Searching and Sorting

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

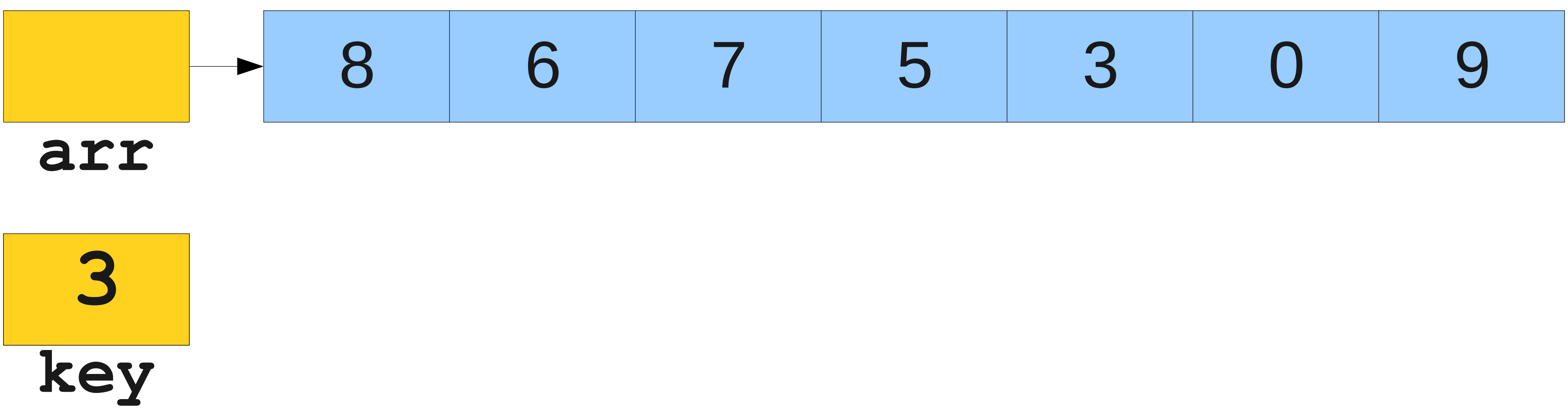
Linear Search

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;
}
```

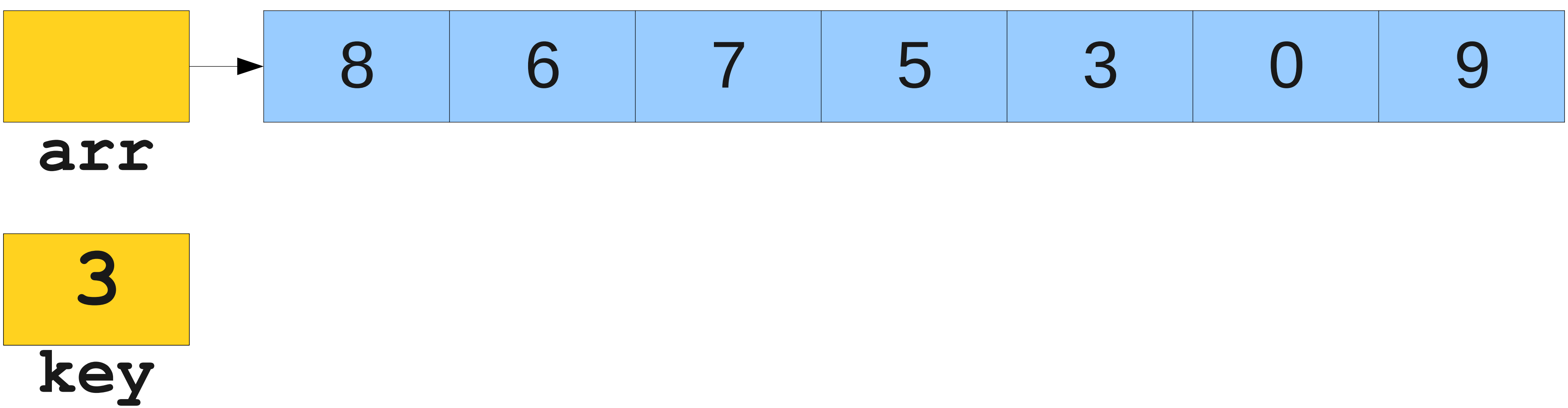
Linear Search

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

    return -1;
}
```



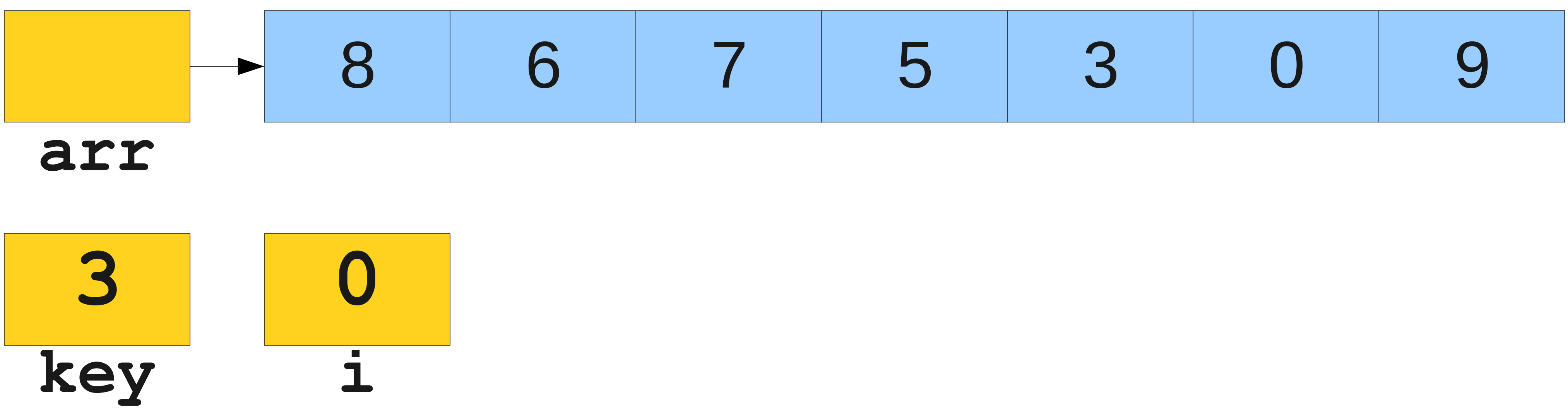
Linear Search

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

    return -1;
}
```



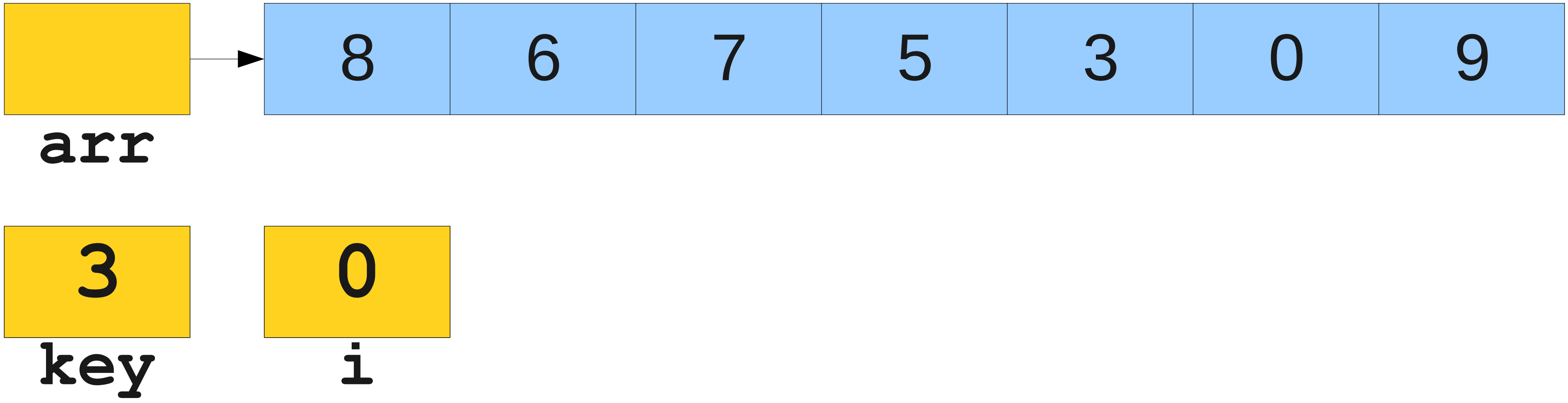
Linear Search

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

    return -1;
}
```



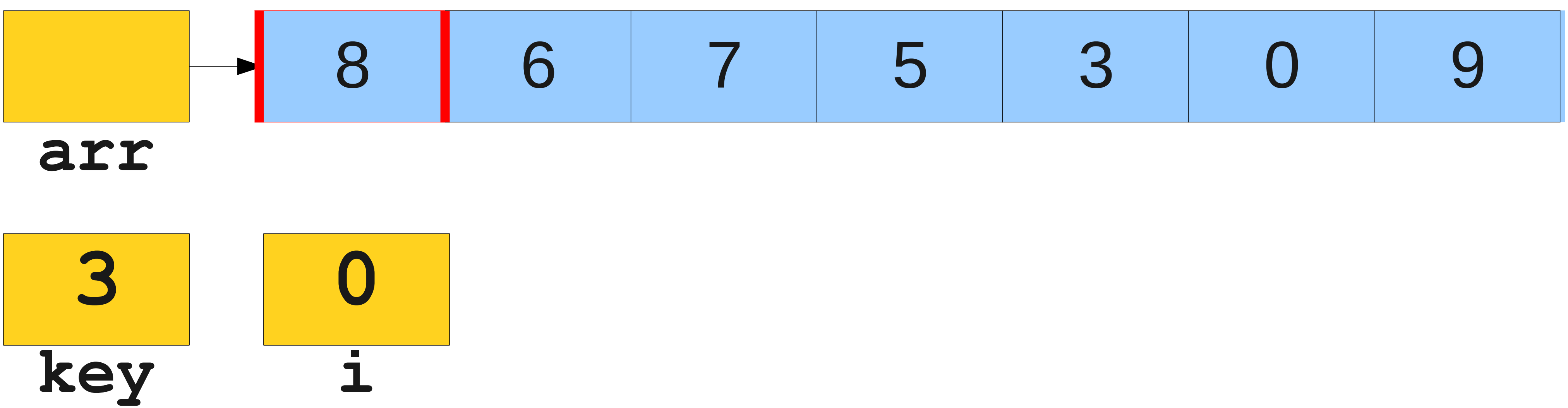
Linear Search

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

    return -1;
}
```



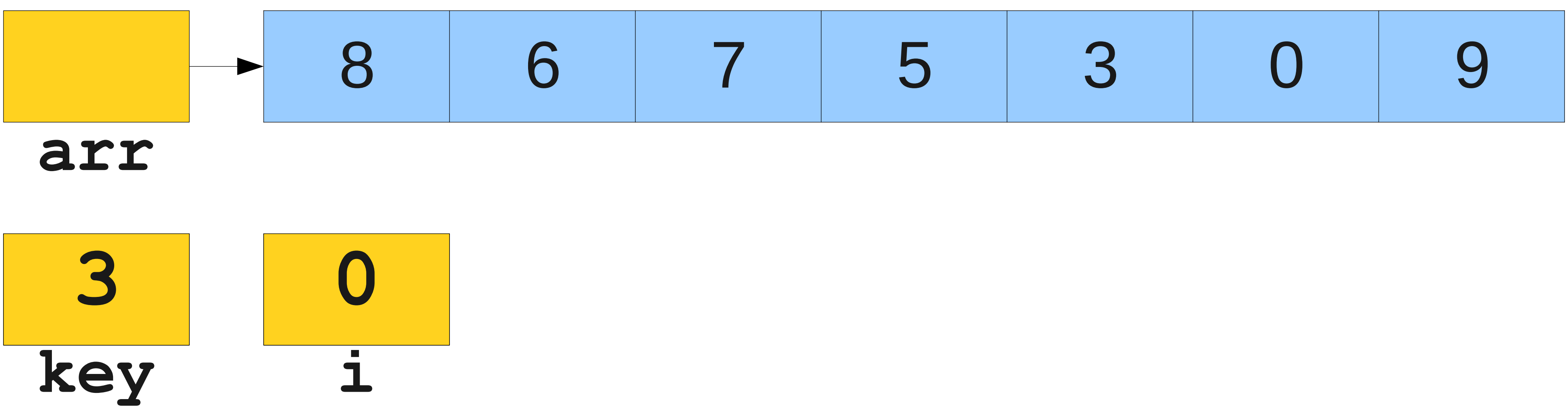
Linear Search

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

    return -1;
}
```



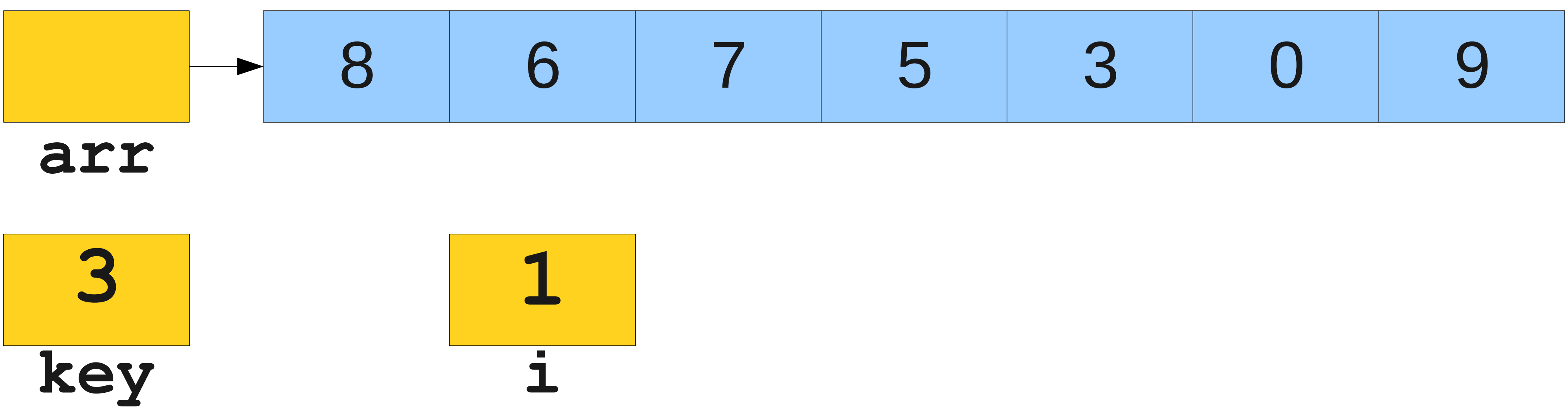
Linear Search

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

    return -1;
}
```



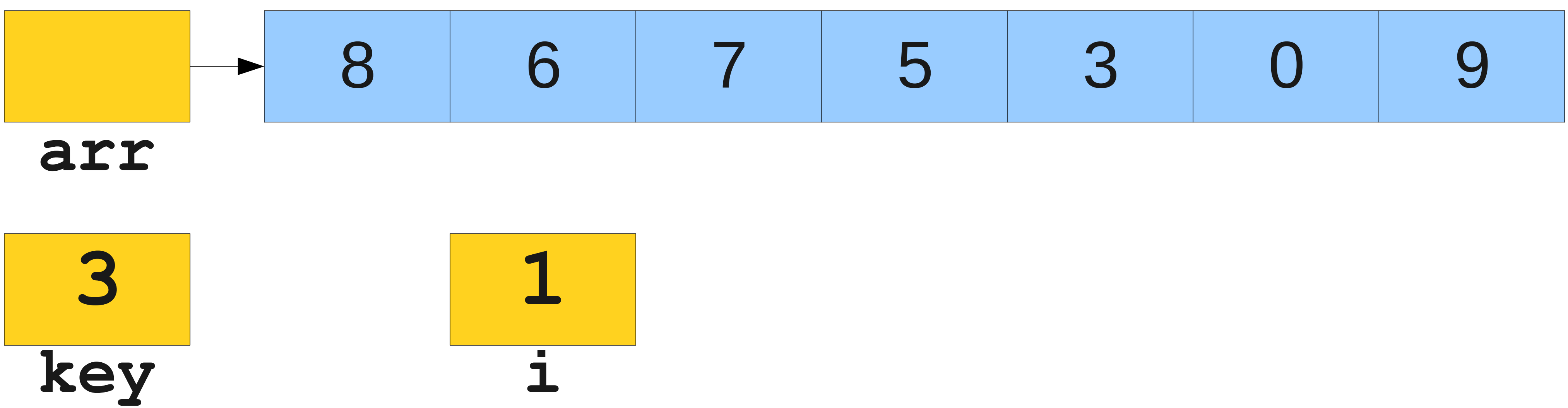
Linear Search

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

    return -1;
}
```



