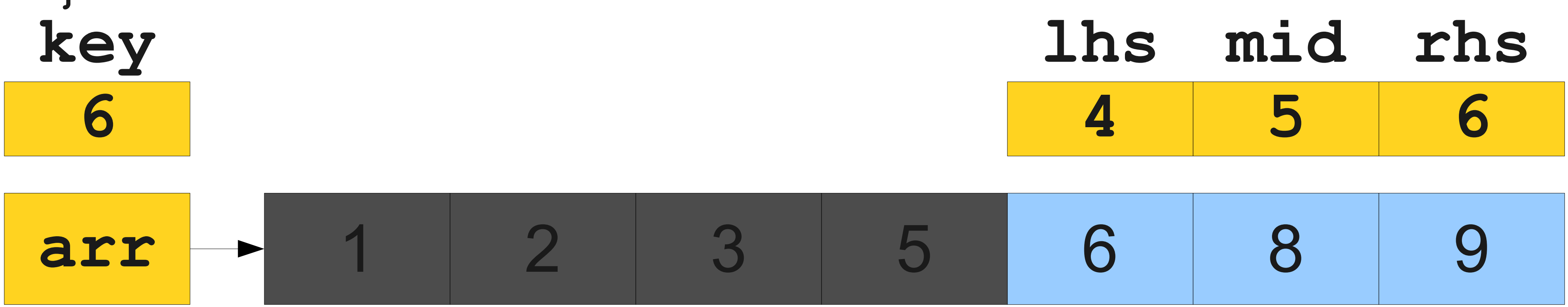
Binary Search

```
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;
}
```



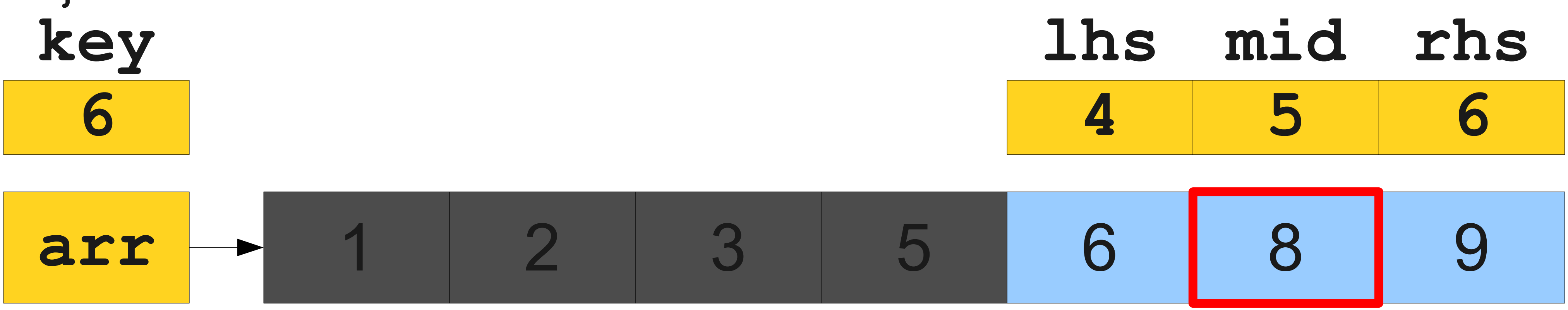
Binary Search

```
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;
}
```



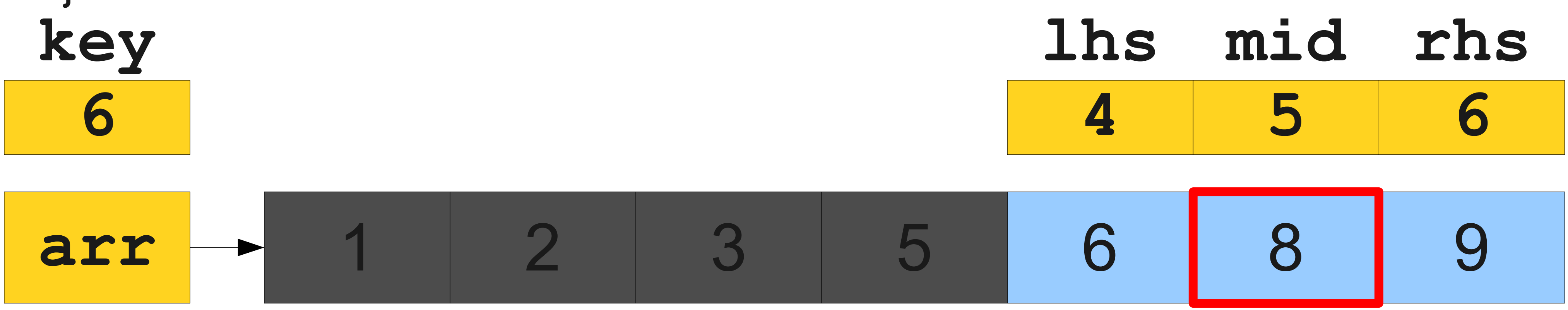
Binary Search

```java
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;
}
```



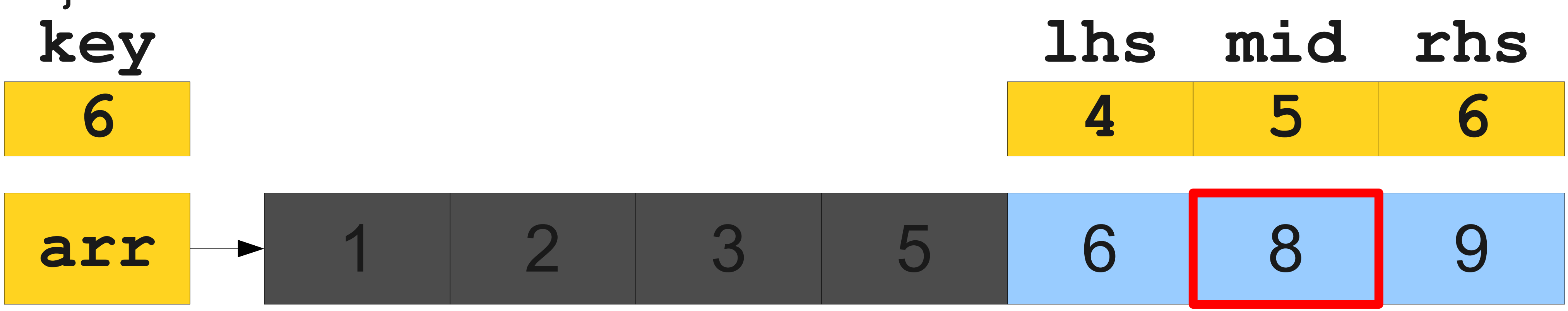
Binary Search

```
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;
}
```



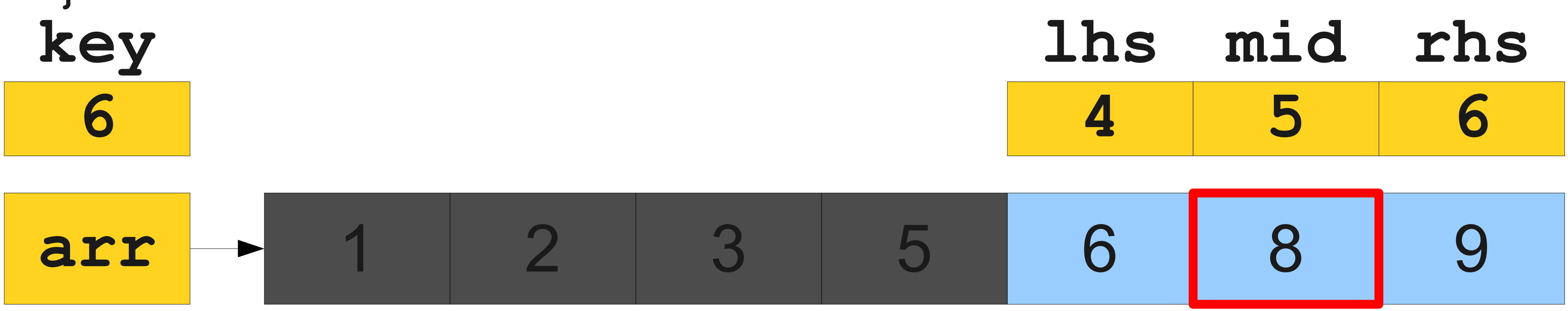
Binary Search

```
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;
}
```



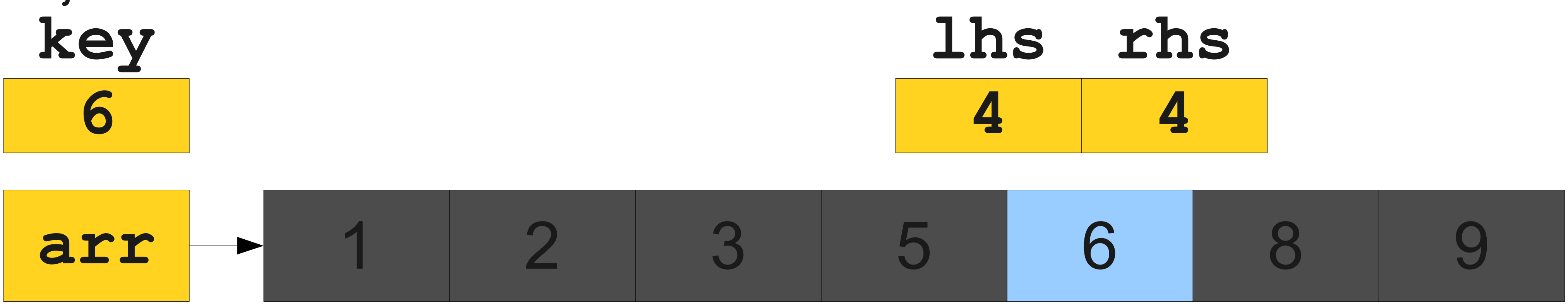
Binary Search

```
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;
}
```



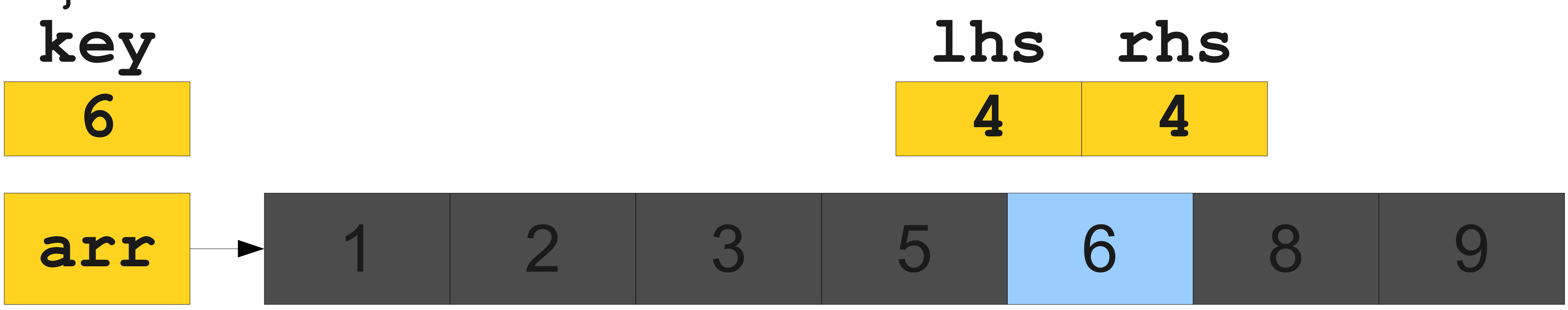
Binary Search

```
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;
}
```



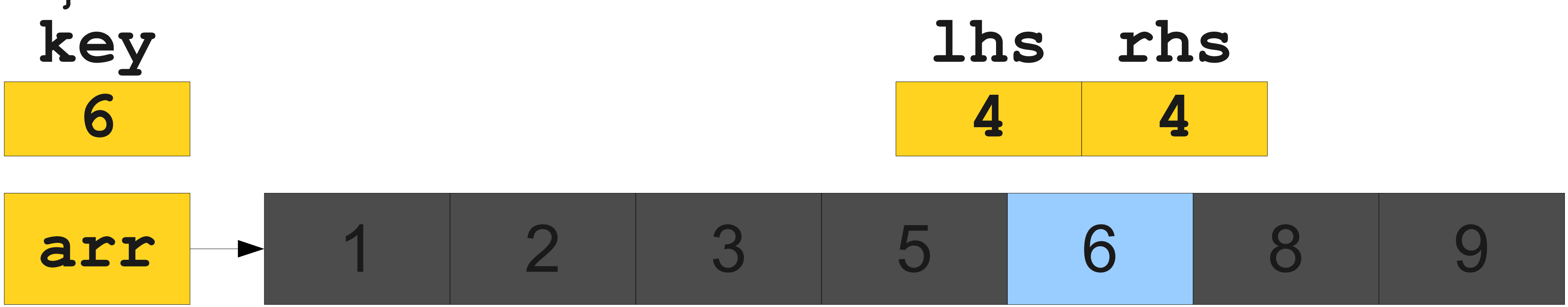
Binary Search

```
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;
}
```



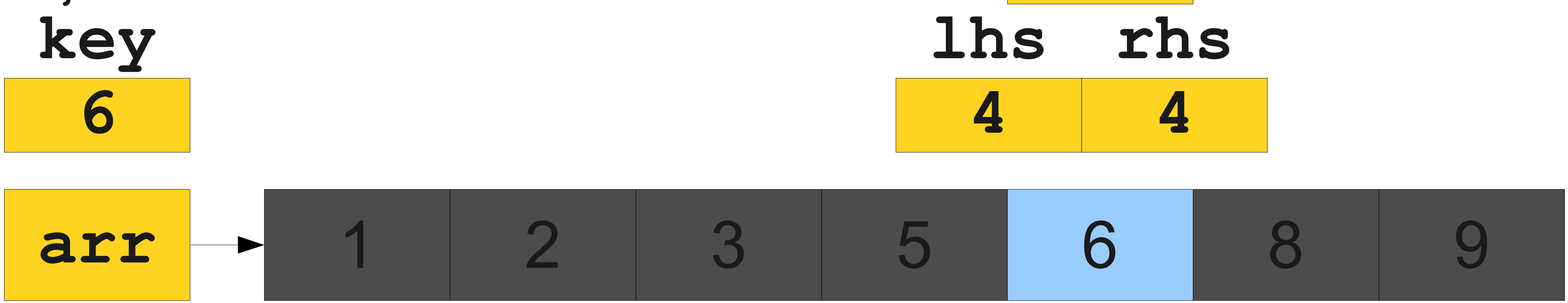
Binary Search

```
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;
}
```



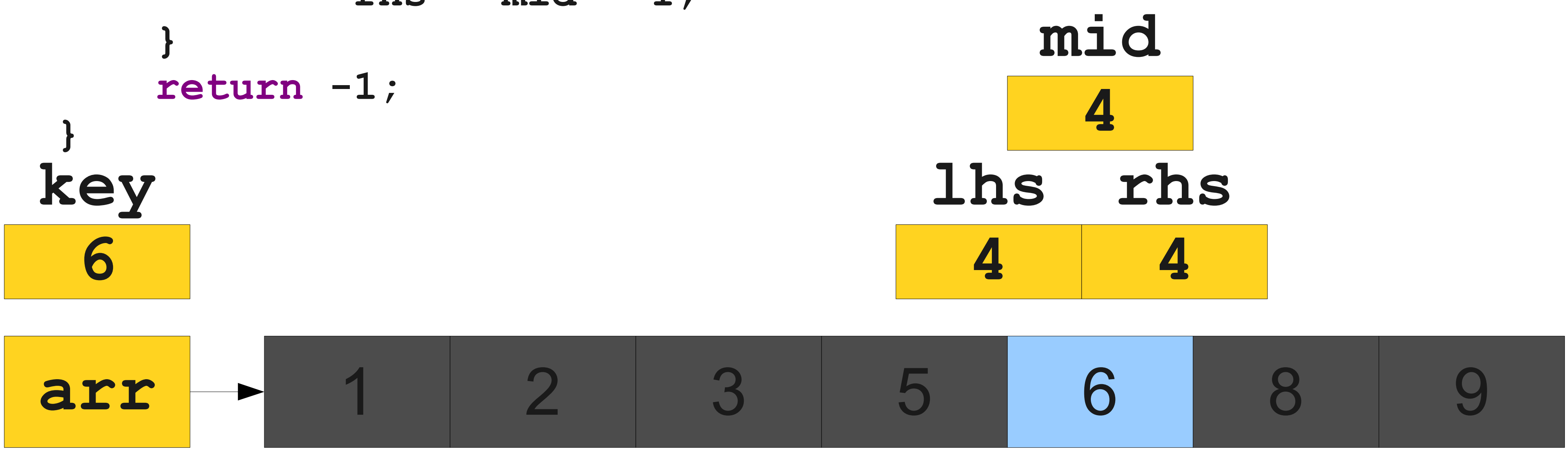
Binary Search

```
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;
}
```



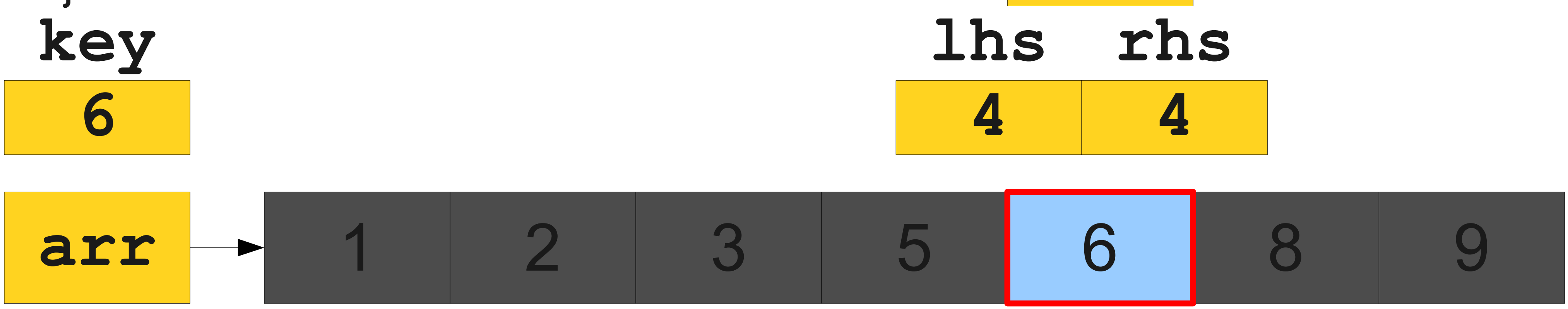
Binary Search

```
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;
}
```



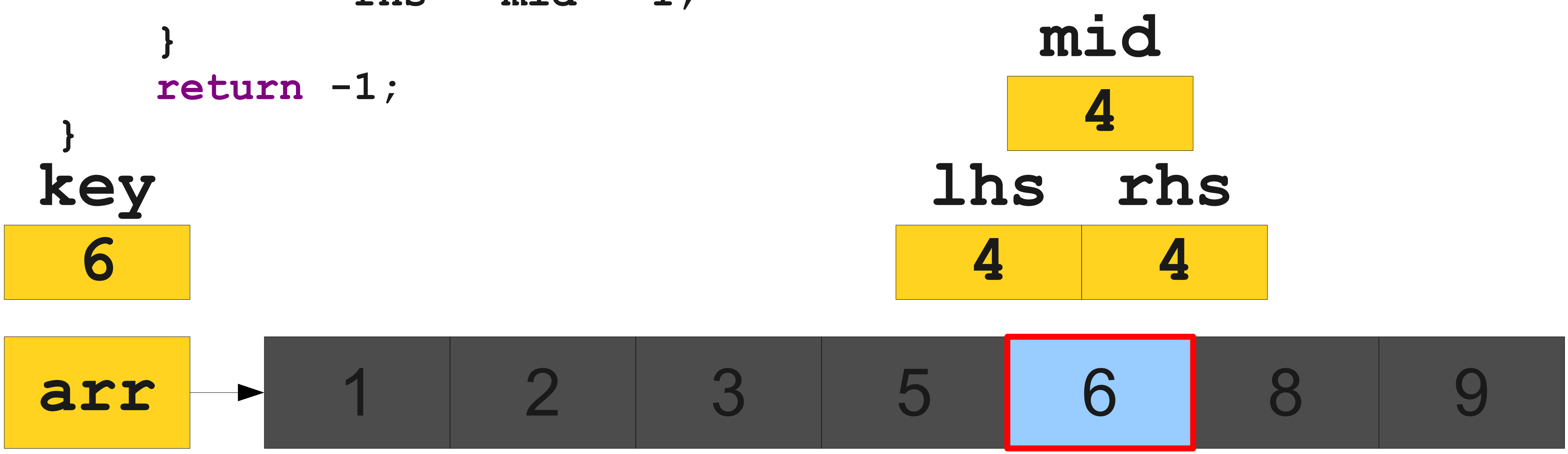
Binary Search

```java
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;
}
```