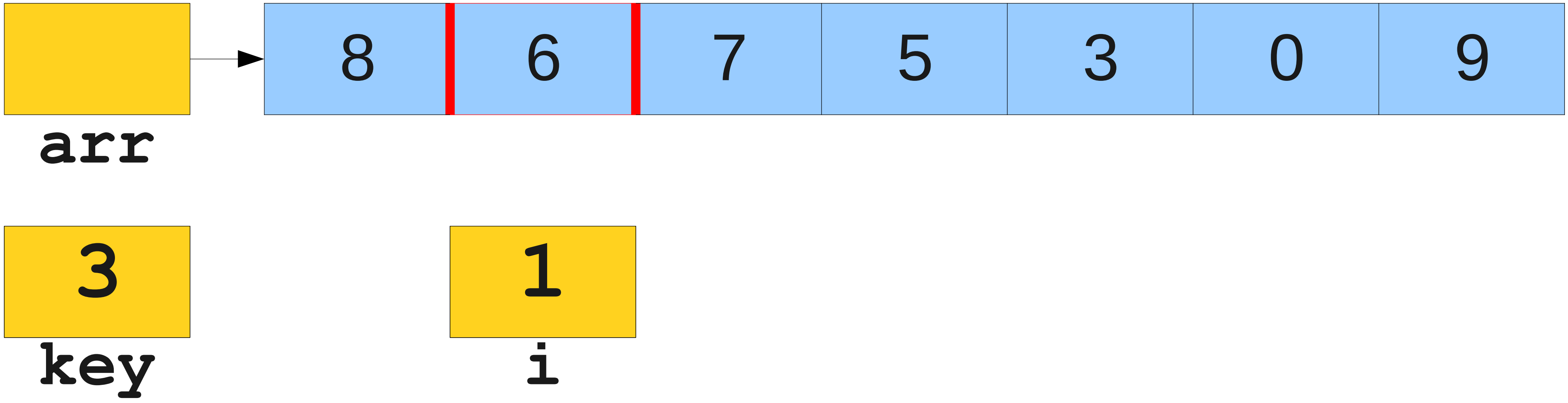
Linear Search

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

    return -1;
}
```



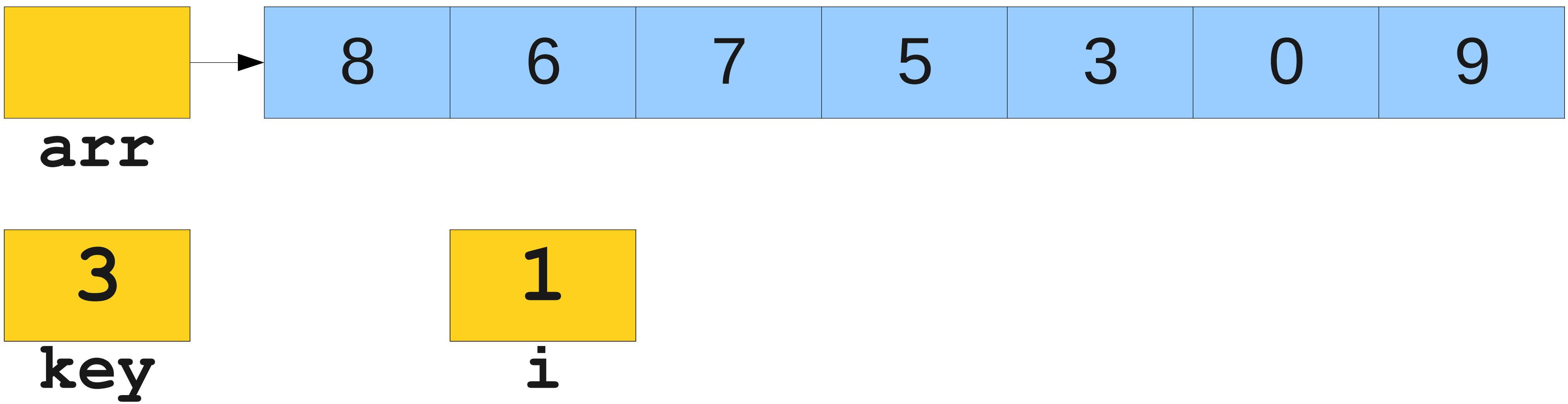
Linear Search

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

    return -1;
}
```



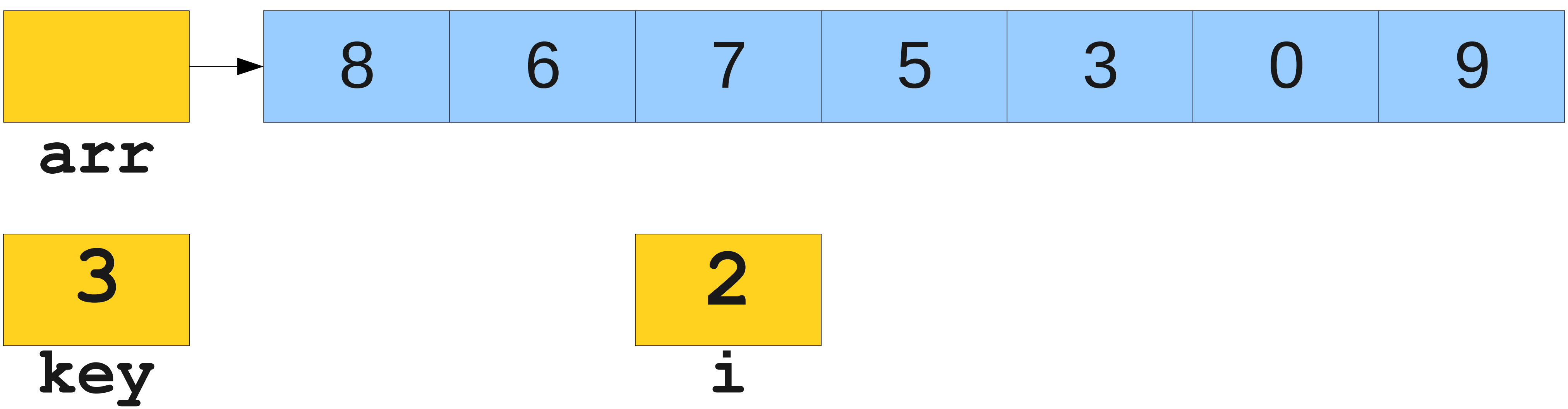
Linear Search

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

    return -1;
}
```



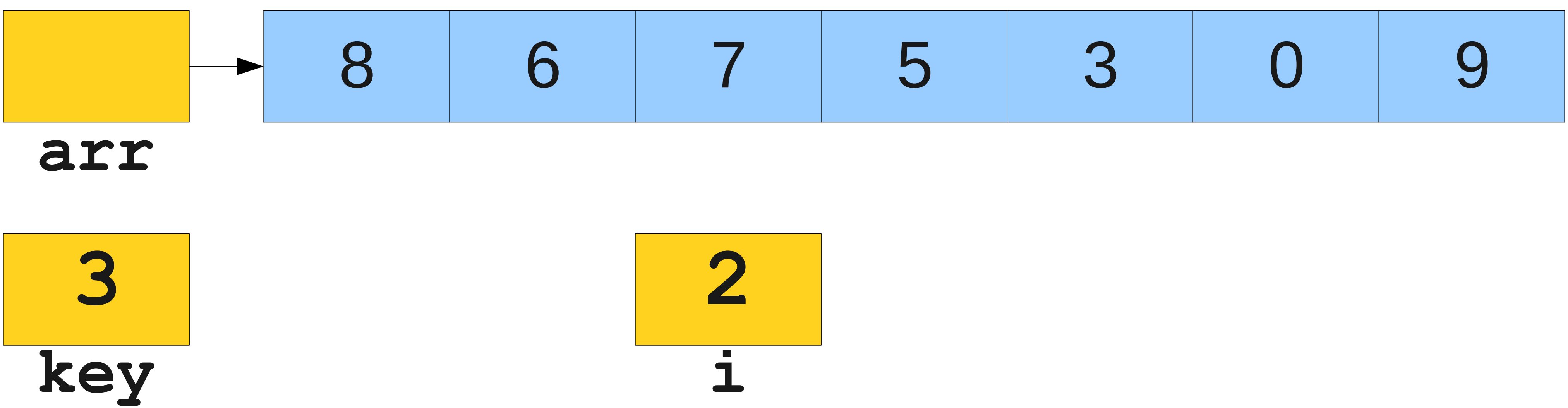
Linear Search

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

    return -1;
}
```



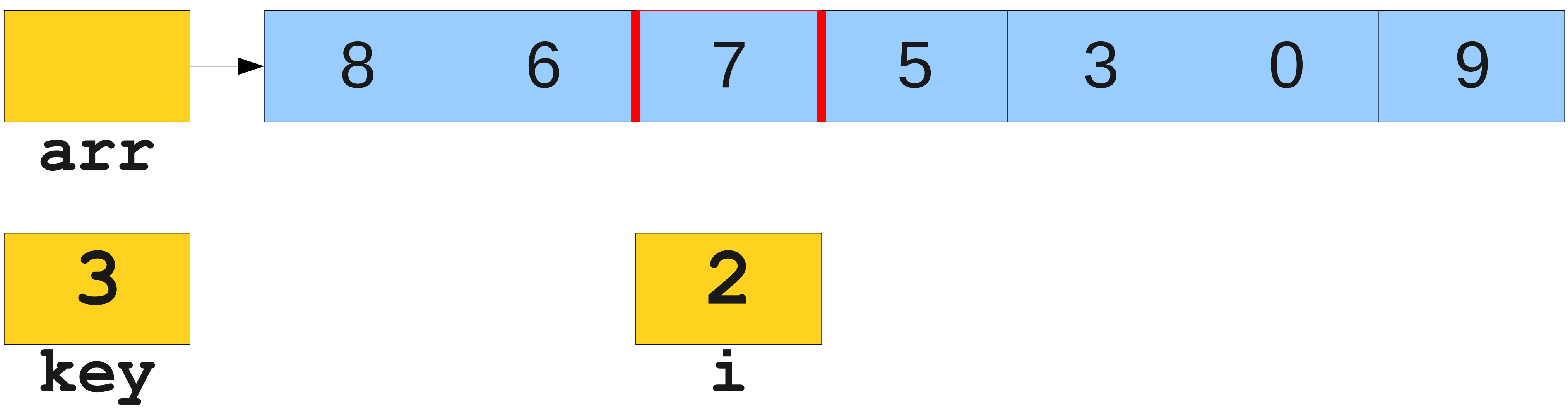
Linear Search

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

    return -1;
}
```



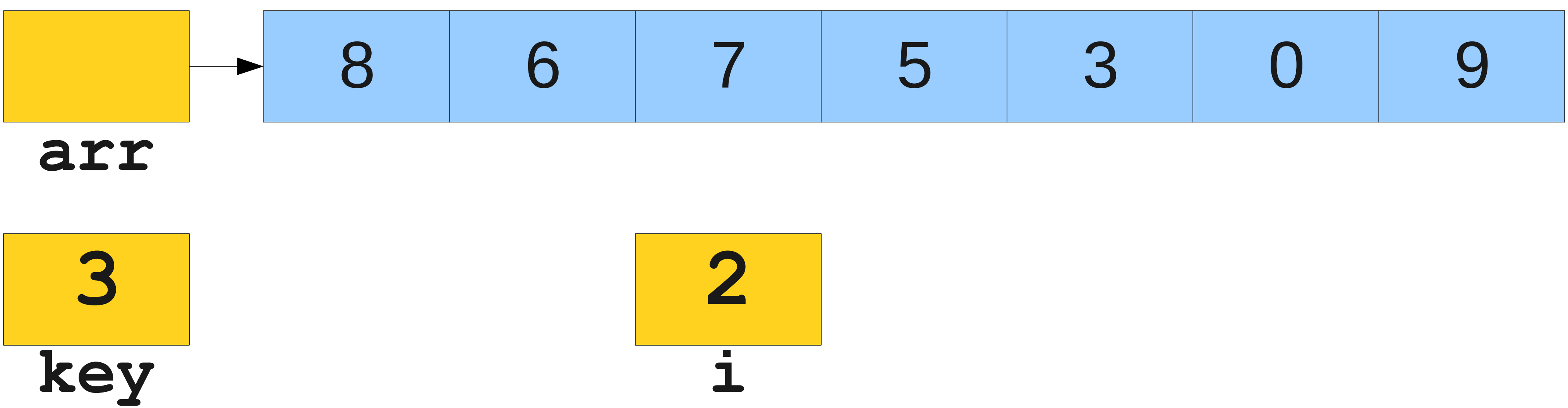
Linear Search

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

    return -1;
}
```



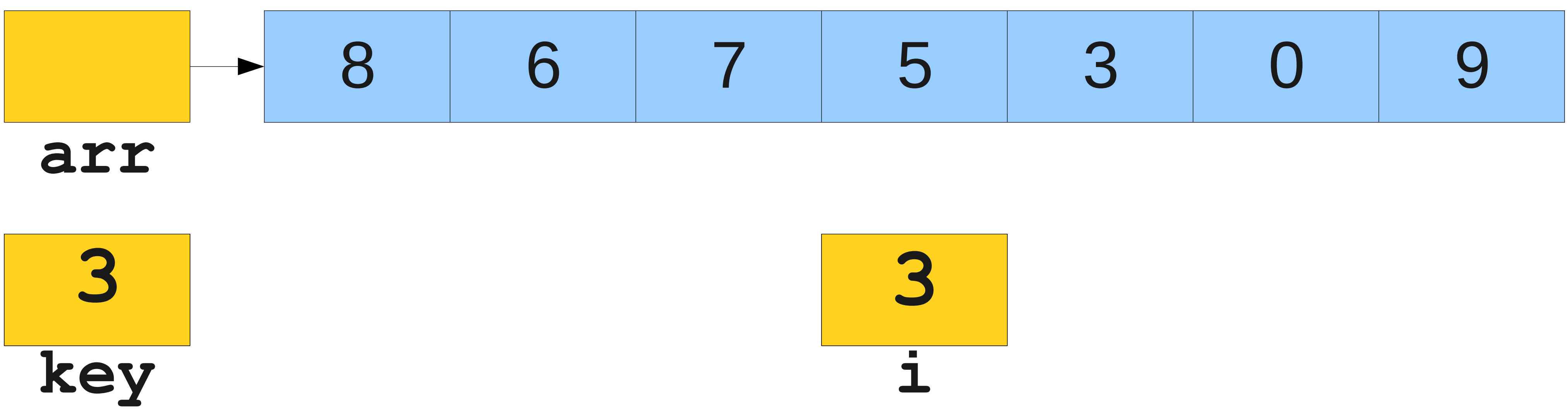
Linear Search

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

    return -1;
}
```



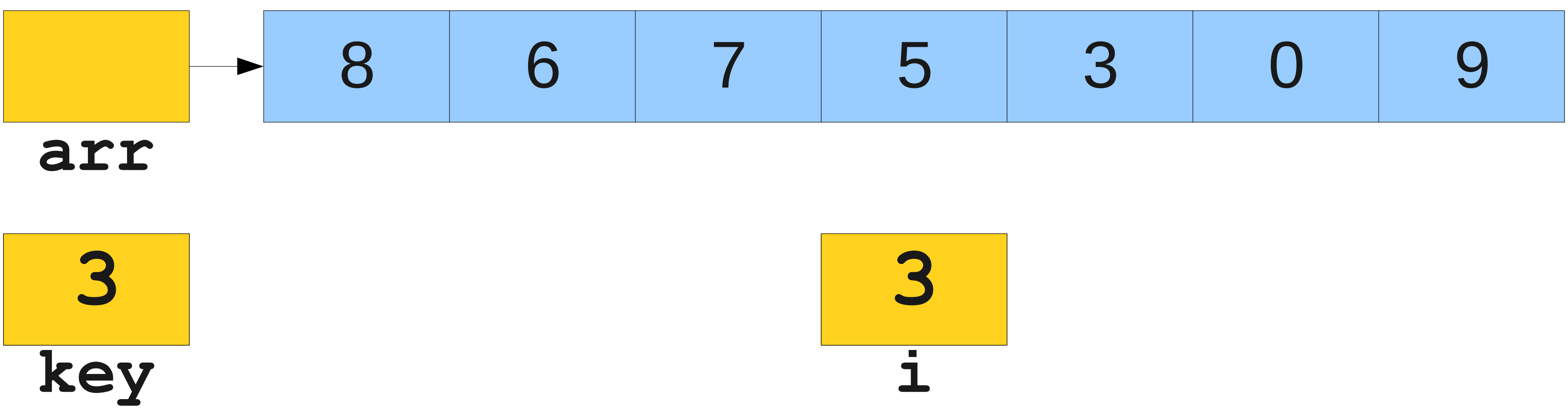
Linear Search

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

    return -1;
}
```