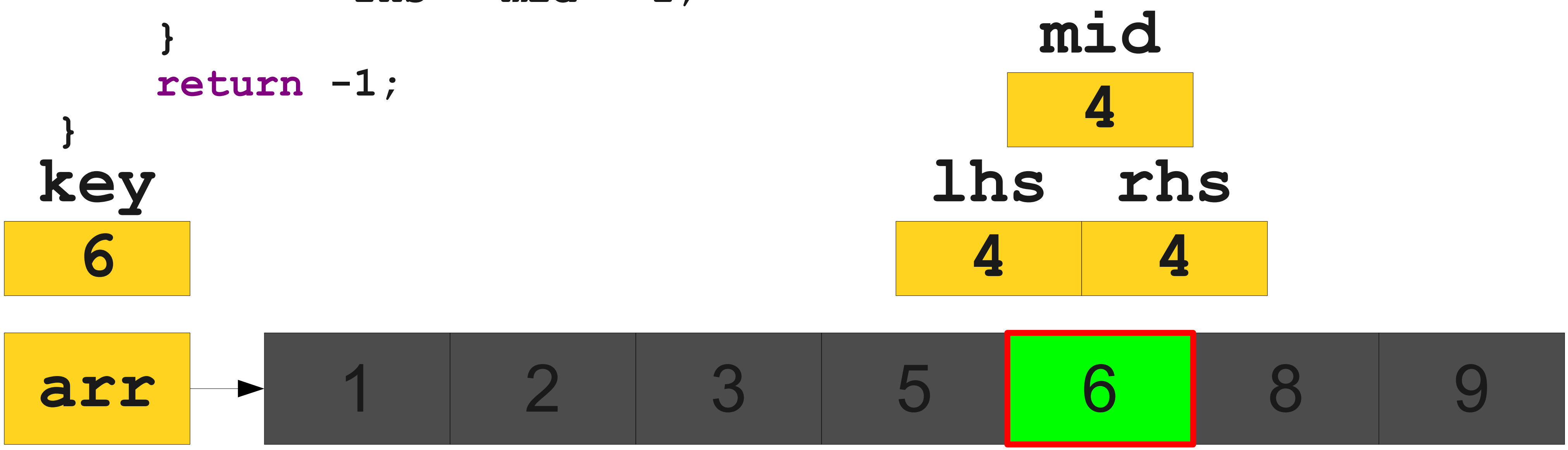
Binary Search

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

mid: 4

key: 6

lhs: 4 rhs: 4

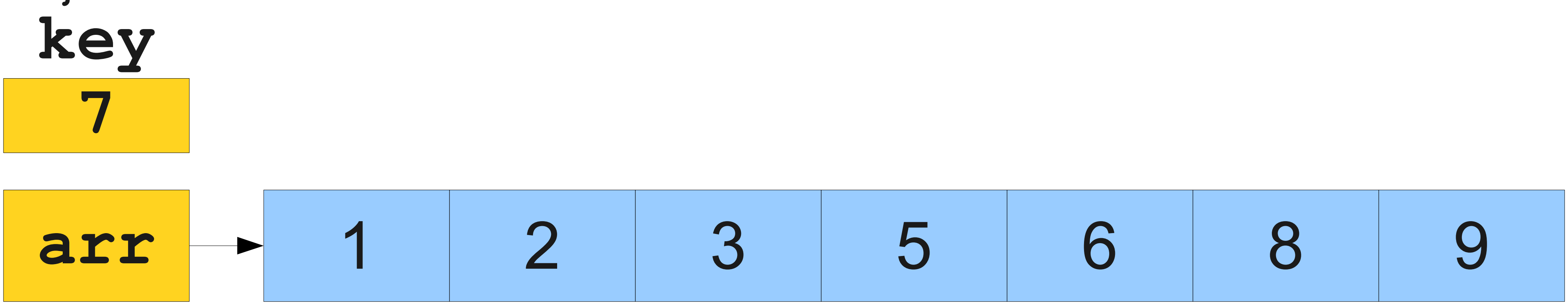
arr → | 1 | 2 | 3 | 5 | 6 | 8 | 9 |

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;
}
```

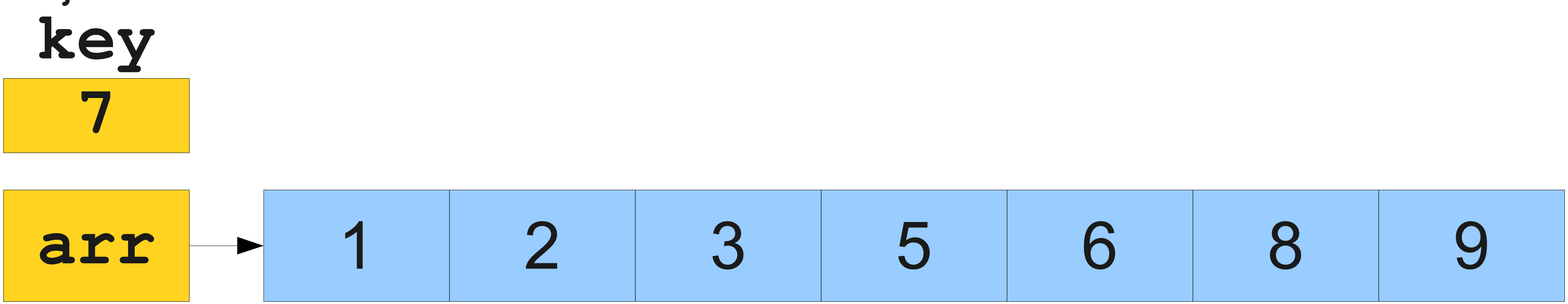
Binary Search

```java
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;
}
```



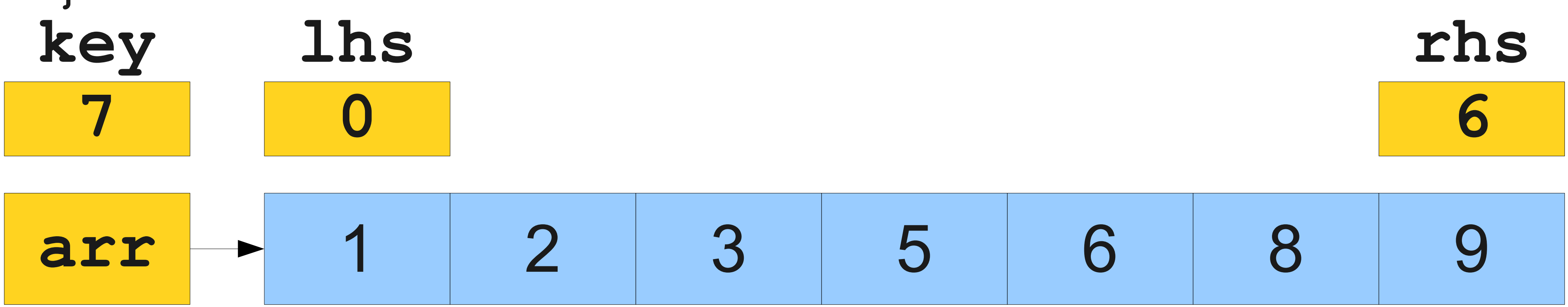
Binary Search

```
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;
}
```



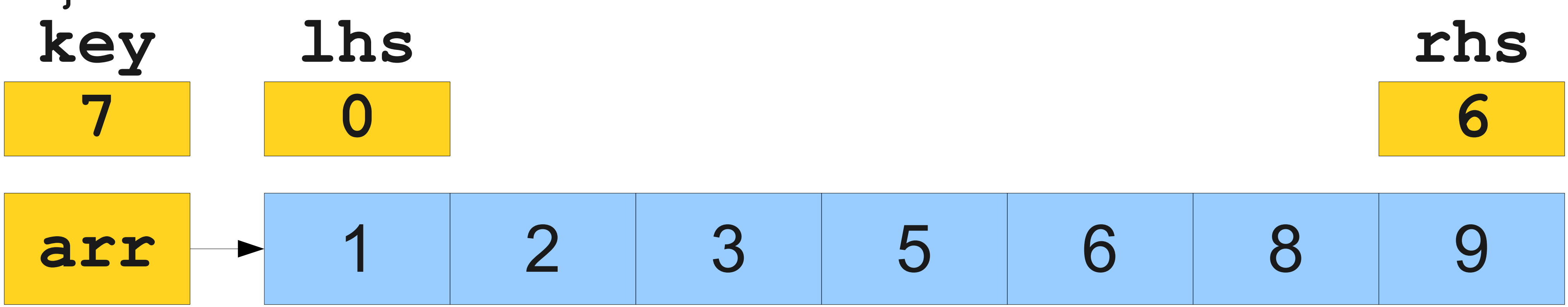
Binary Search

```
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;
}
```



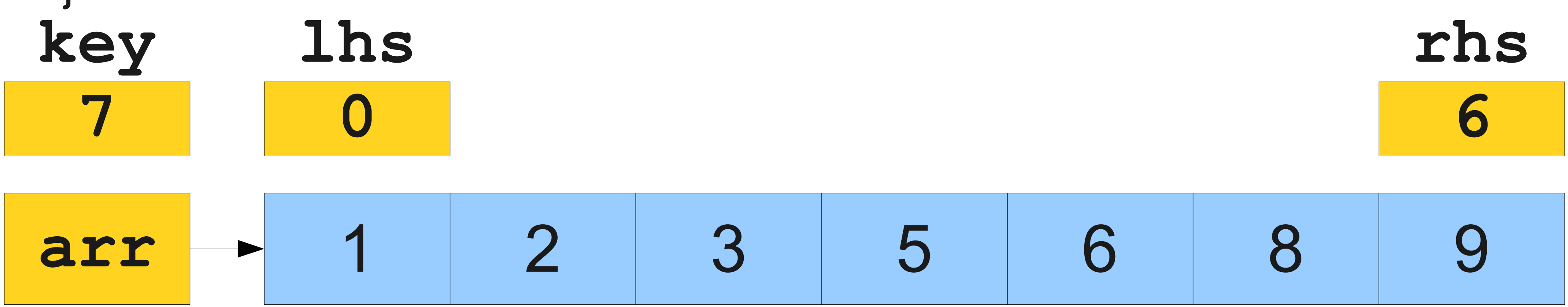
Binary Search

```
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;
}
```



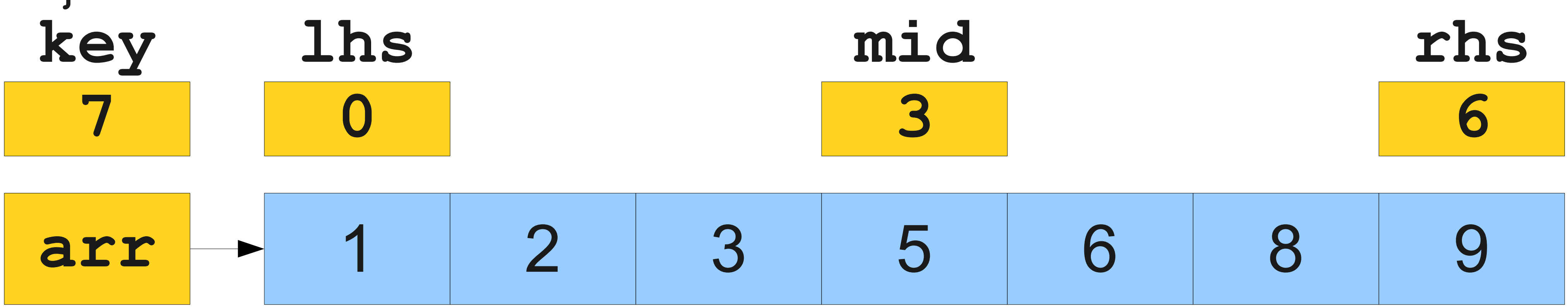
Binary Search

```
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;
}
```



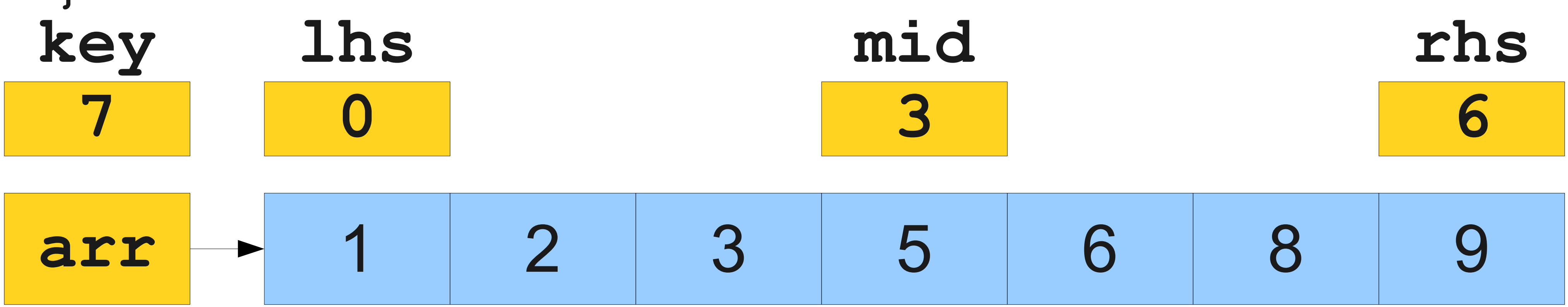
Binary Search

```
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;
}
```



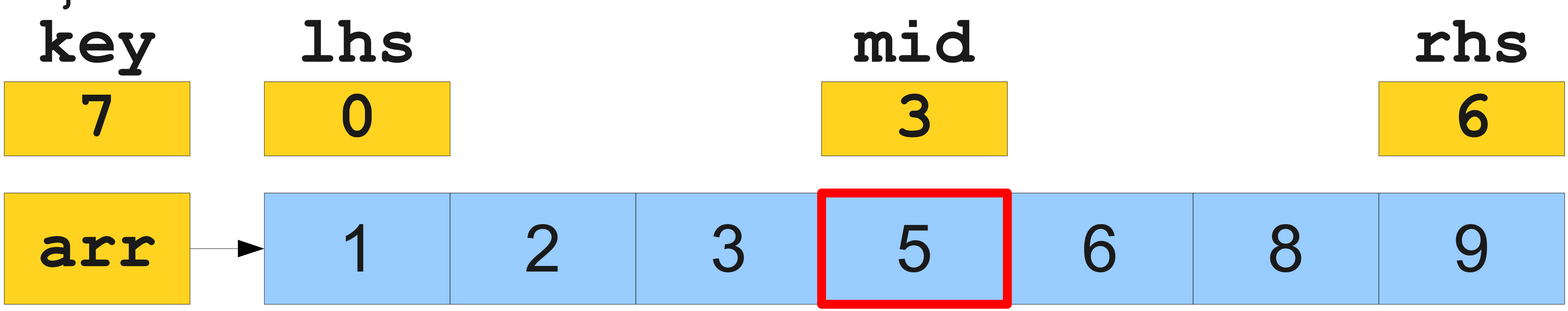
Binary Search

```
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;
}
```



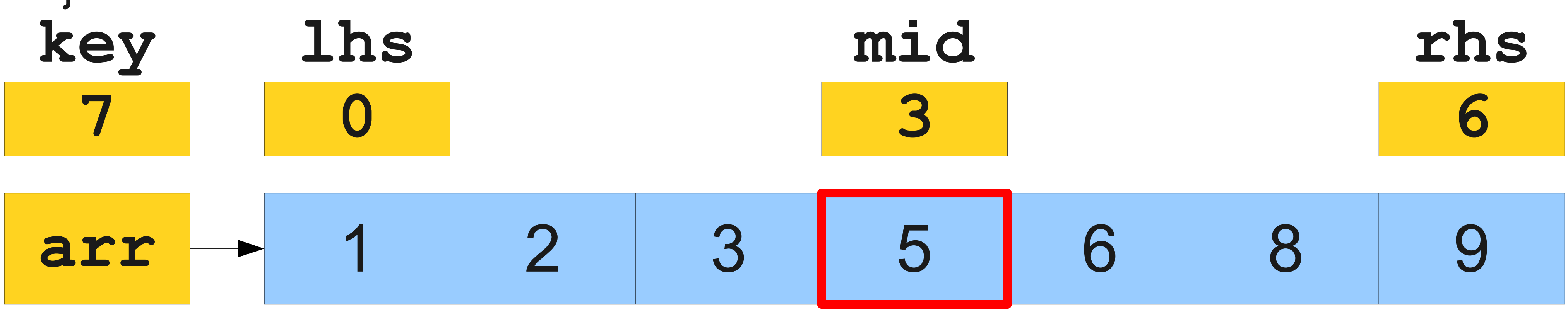
Binary Search

```
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;
}
```



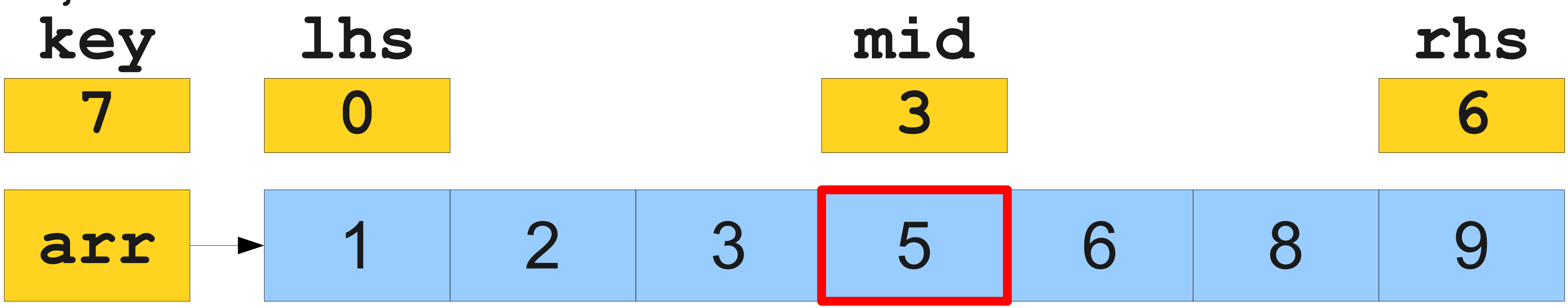
Binary Search

```
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;
}
```



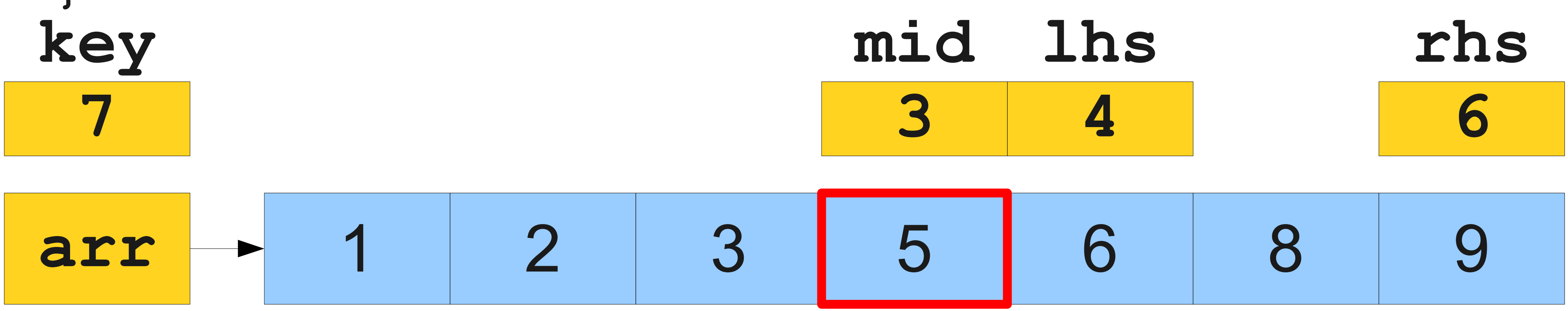
Binary Search

```
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;
}
```



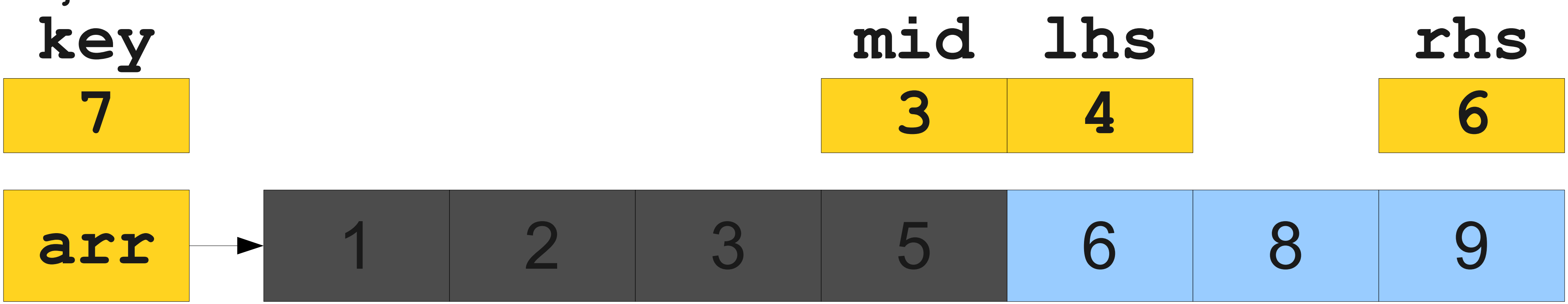
Binary Search