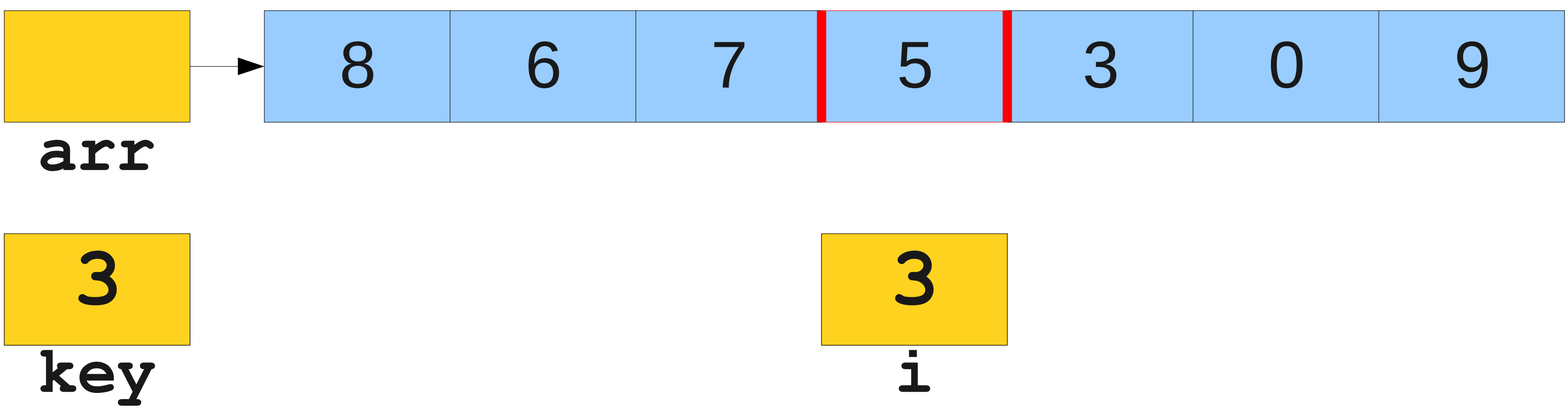
Linear Search

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

    return -1;
}
```



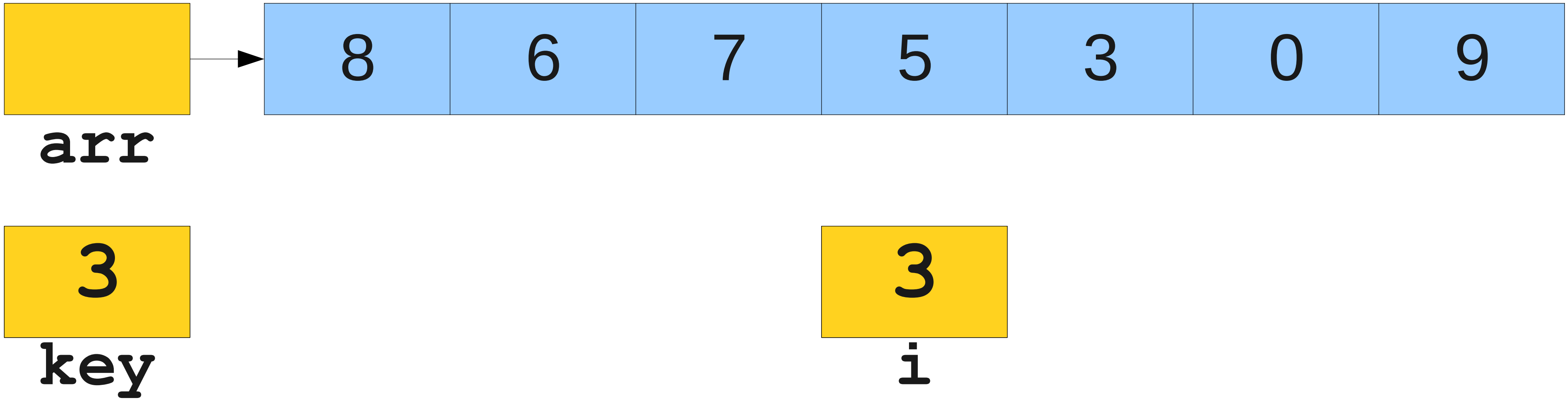
Linear Search

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

    return -1;
}
```



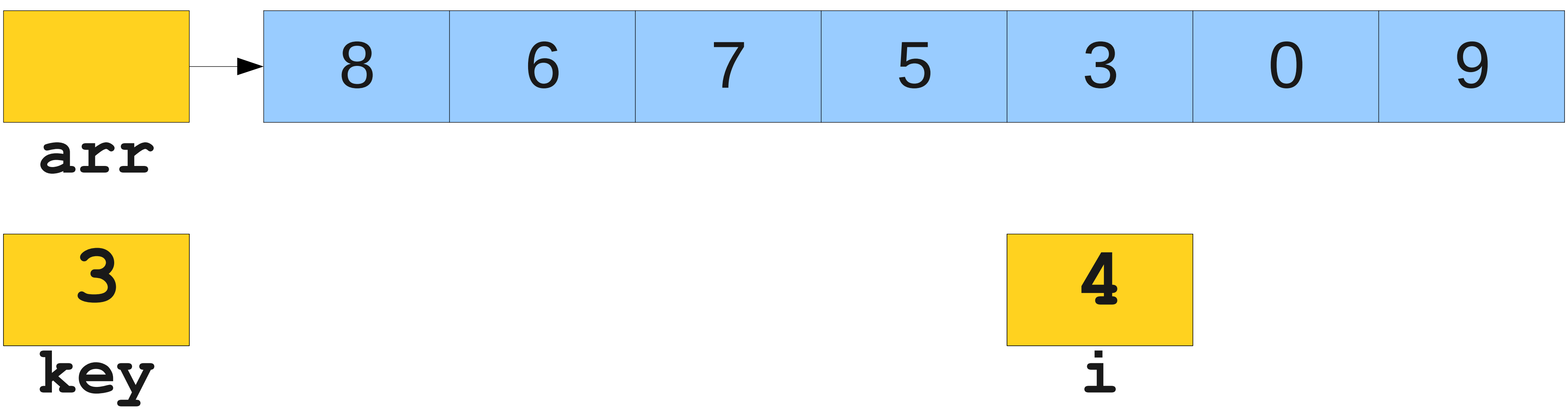
Linear Search

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

    return -1;
}
```



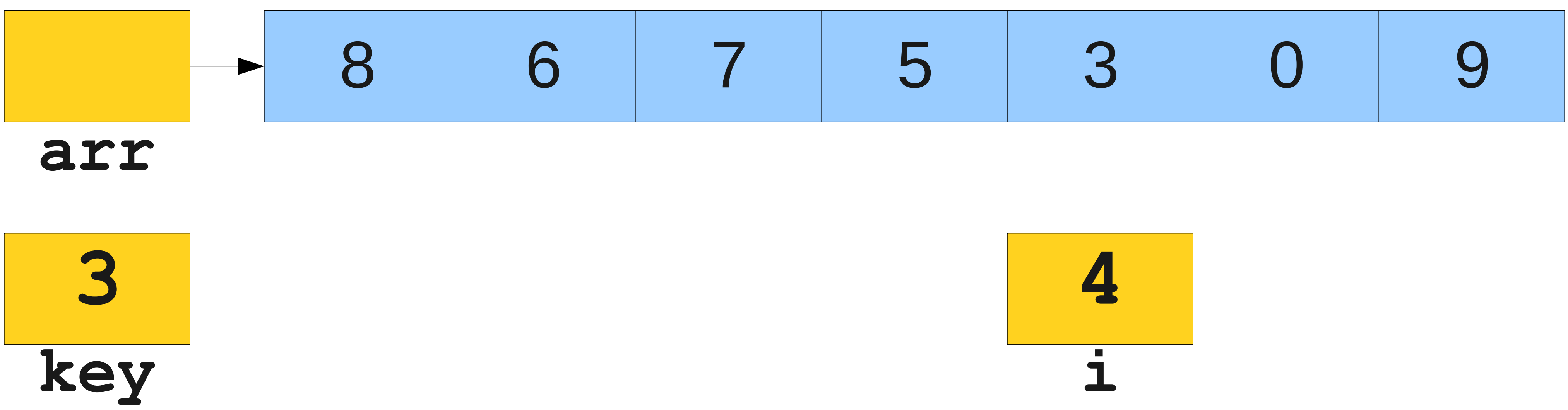
Linear Search

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

    return -1;
}
```



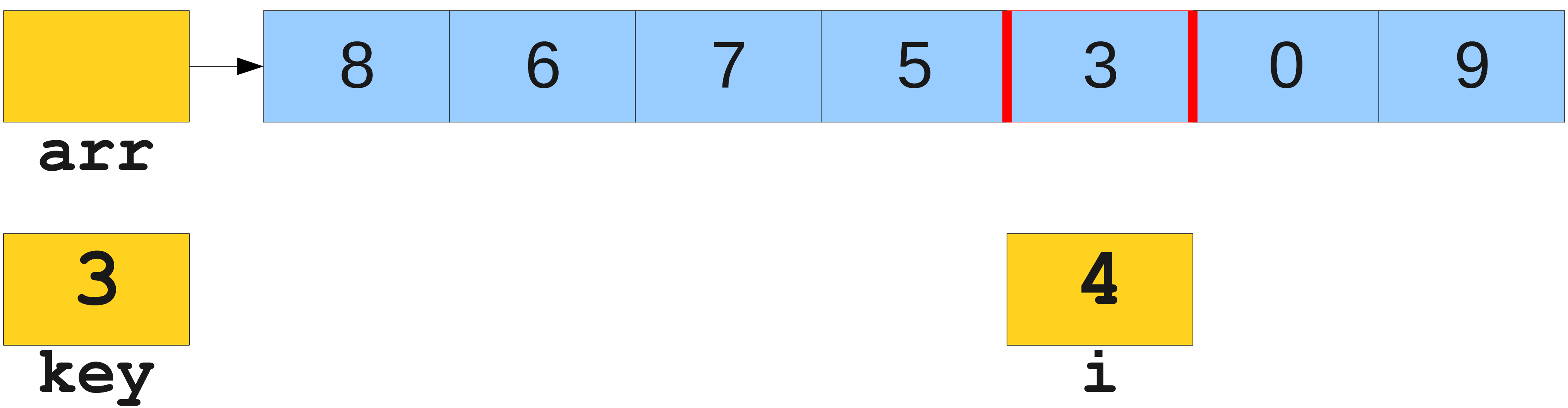
Linear Search

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

    return -1;
}
```



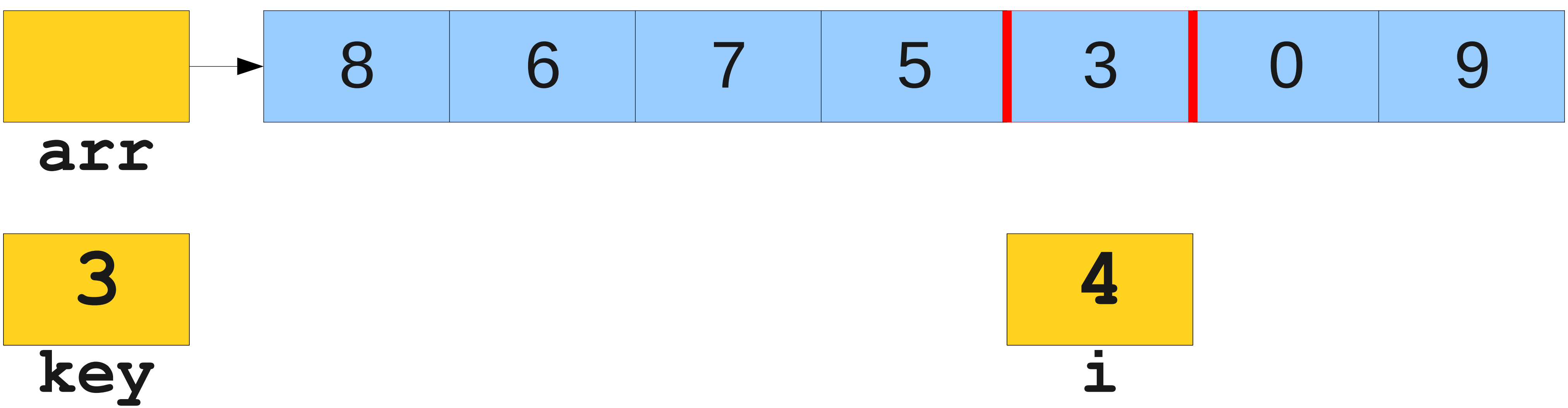
Linear Search

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

    return -1;
}
```



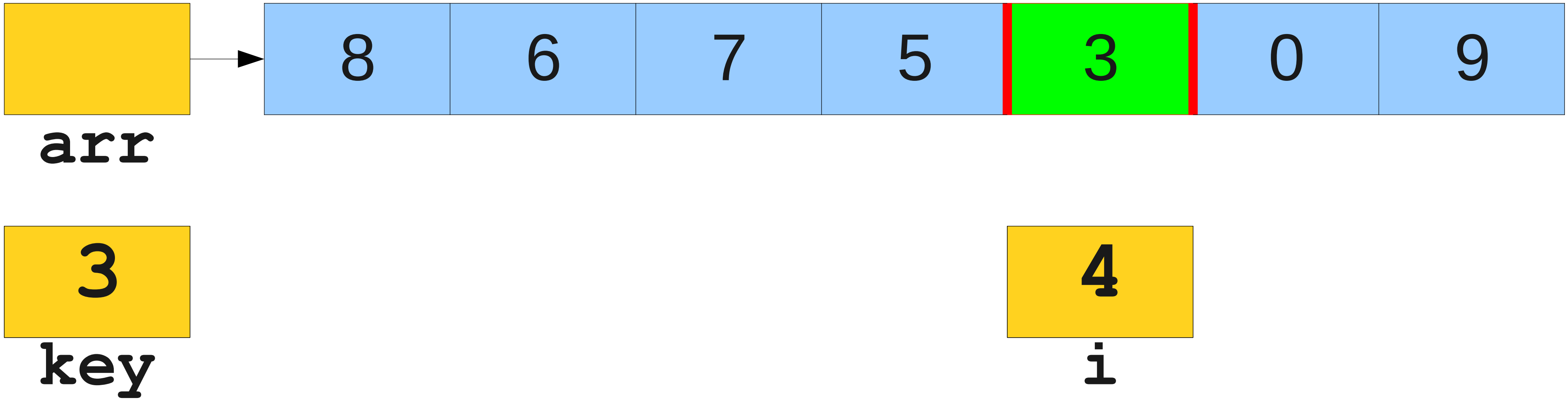
Linear Search

```
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;
}
```

a			3	0	9

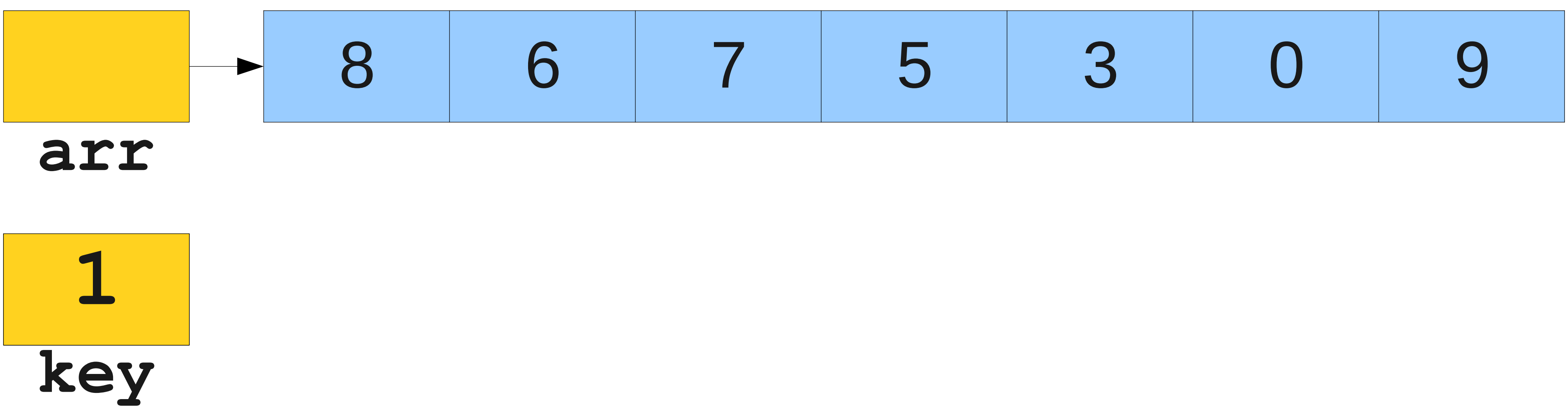
key	i
3	4

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;
}
```

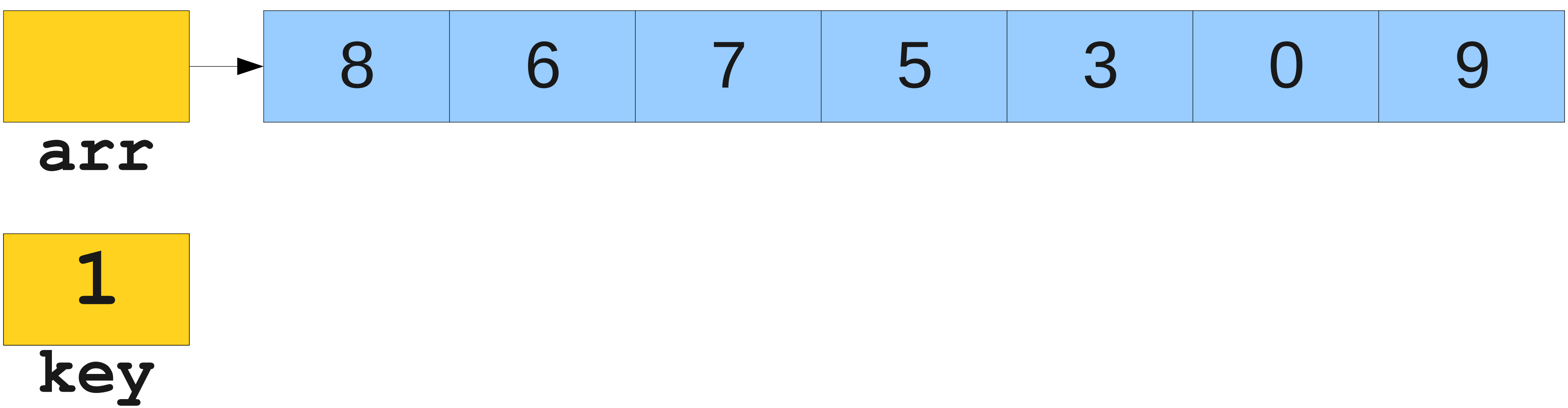
Linear Search

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

    return -1;
}
```



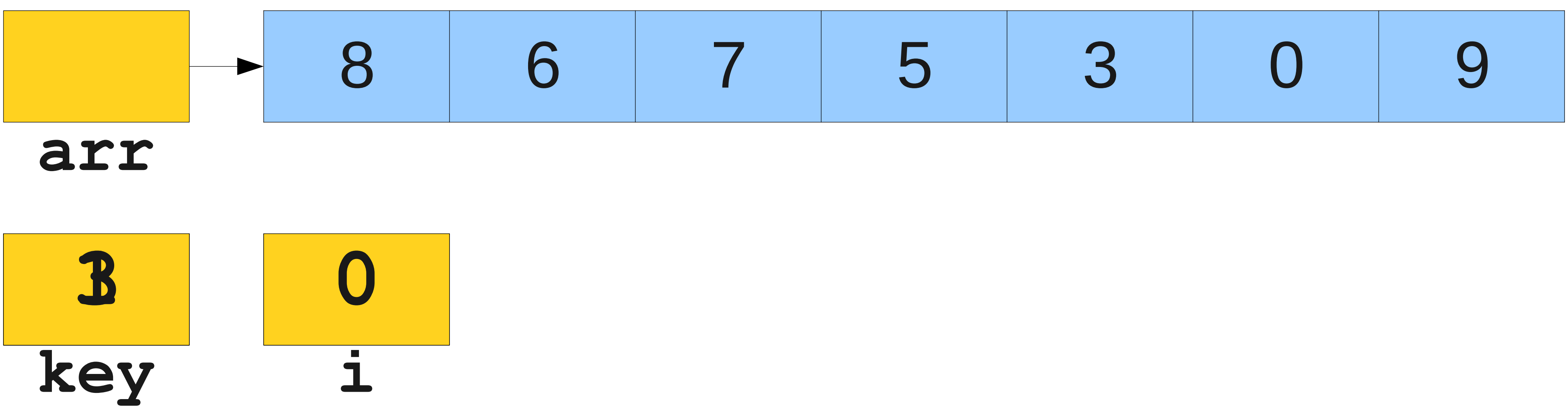
Linear Search

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

    return -1;
}
```



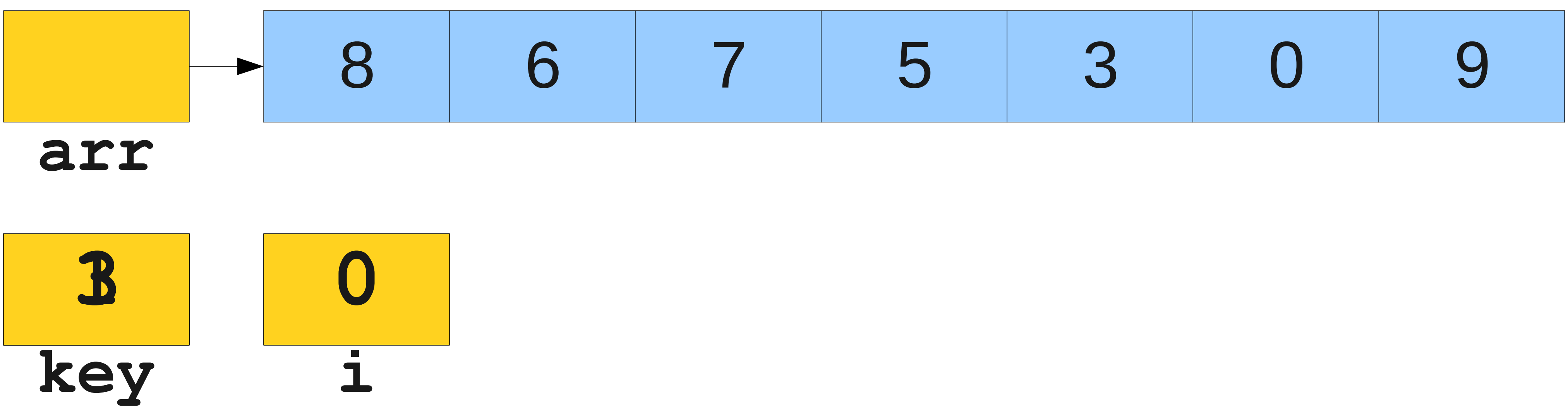
Linear Search

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

    return -1;
}
```



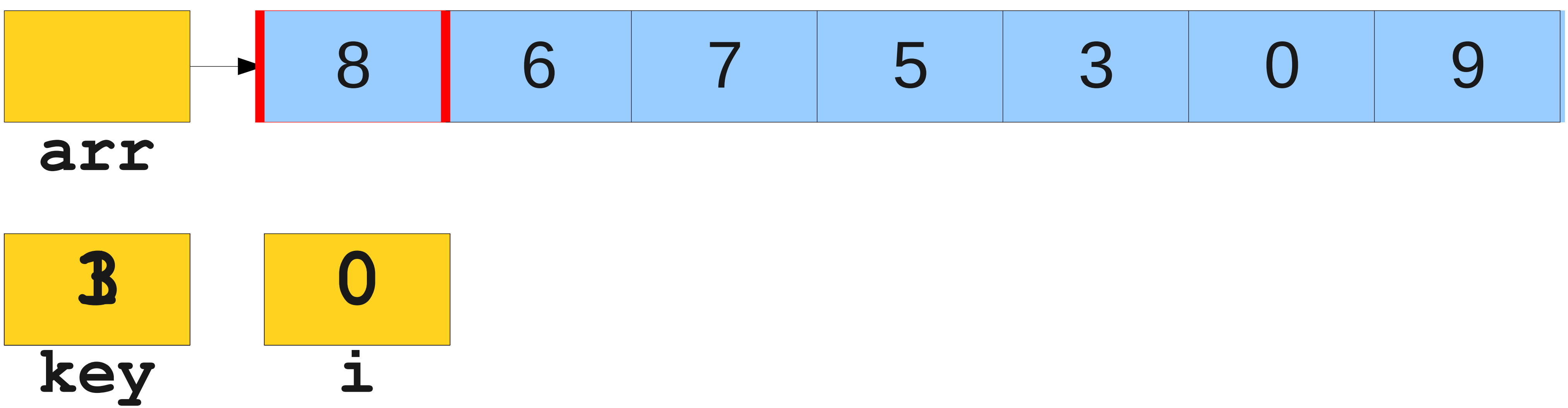
Linear Search

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

    return -1;
}
```



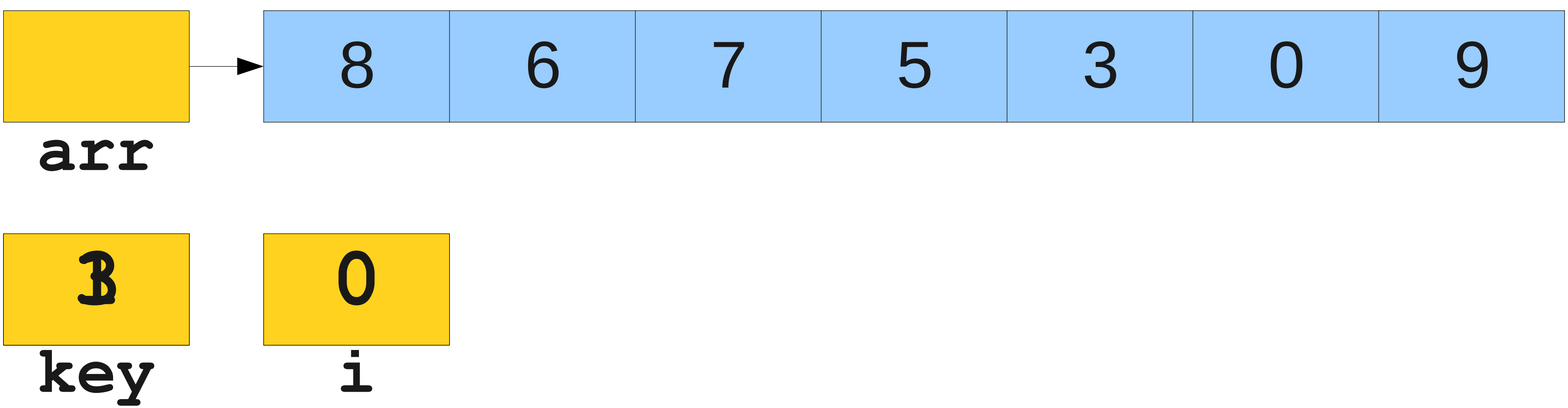
Linear Search

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

    return -1;
}
```



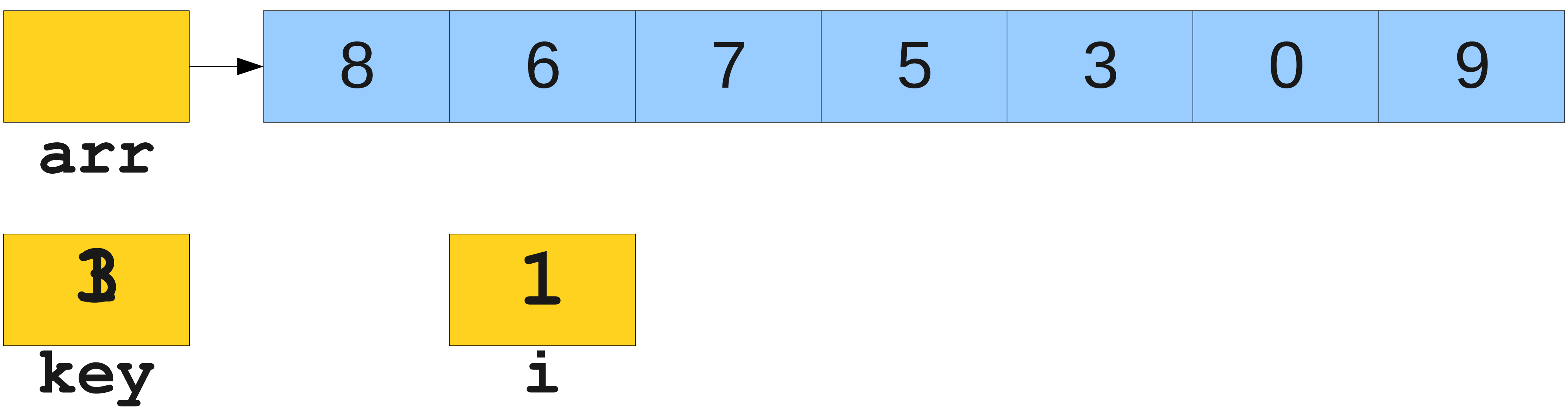
Linear Search

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

    return -1;
}
```



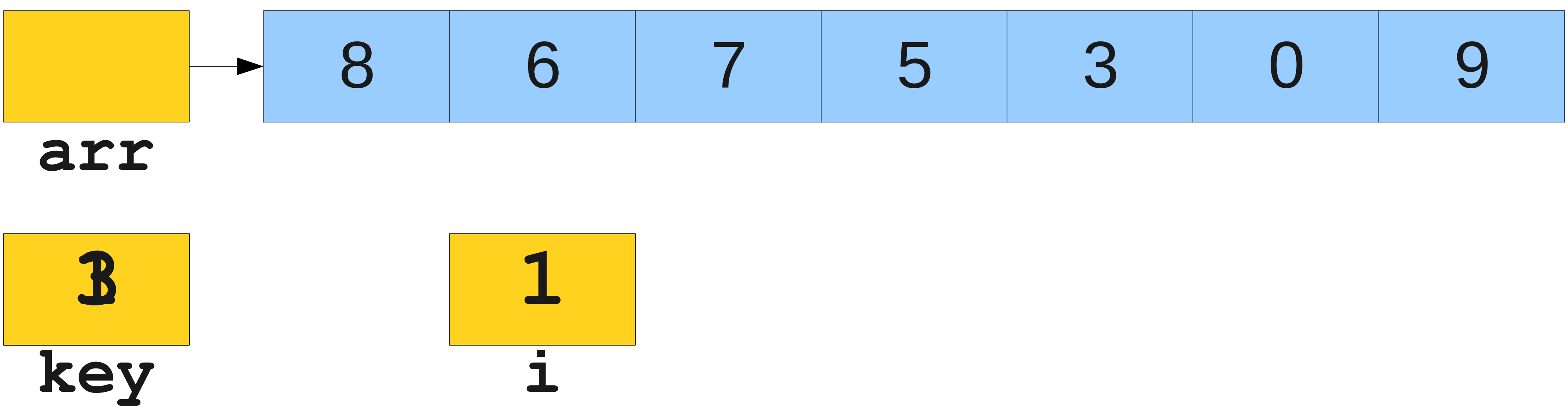
Linear Search

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

    return -1;
}
```