```
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;
}
```



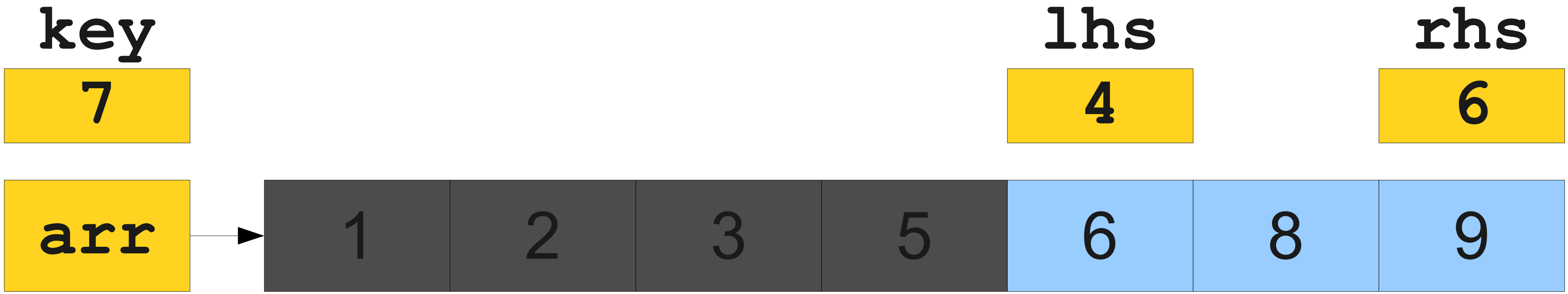
Binary Search

```
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;
}
```



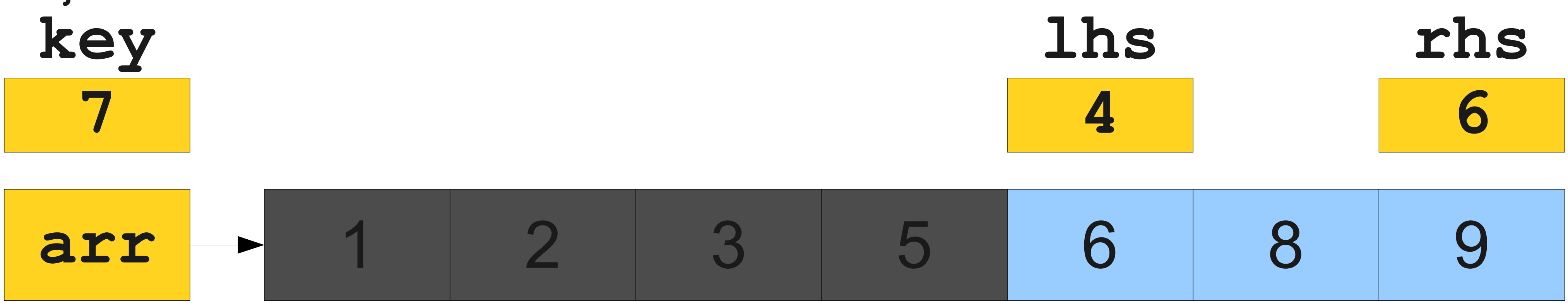
Binary Search

```
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;
}
```



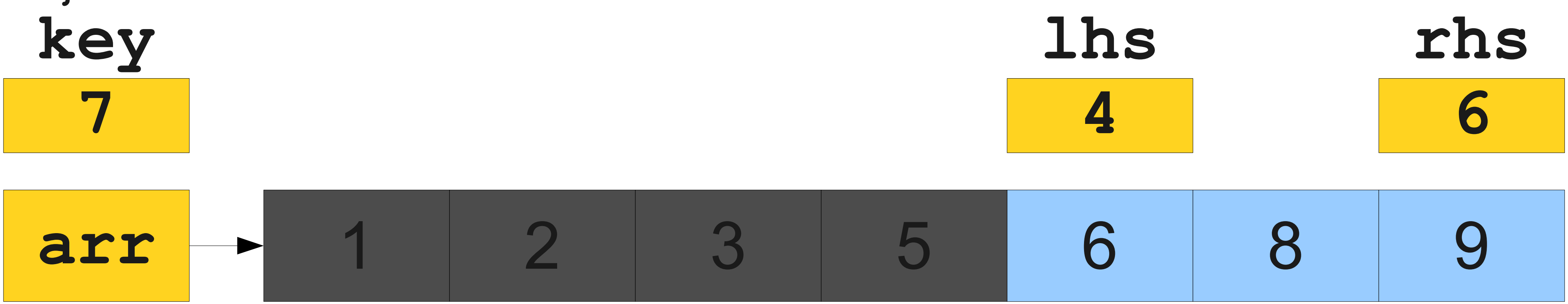
Binary Search

```
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;
}
```



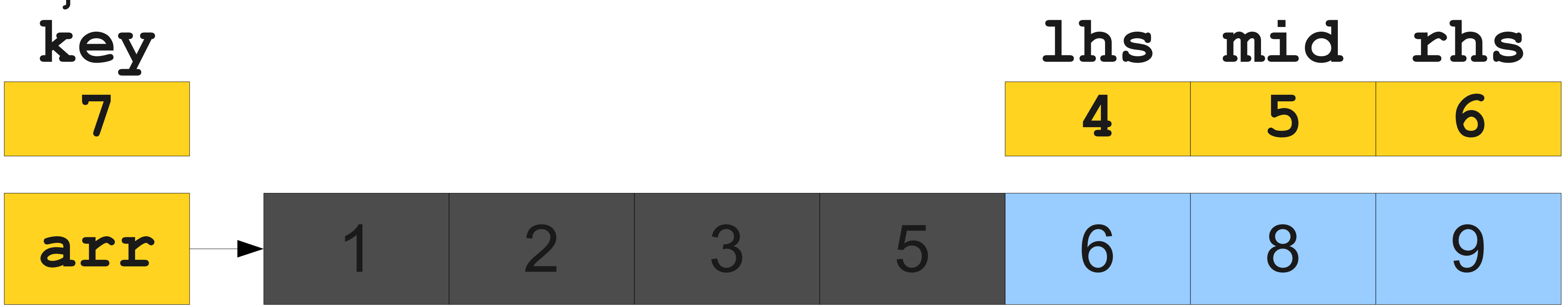
Binary Search

```
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;
}
```



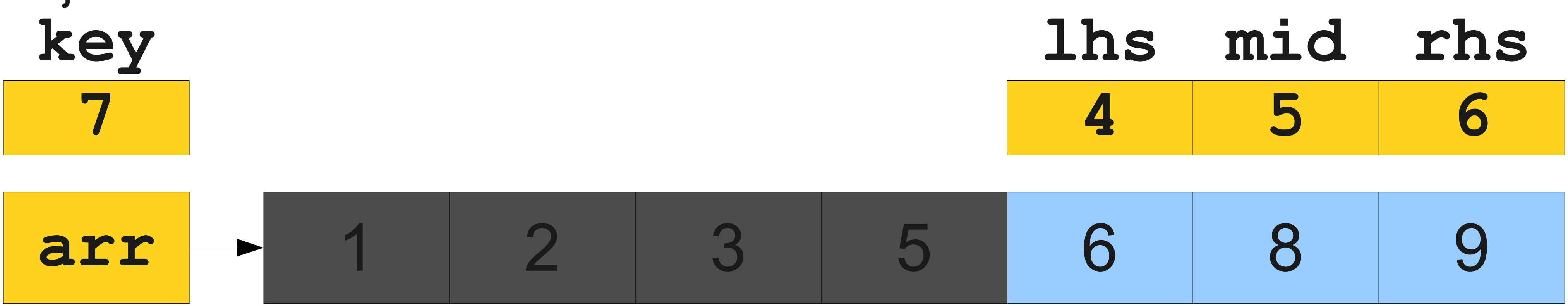
Binary Search

```java
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;
}
```



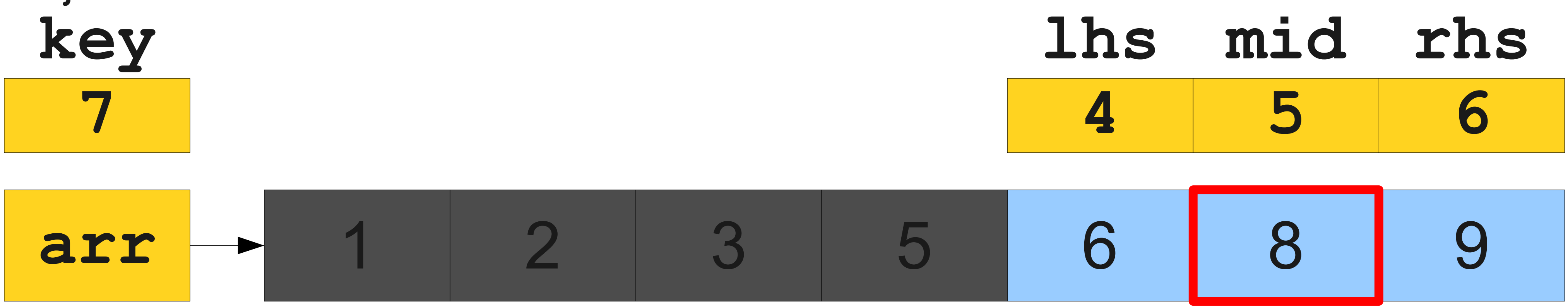
Binary Search

```
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;
}
```



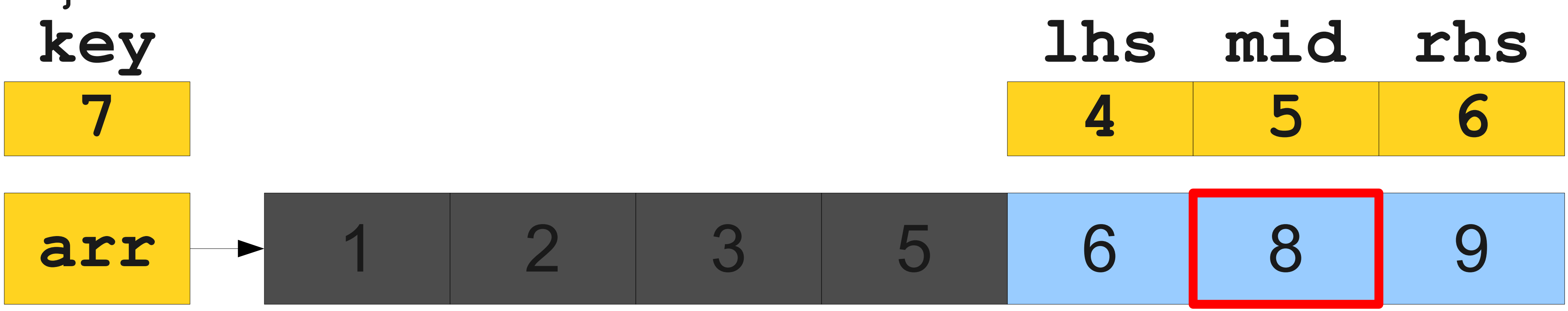
Binary Search

```
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;
}
```



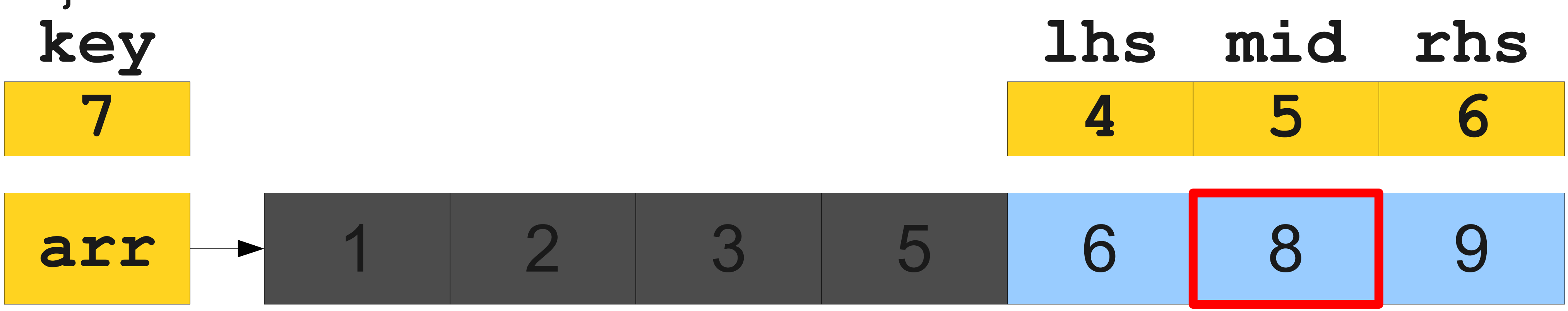
Binary Search

```
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;
}
```



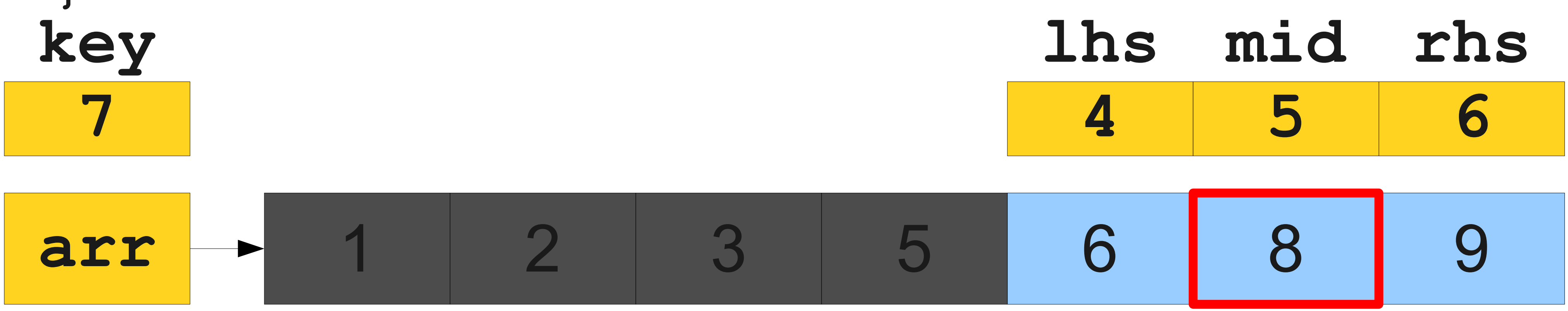
Binary Search

```
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;
}
```



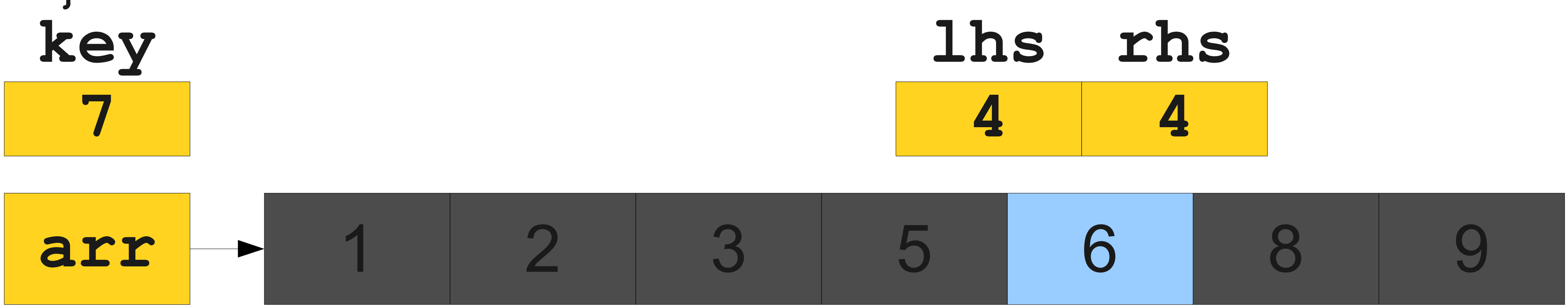
Binary Search

```
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;
}
```



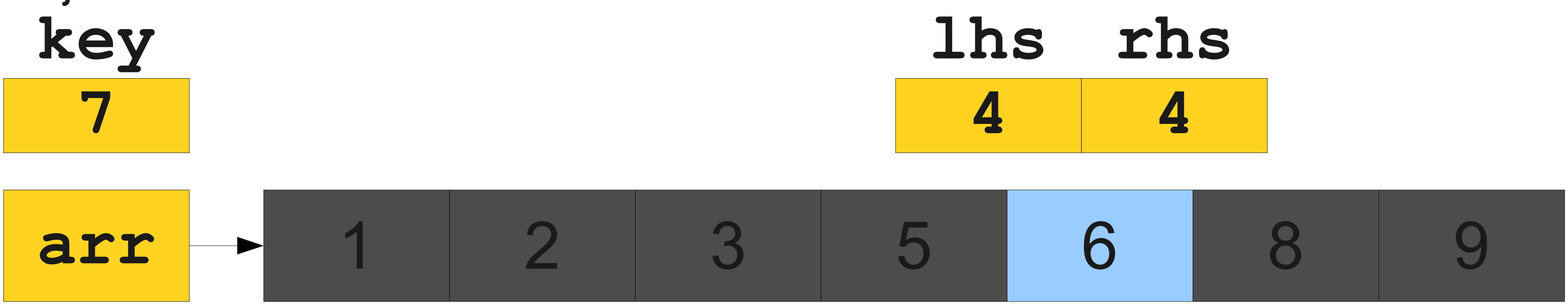
Binary Search

```java
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;
}
```



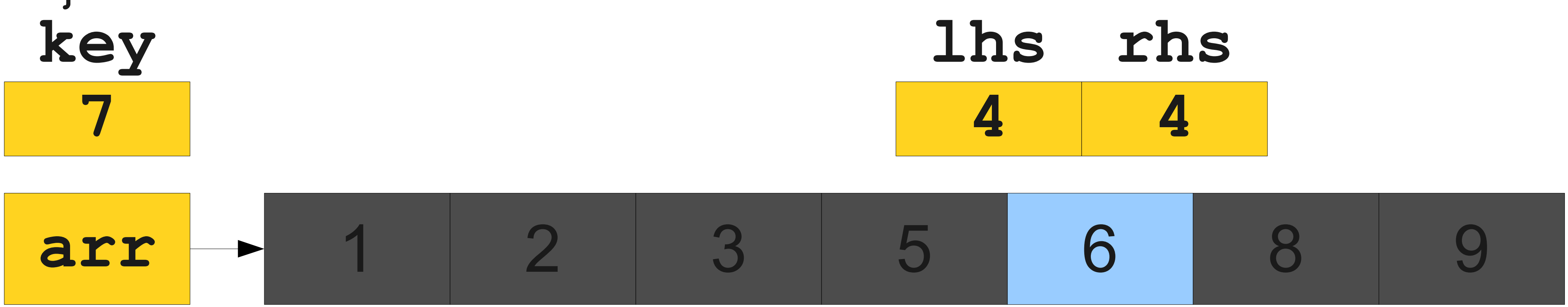
Binary Search

```
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;
}
```



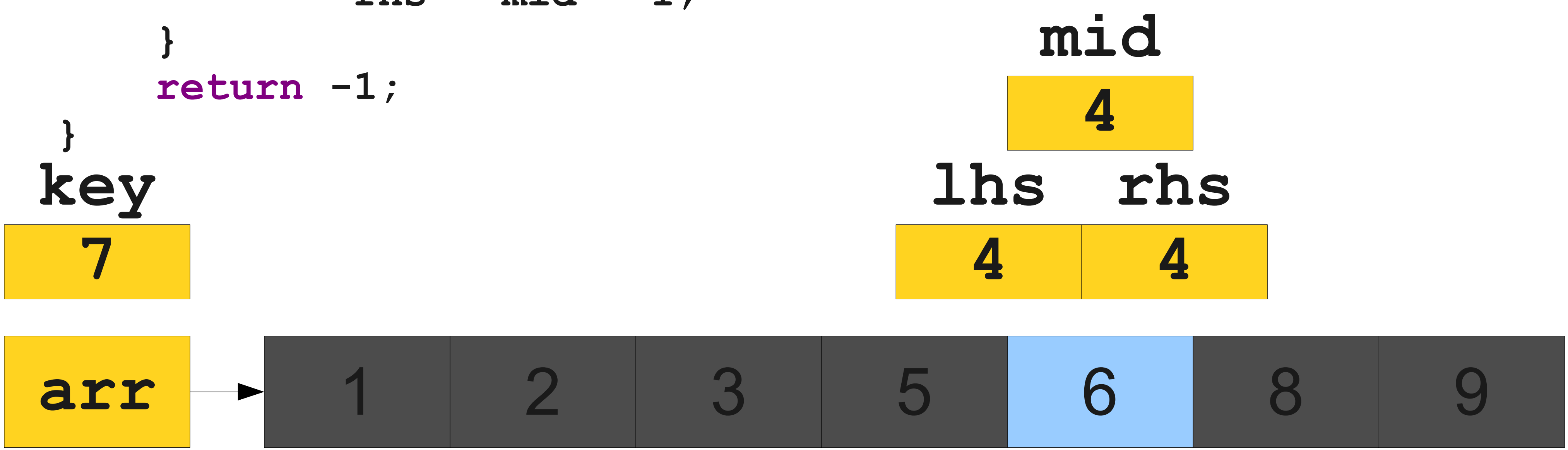
Binary Search