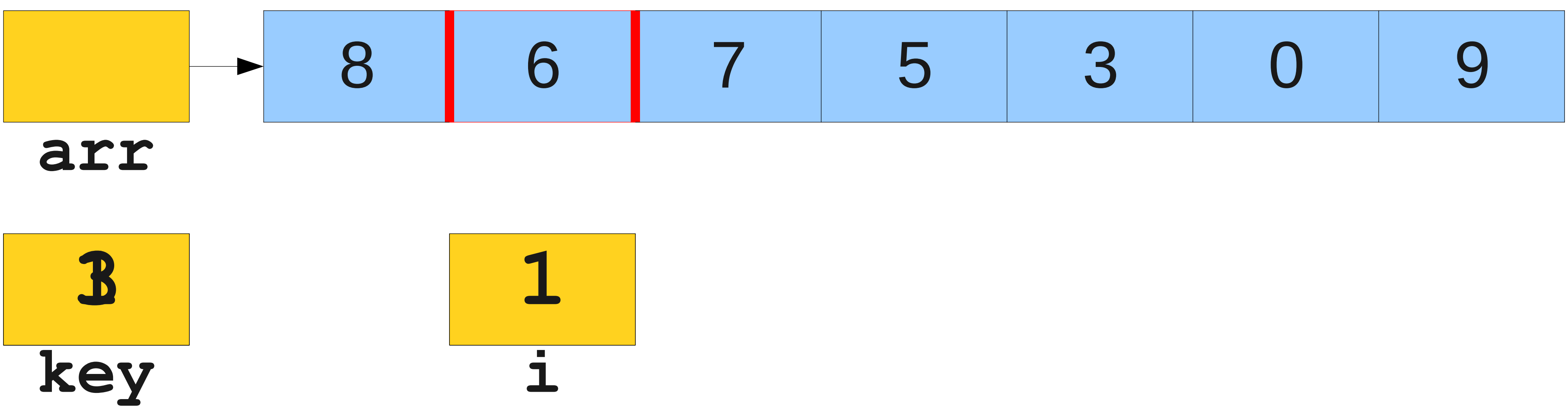
Linear Search

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

    return -1;
}
```



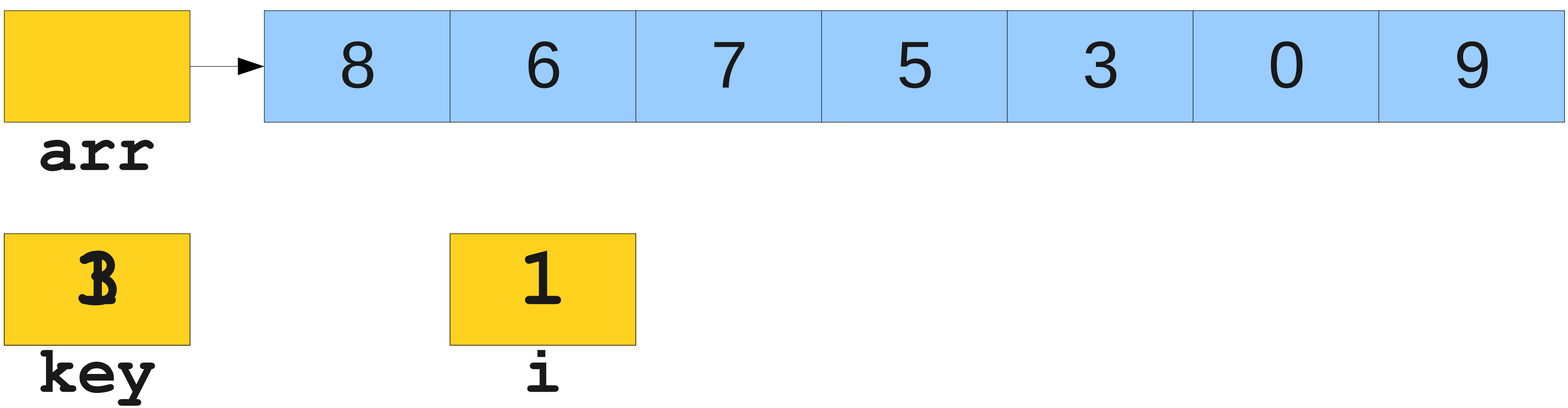
Linear Search

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

    return -1;
}
```



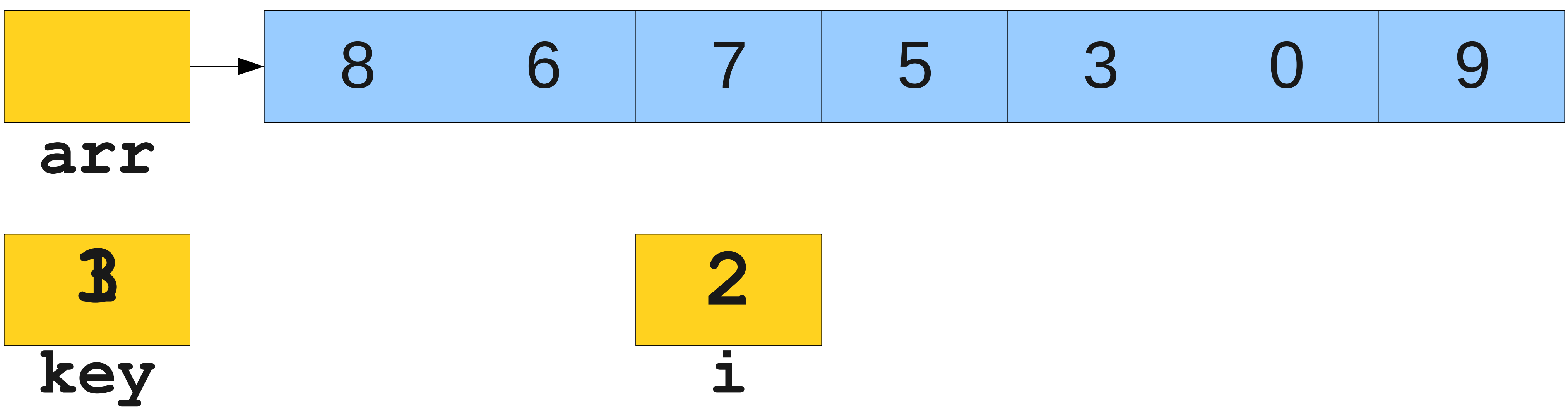
Linear Search

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

    return -1;
}
```



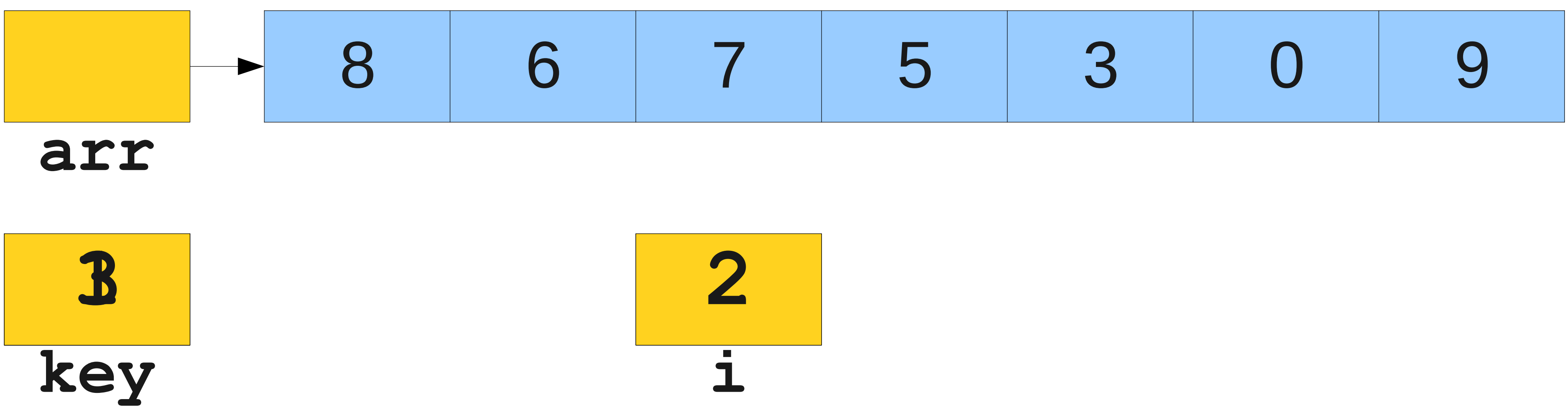
Linear Search

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

    return -1;
}
```



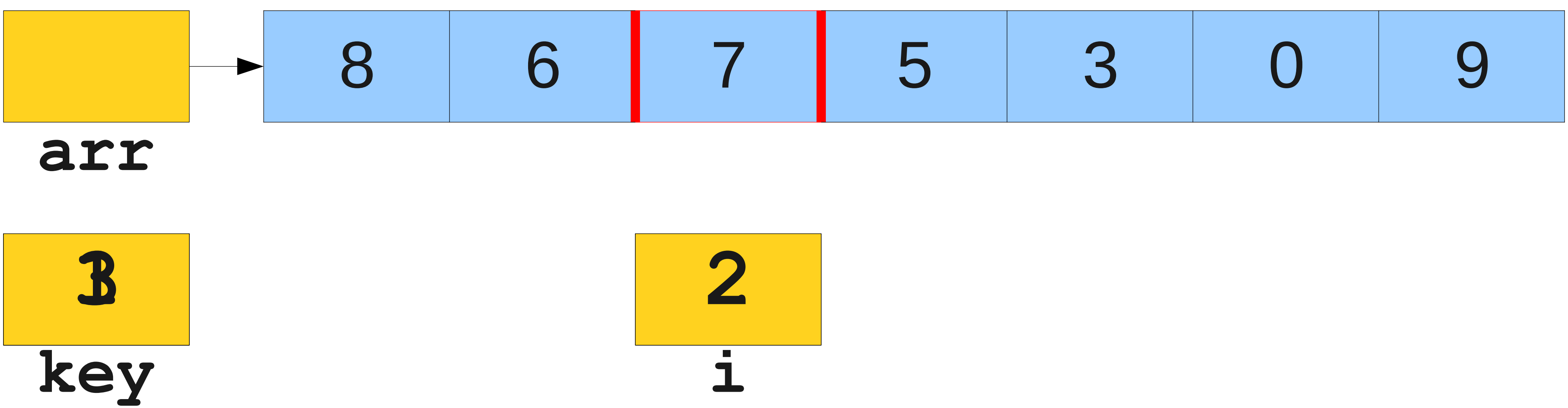
Linear Search

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

    return -1;
}
```



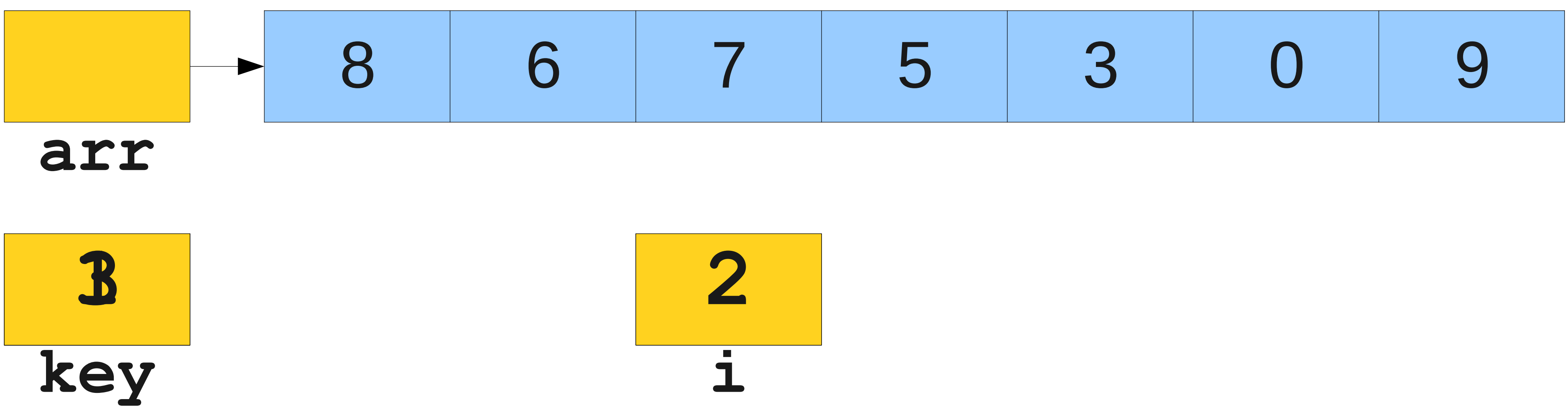
Linear Search

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

    return -1;
}
```



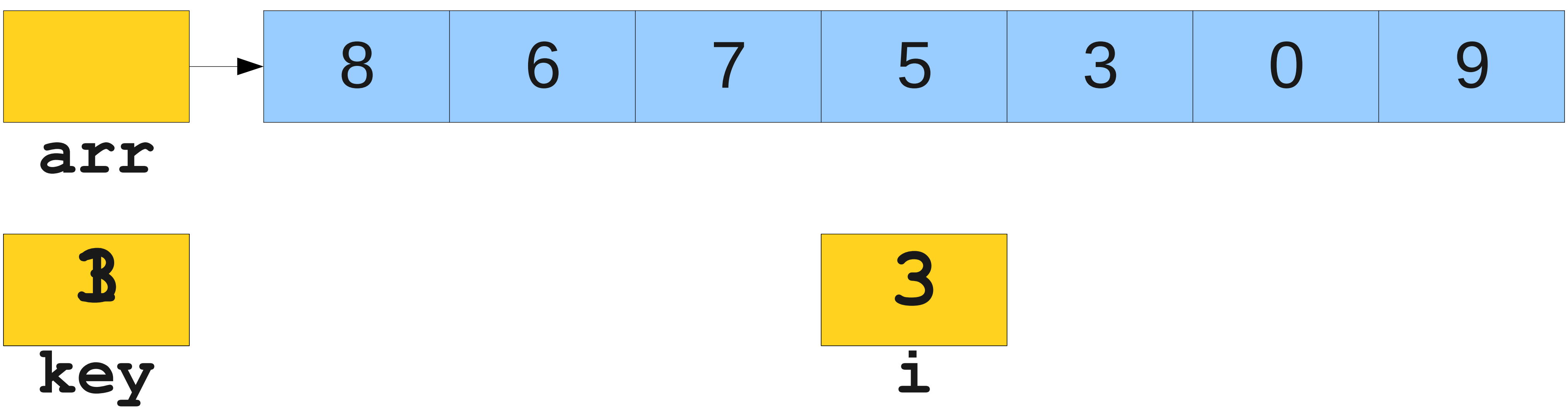
Linear Search

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

    return -1;
}
```



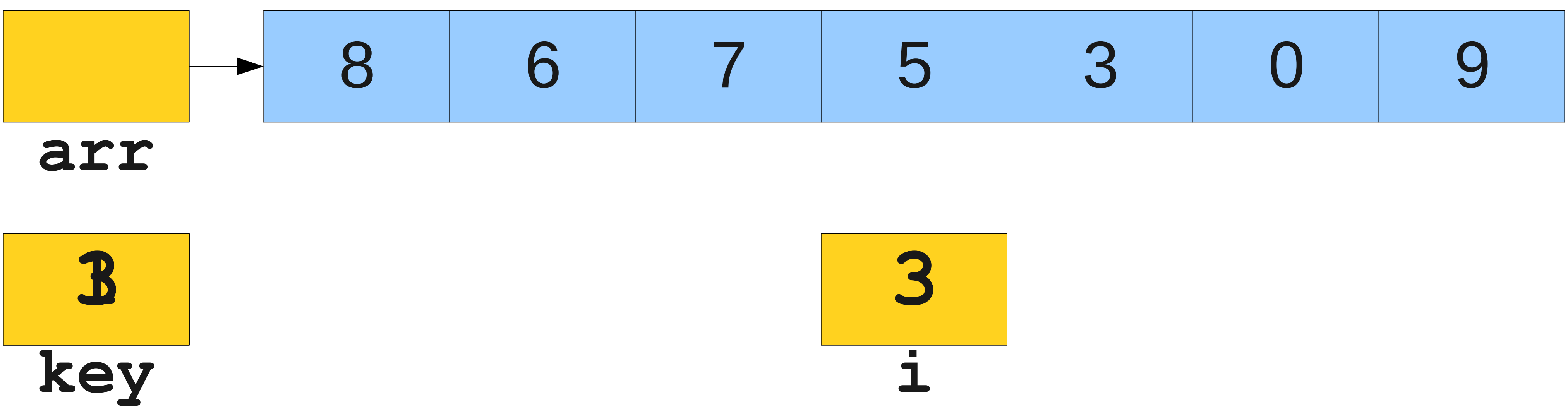
Linear Search

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

    return -1;
}
```