```
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;
}
```



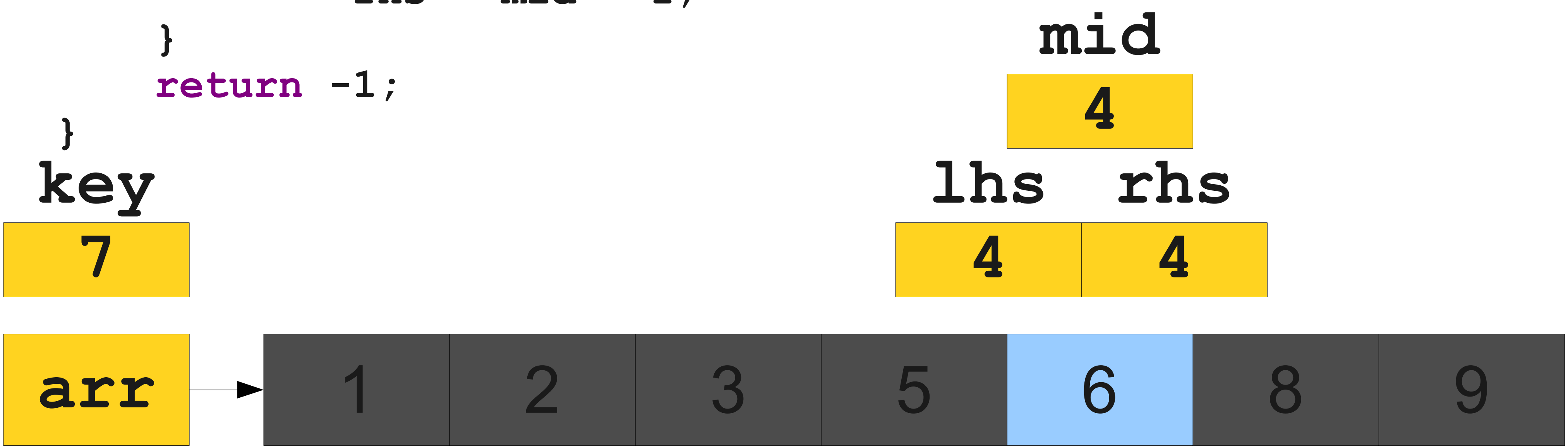
Binary Search

```
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;
}
```



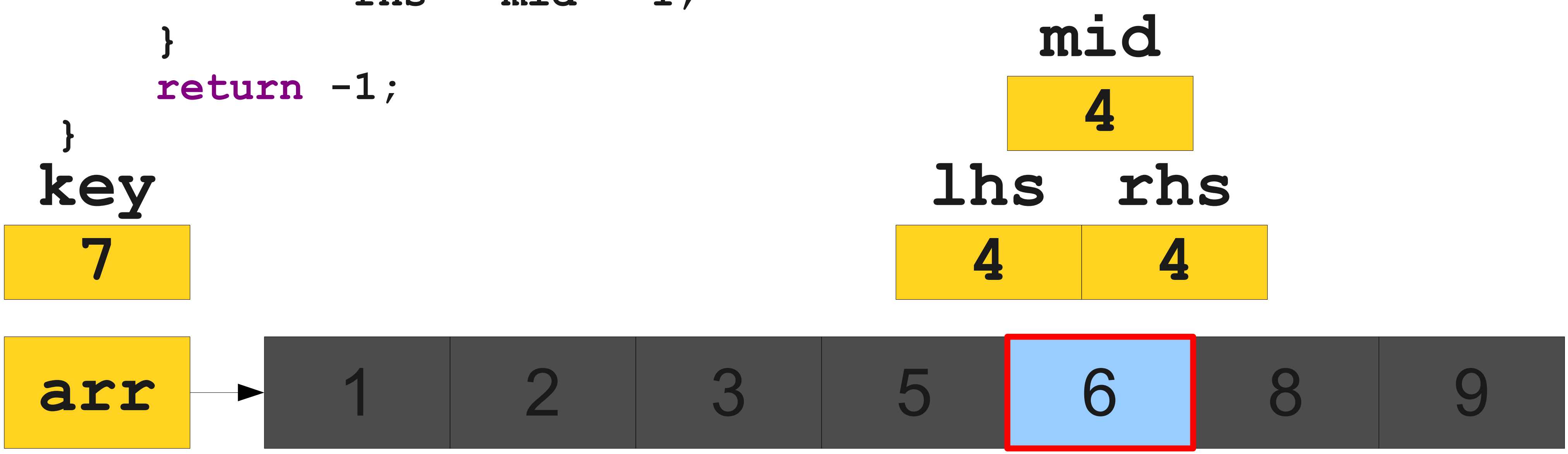
Binary Search

```
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;
}
```



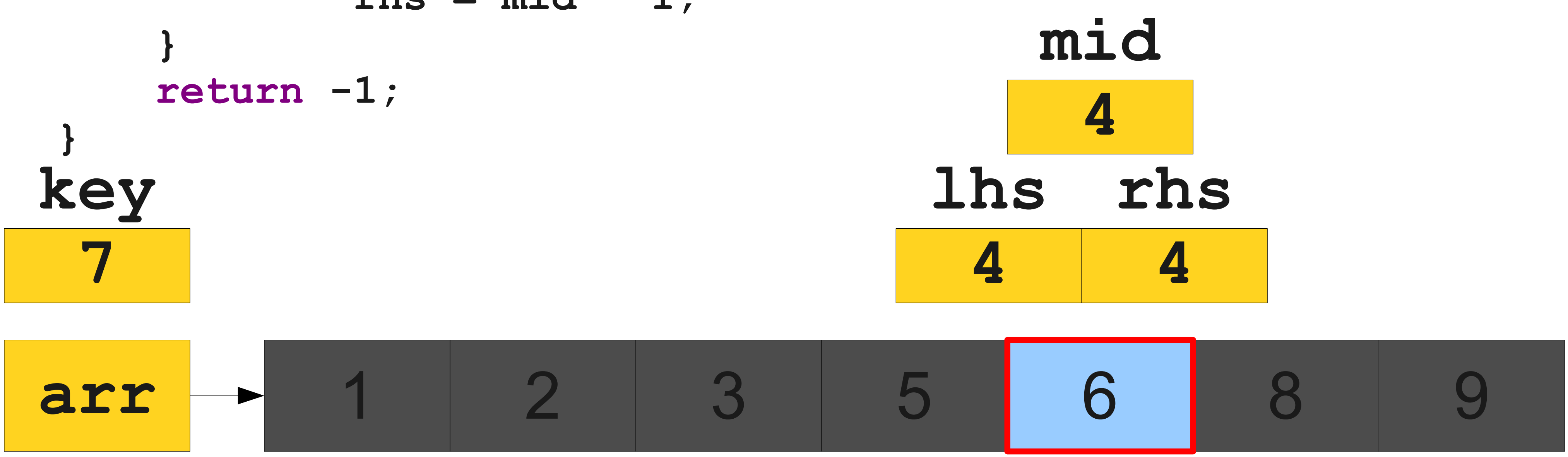
Binary Search

```java
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;
}
```



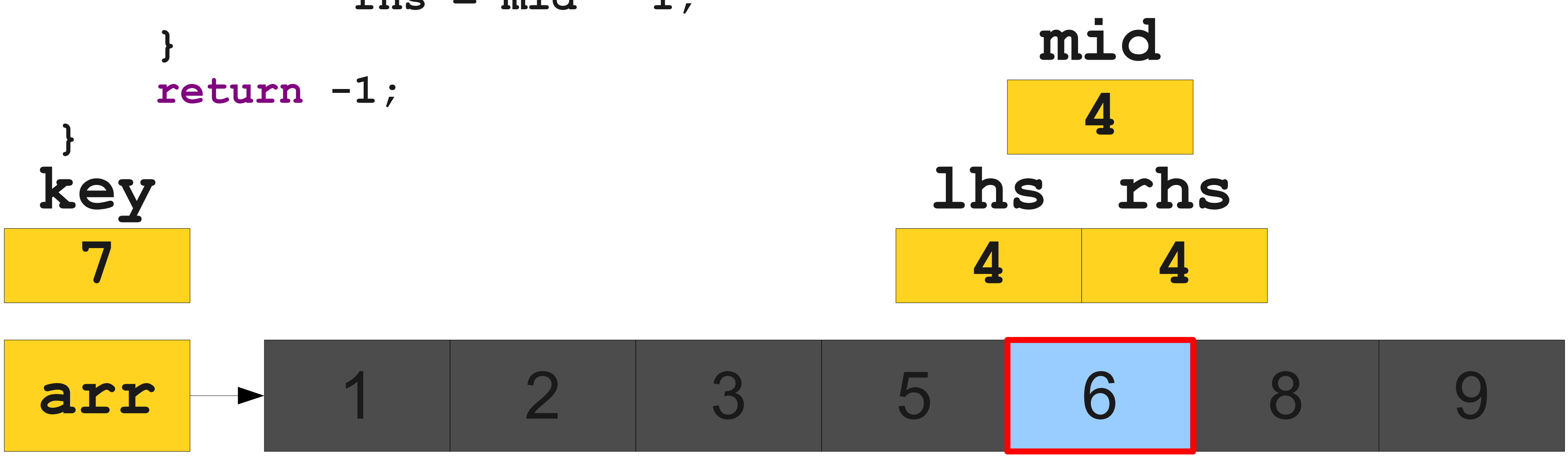
Binary Search