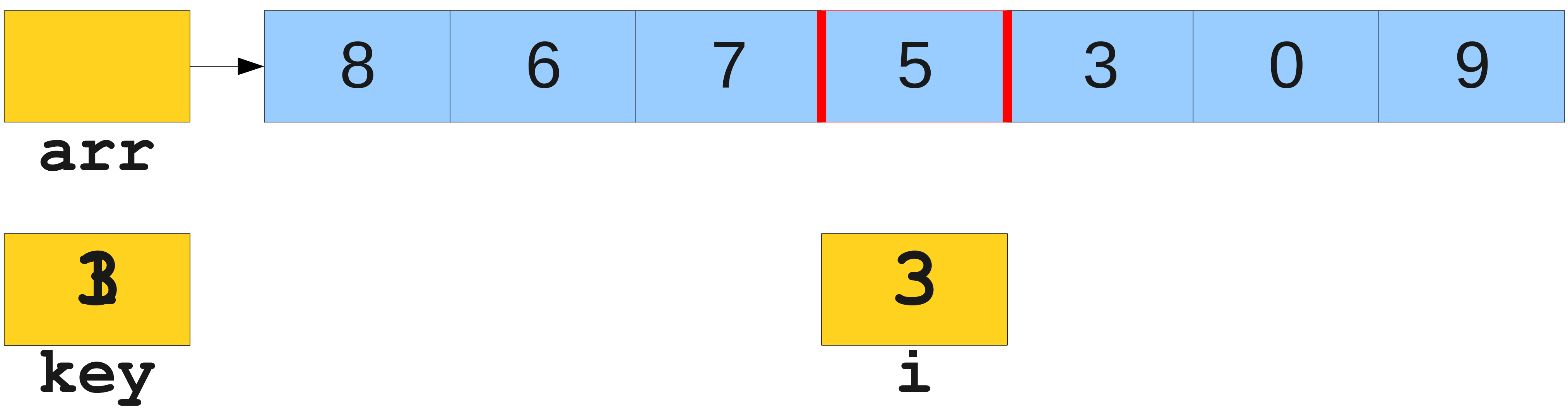
Linear Search

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

    return -1;
}
```



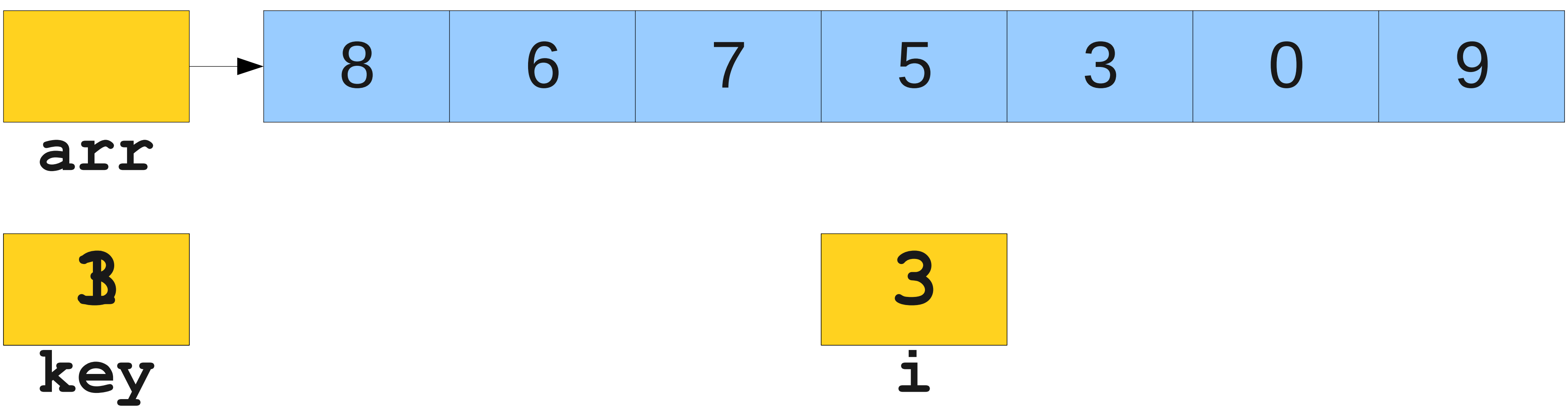
Linear Search

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

    return -1;
}
```



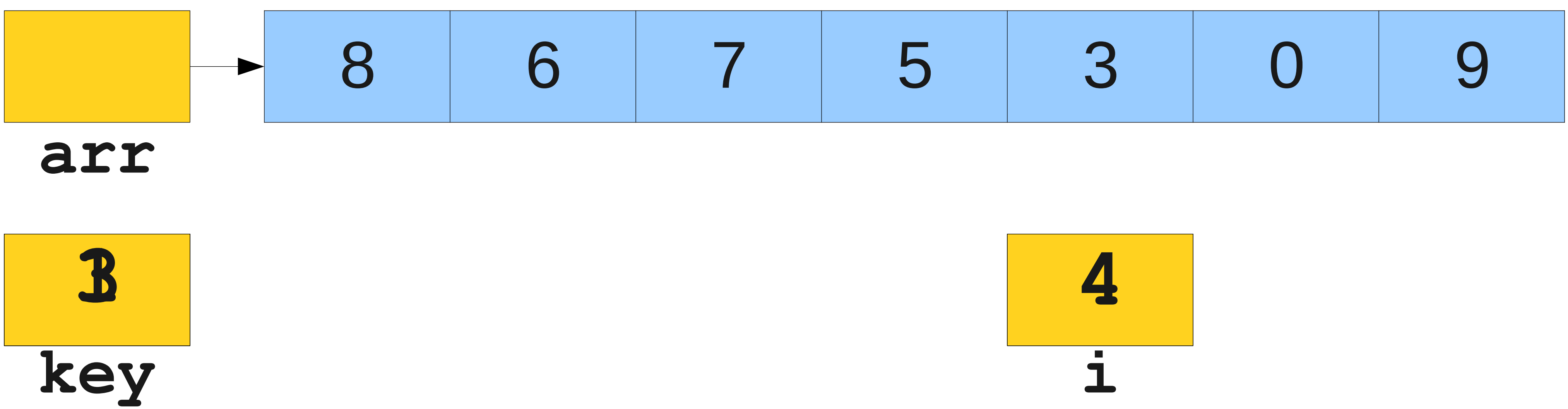
Linear Search

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

    return -1;
}
```



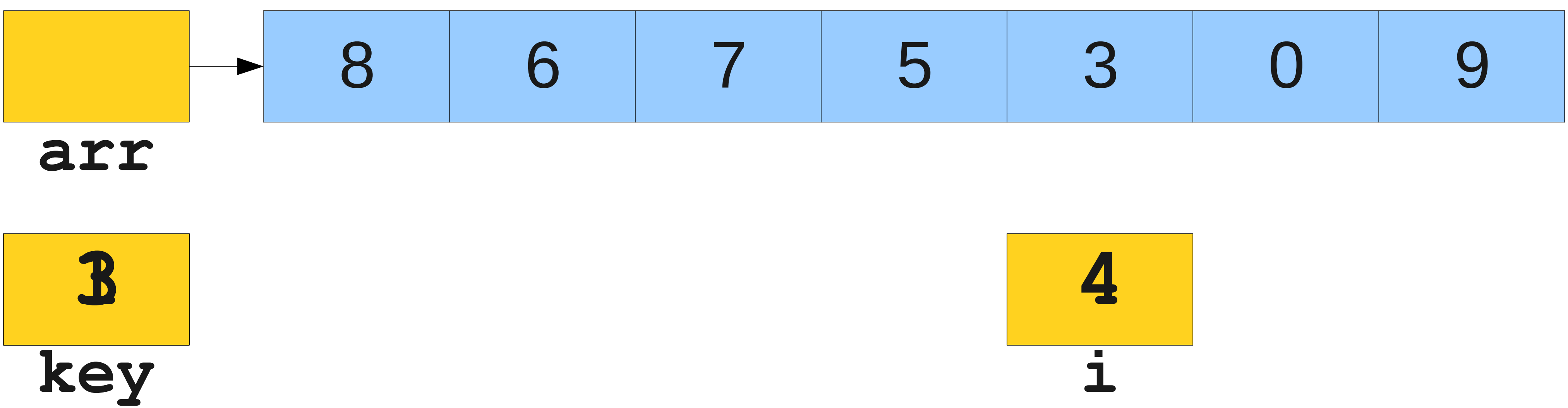
Linear Search

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

    return -1;
}
```



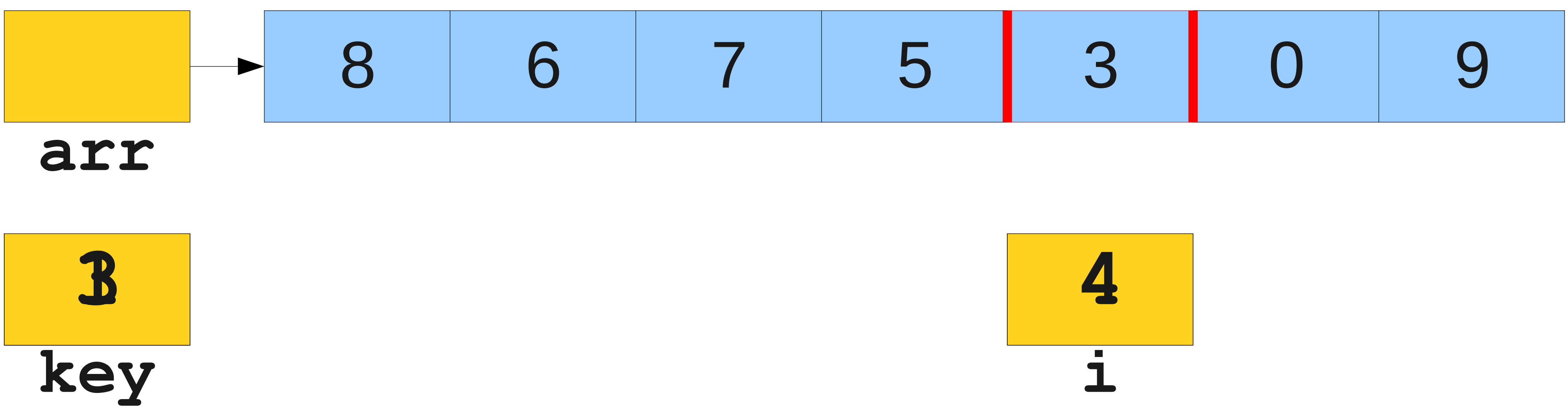
Linear Search

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

    return -1;
}
```



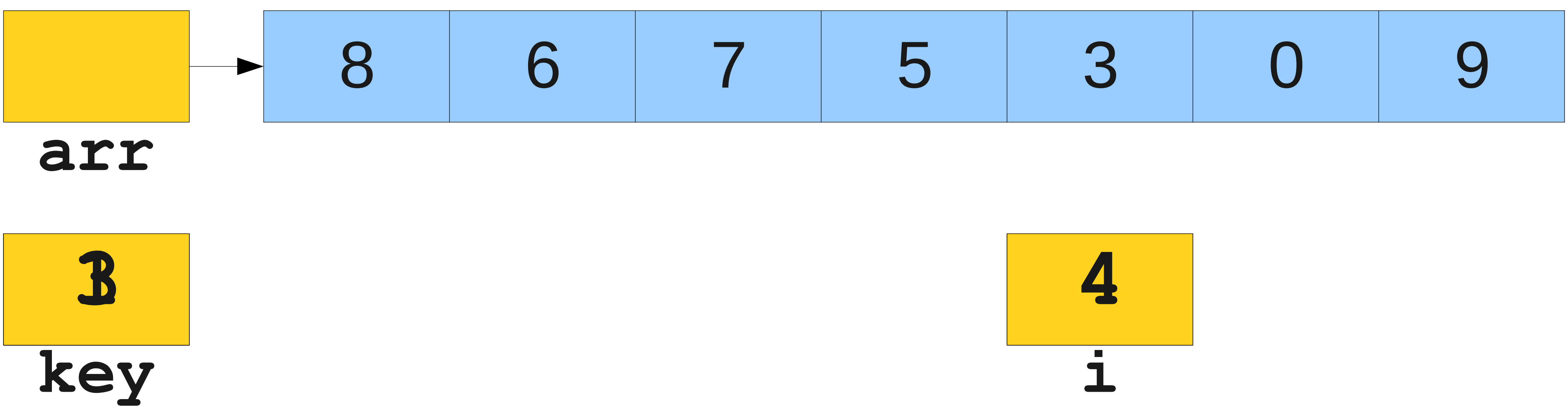
Linear Search

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

    return -1;
}
```



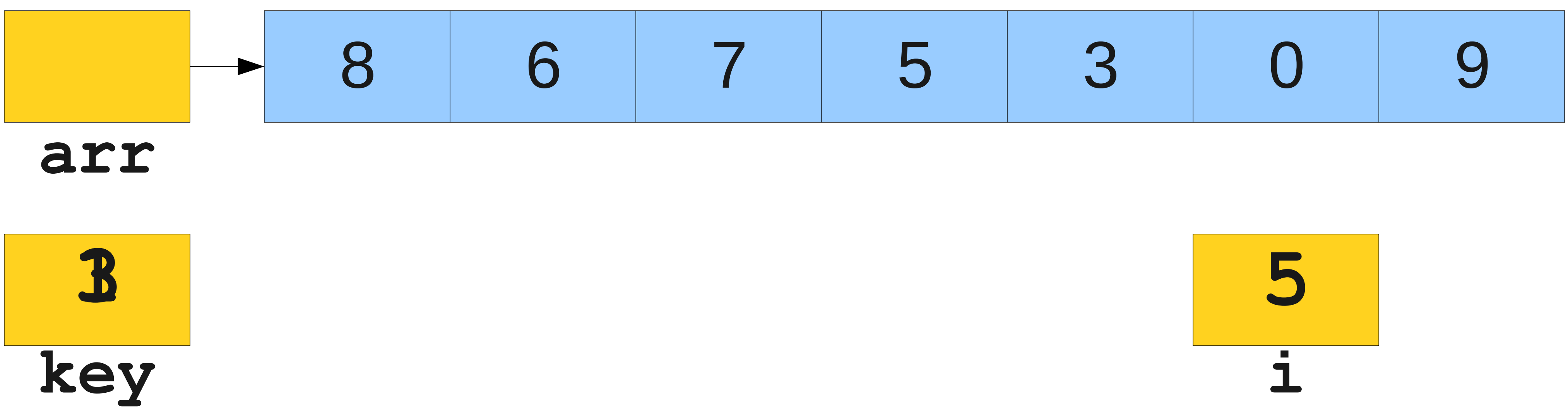
Linear Search

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

    return -1;
}
```



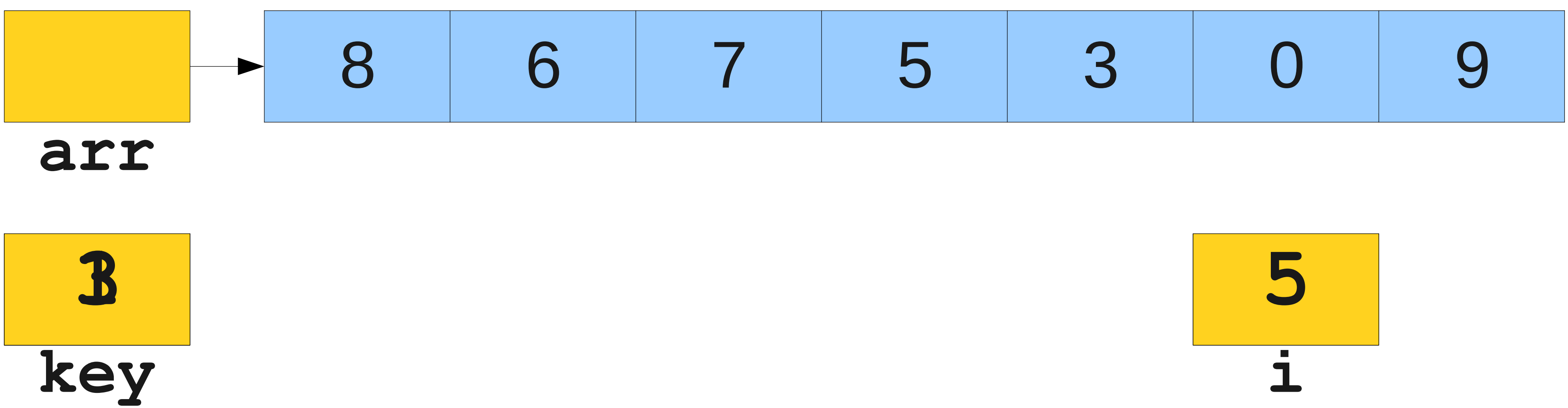
Linear Search

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

    return -1;
}
```



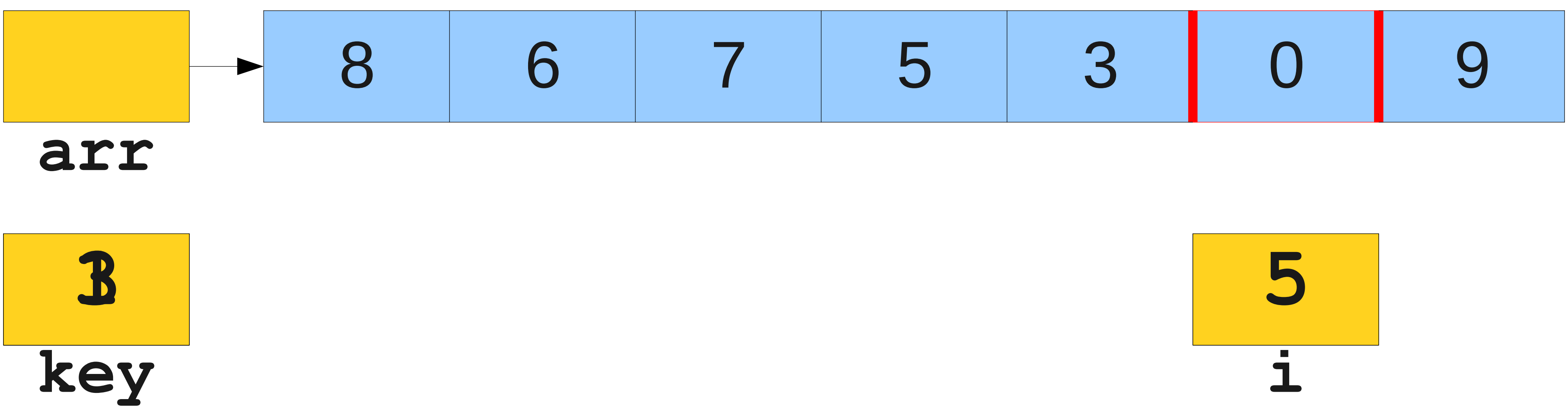
Linear Search

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

    return -1;
}
```



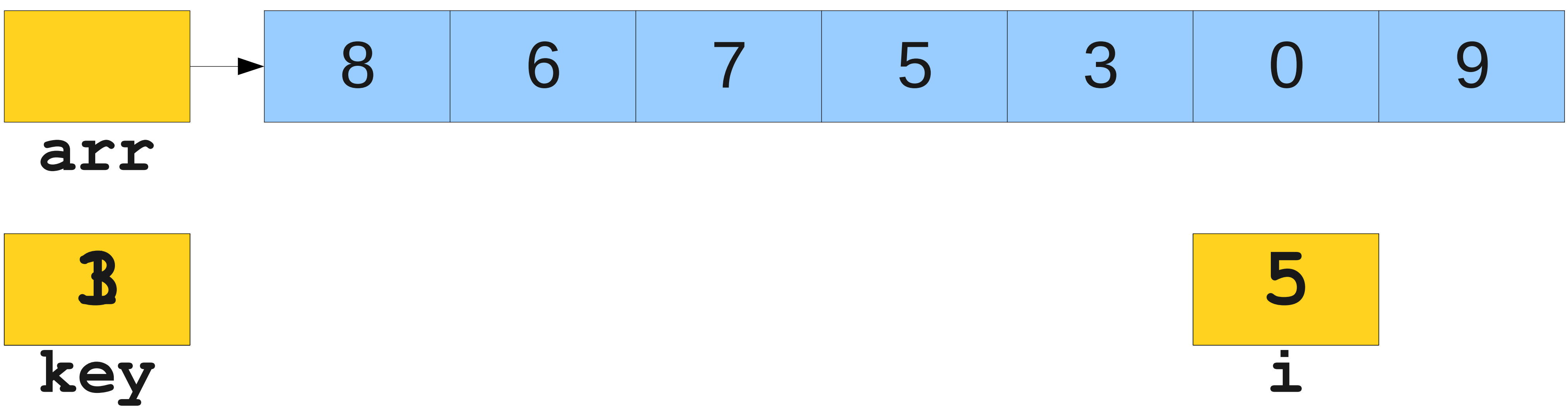
Linear Search

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

    return -1;
}
```



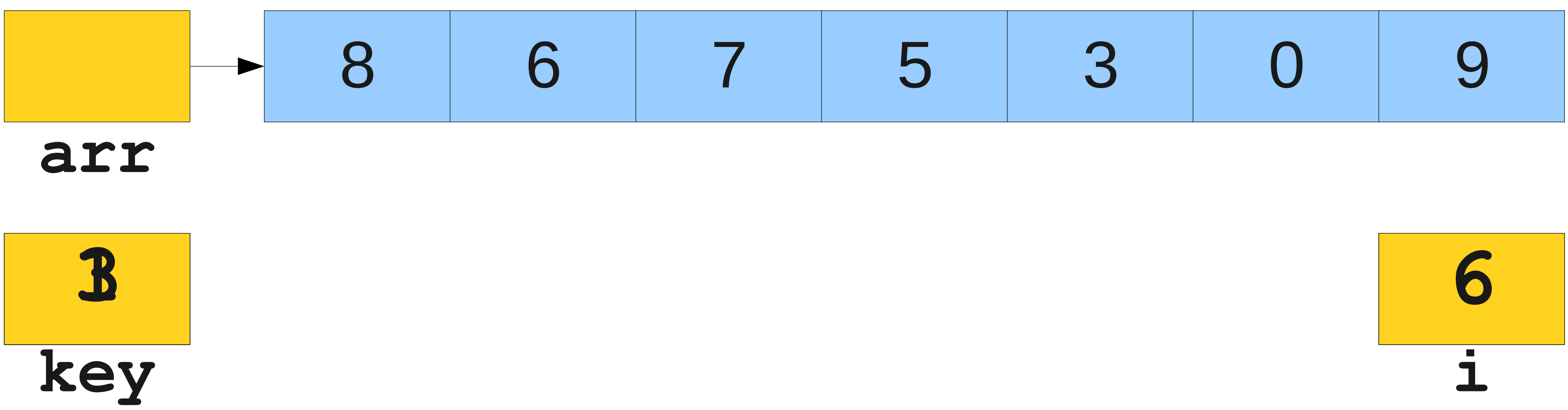
Linear Search

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

    return -1;
}
```



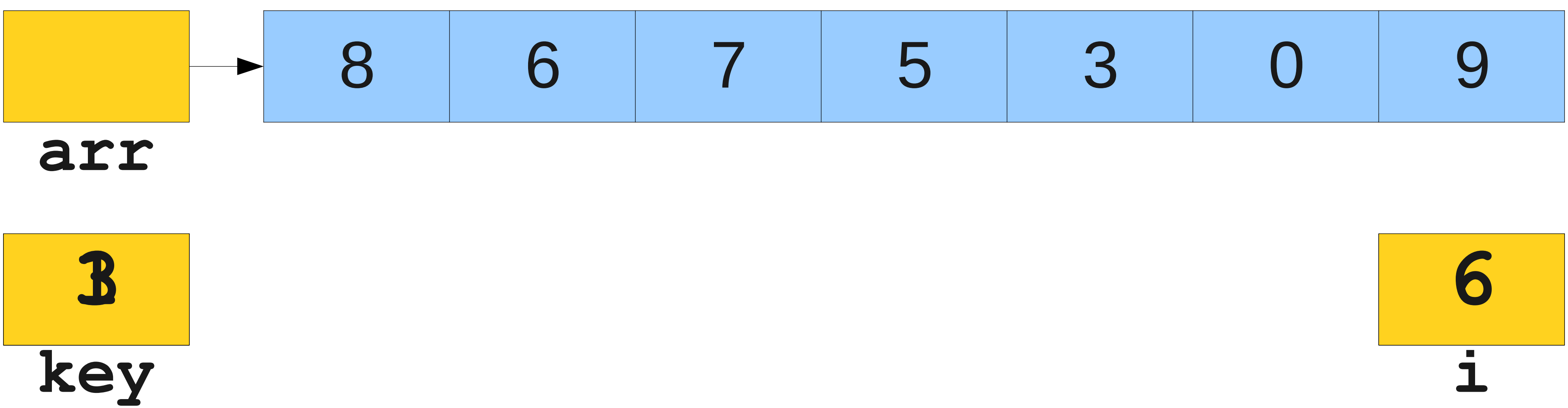
Linear Search

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

    return -1;
}
```



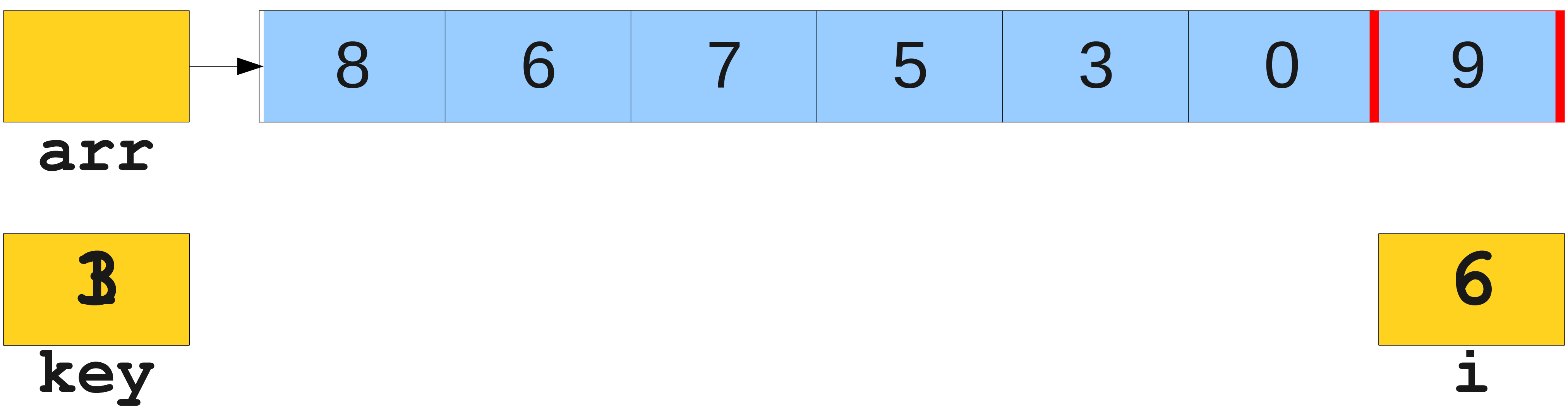
Linear Search

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

    return -1;
}
```



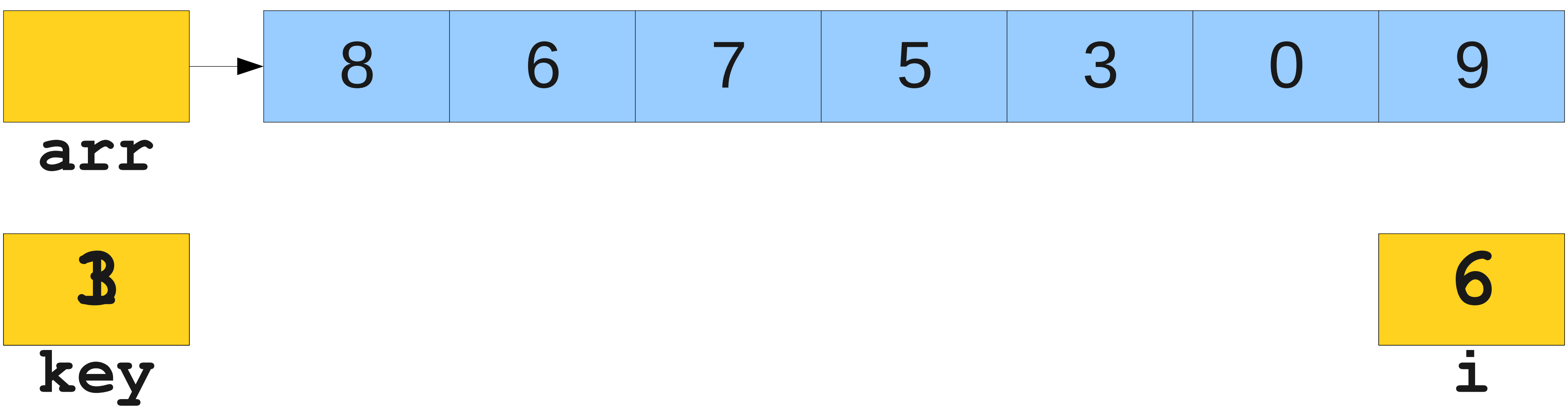
Linear Search

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

    return -1;
}
```



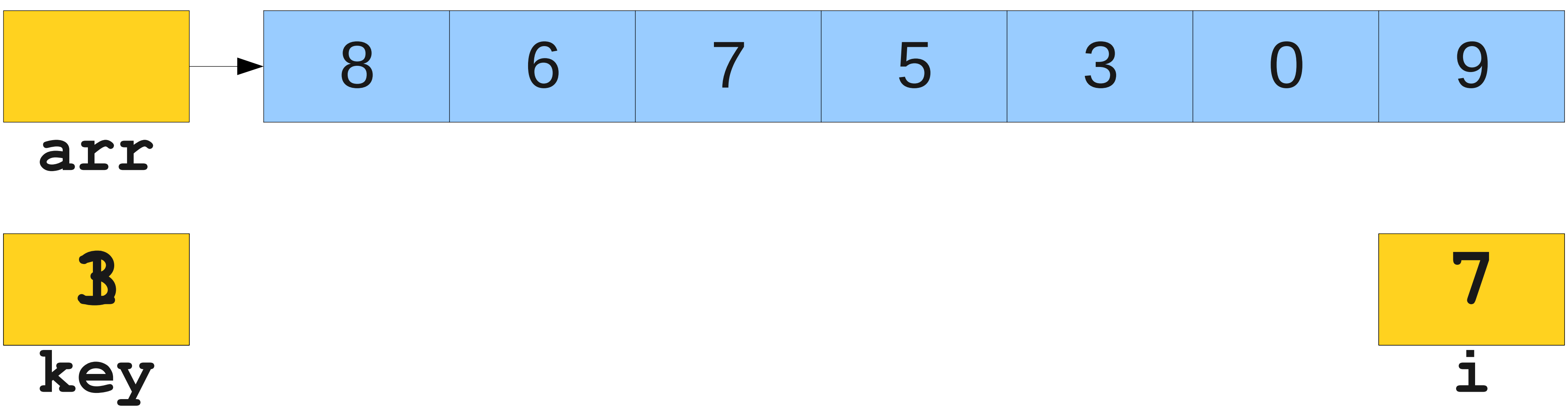
Linear Search

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

    return -1;
}
```



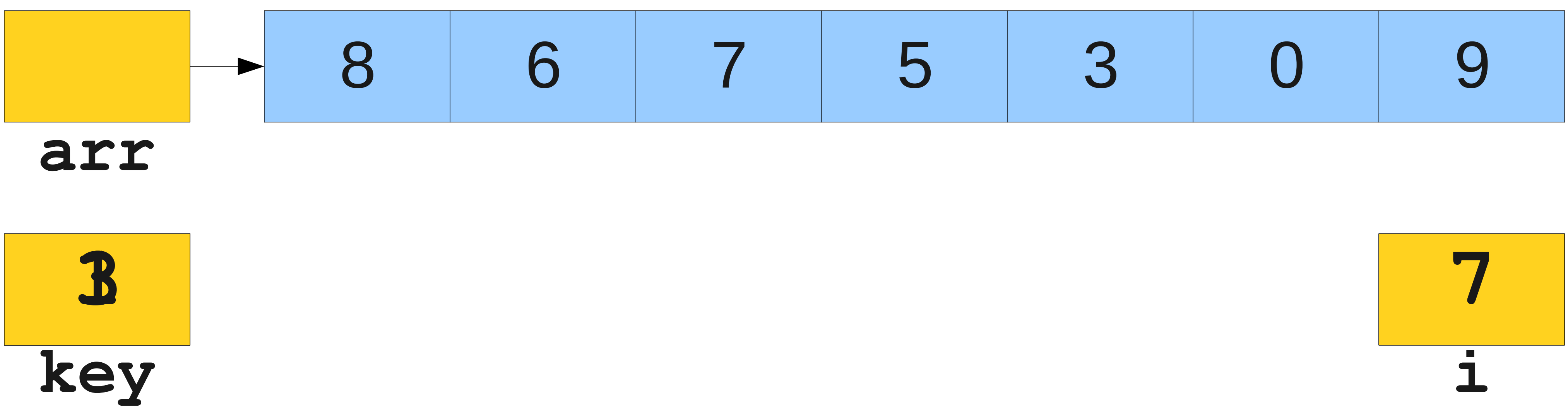
Linear Search

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

    return -1;
}
```