```
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;
}
```



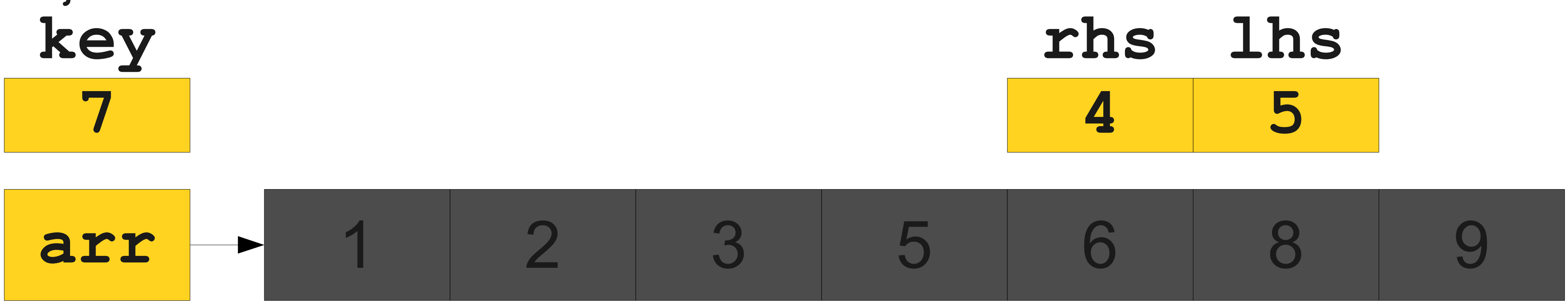
Binary Search

```
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;
}
```



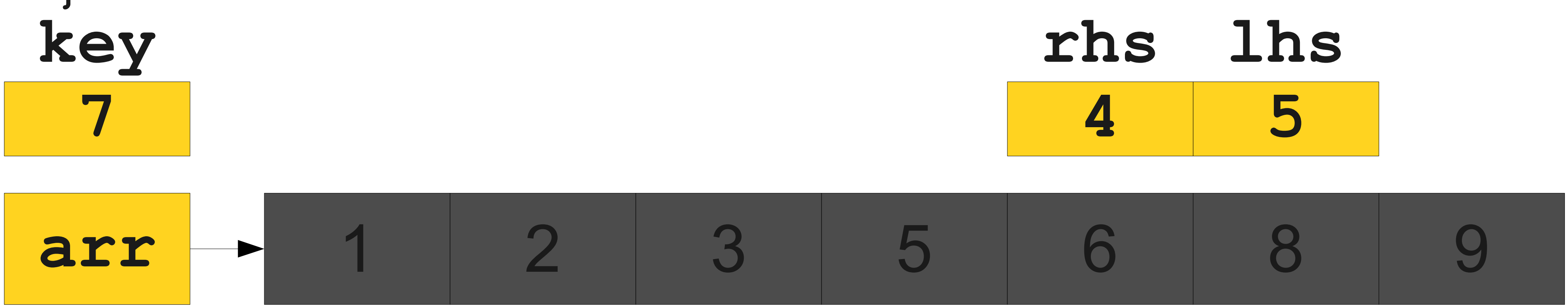
Binary Search

```
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;
}
```



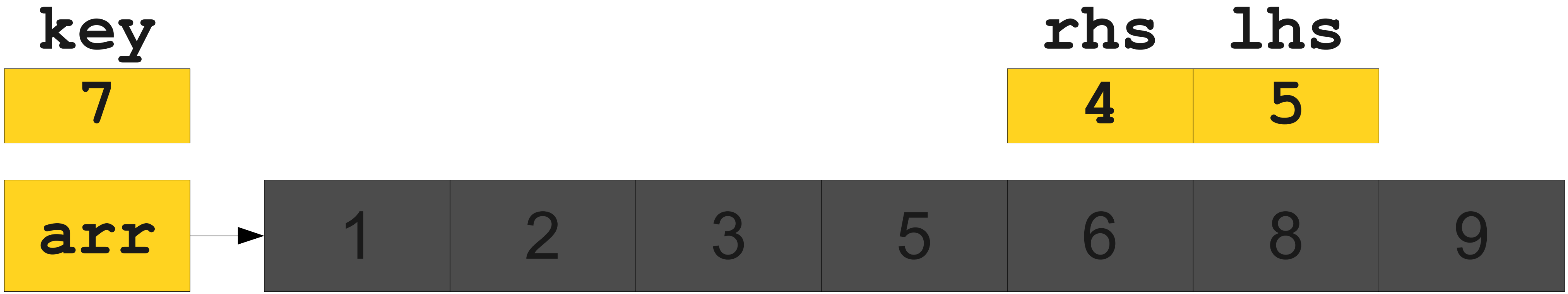
Binary Search

```
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;
}
```



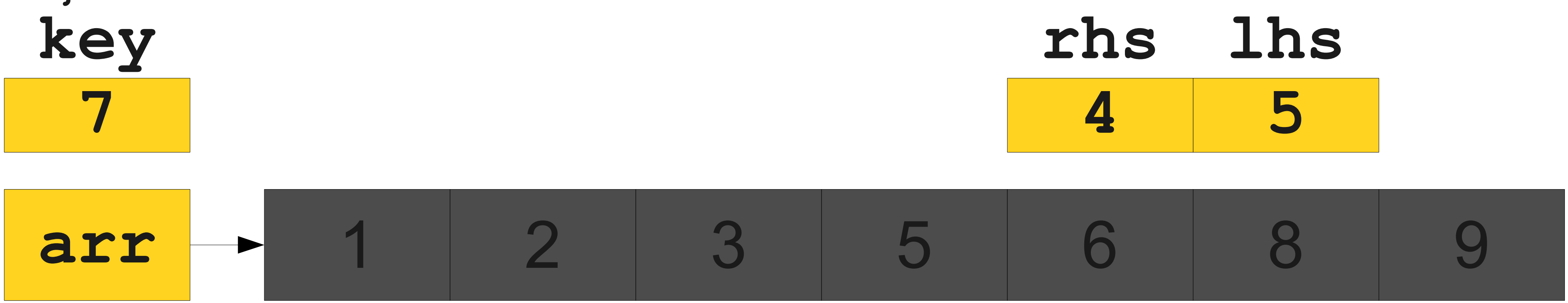
Binary Search

```
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) /

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



Analyzing the Algorithms

For Comparison

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

    return -1;
}
```

```java
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;
}
```

Some Quick Facts

- Elementary operations (arithmetic, choosing an if/else branch, deciding whether to loop again, etc.) all take roughly the same amount of time to complete.
- Array accesses take roughly the same time to complete regardless of the index.

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 $\mathbf{\log_2 N}$ 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}.
- $\log_2 10^{100} \approx 300$.
- 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.

Why Efficiency Matters

An Interesting Lecture Excerpt

Admiral Grace Hopper on nanoseconds:

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

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!

Barack Obama, in 2008.

Eric Schmidt, then CEO of Google.

A Presidential Decree

“The bubble sort would be the wrong way to go.”

- Barack Obama

What other algorithms could we use instead?

http://www.whitehouse.gov/sites/default/files/imagecache/admin official lowres/administration-official/ao image/President Official Portrait HiRes.jpg

Quicksort

Yes, This Exists

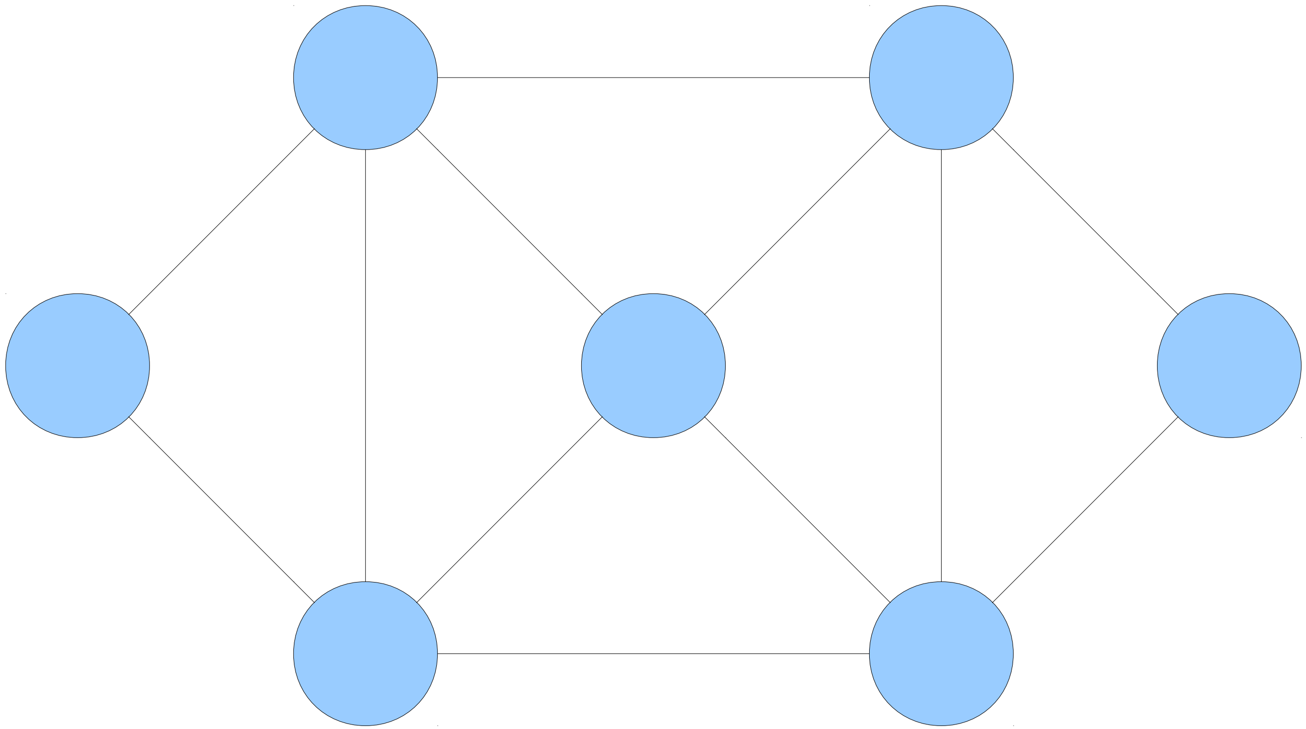
Sorting algorithms done as
Hungarian folk dances:

http://www.youtube.com/user/AlgoRythmics

Theoretical Limits of Efficiency

		7		6		1		
					3		5	2
3			1		5	9		7
6		5		3		8		9
	1						2	
8		2		1		5		4
1		3	2		7			8
5	7		4					
		4		8		7		

2	5	7	9	6	4	1	8	3
4	9	1	8	7	3	6	5	2
3	8	6	1	2	5	9	4	7
6	4	5	7	3	2	8	1	9
7	1	9	5	4	8	3	2	6
8	3	2	6	1	9	5	7	4
1	6	3	2	5	7	4	9	8
5	7	8	4	9	6	2	3	1
9	2	4	3	8	1	7	6	5



If you can check a solution to a problem efficiently, can you necessarily solve that problem efficiently?

This is called the **P versus NP problem** and is the biggest open problem in theoretical computer science.

There is a $1,000,000 prize for the answer.