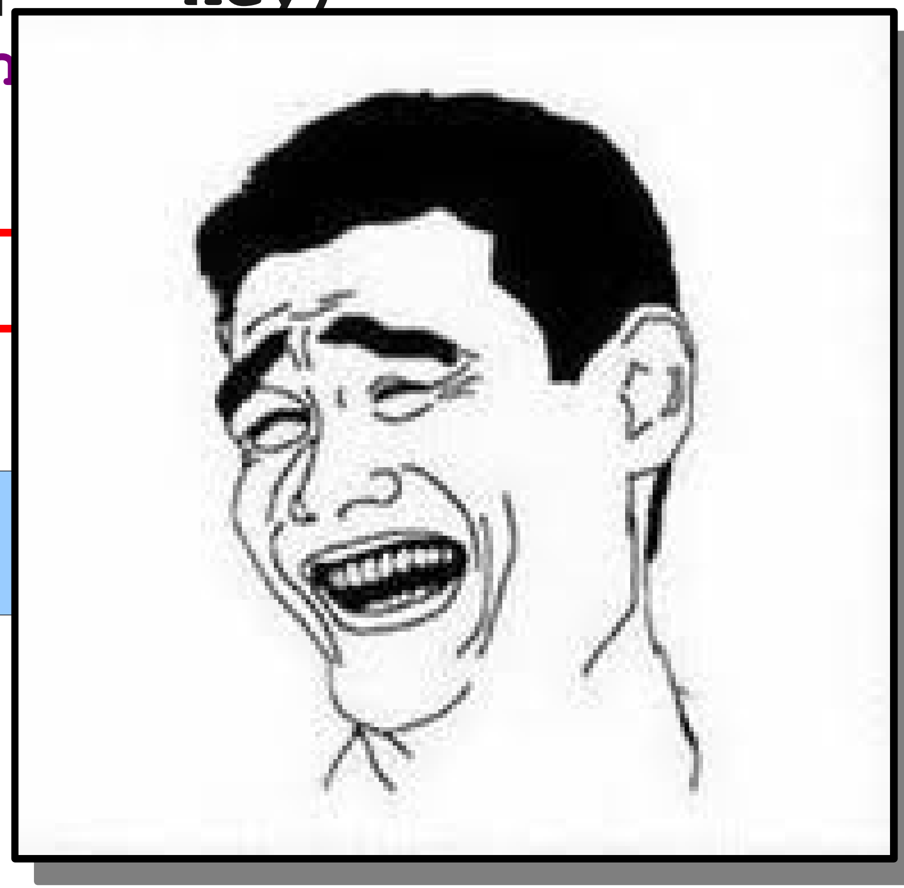
Linear Search

```
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
    }

    return -1;
}
```



8	6	9

arr

key: 3

i: 7

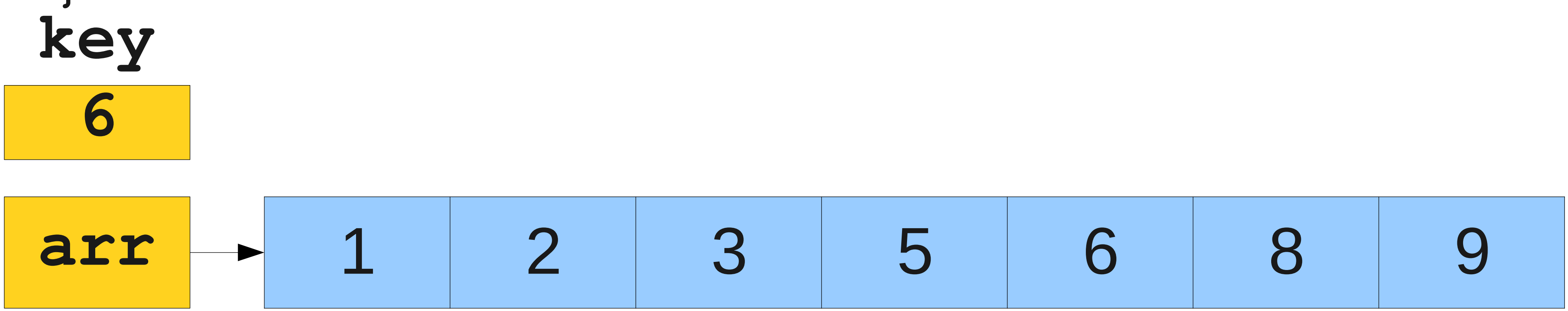
Searching II

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;
}
```

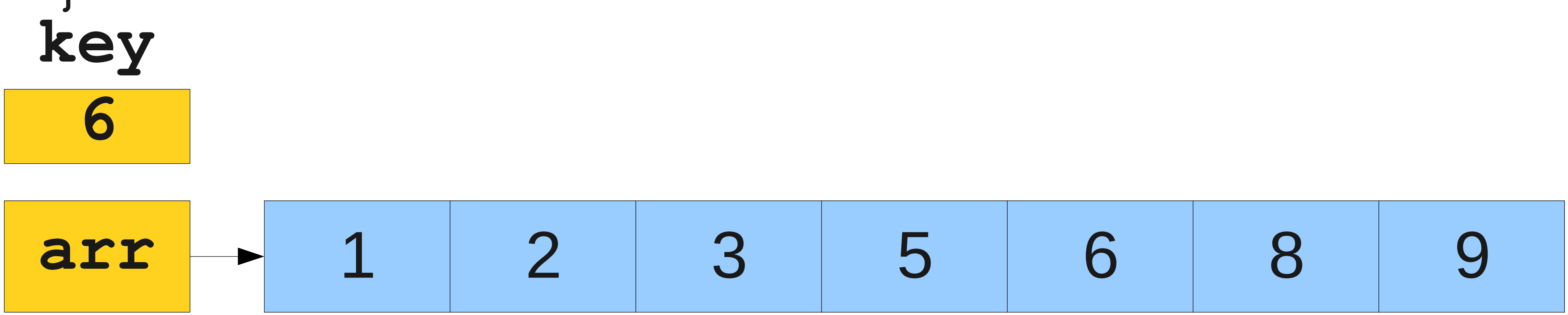
Binary Search

```
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;
}
```



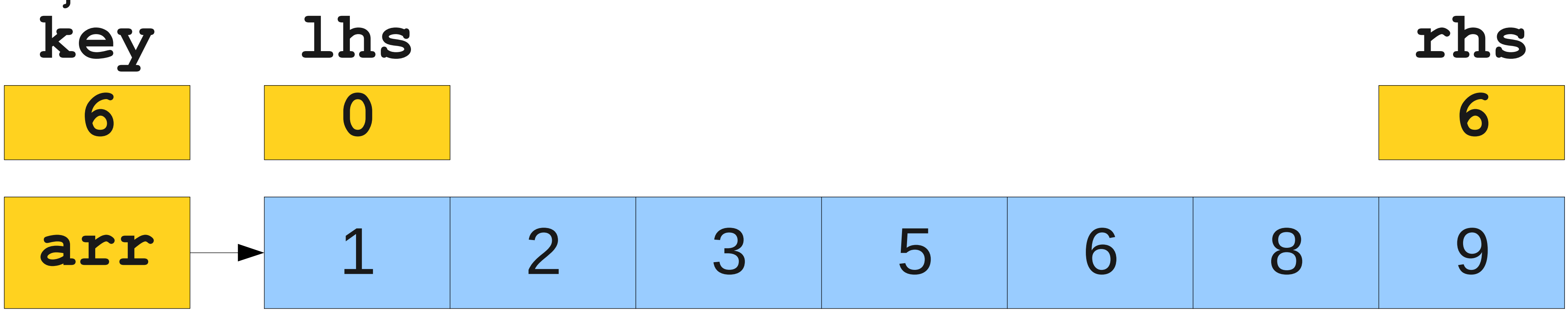
Binary Search

```
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;
}
```



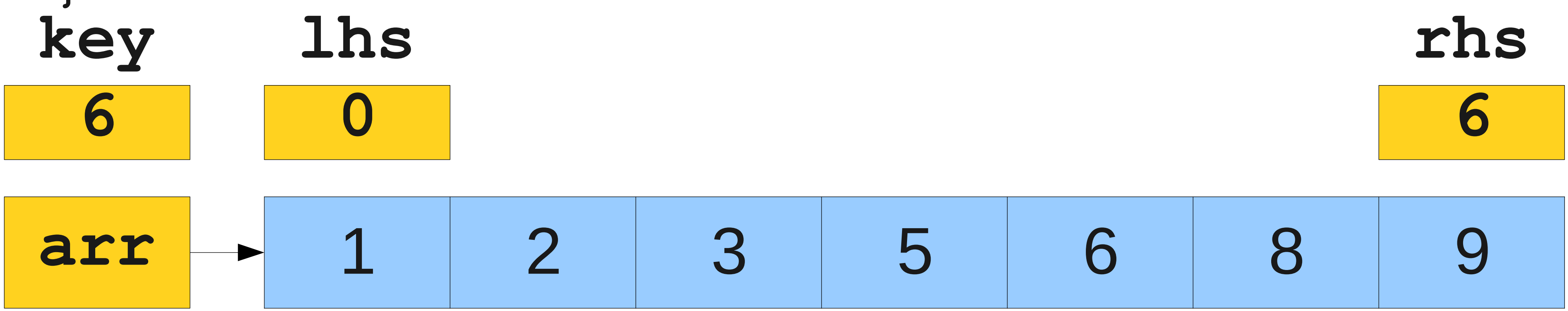
Binary Search

```
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;
}
```



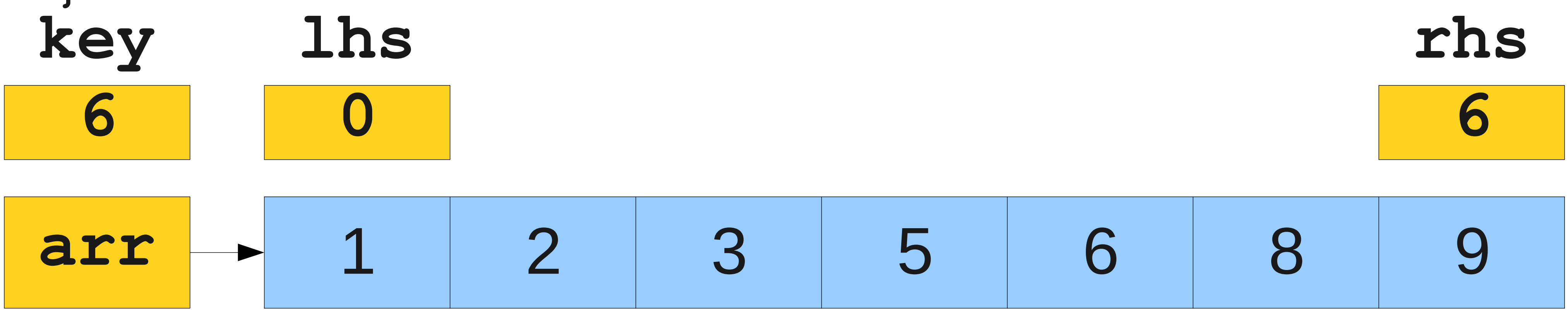
Binary Search

```
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;
}
```



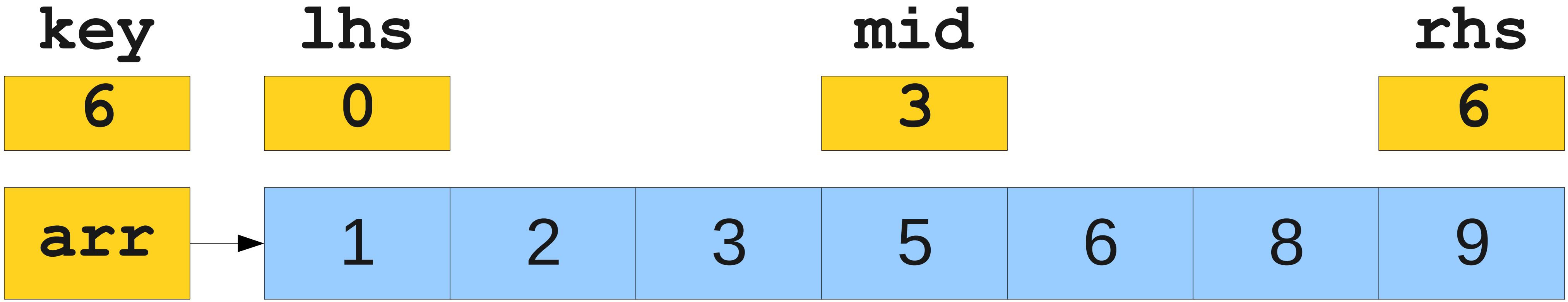
Binary Search

```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;
}
```



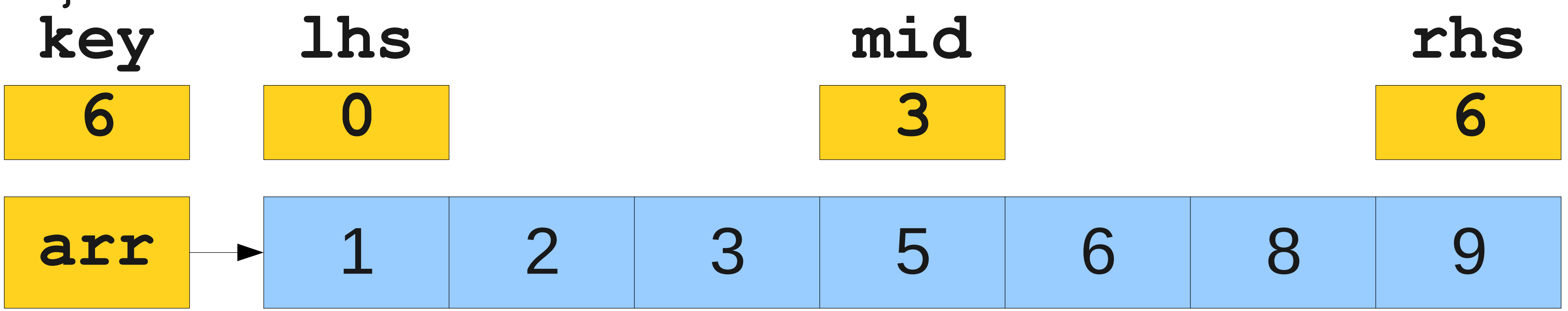
Binary Search

```
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;
}
```



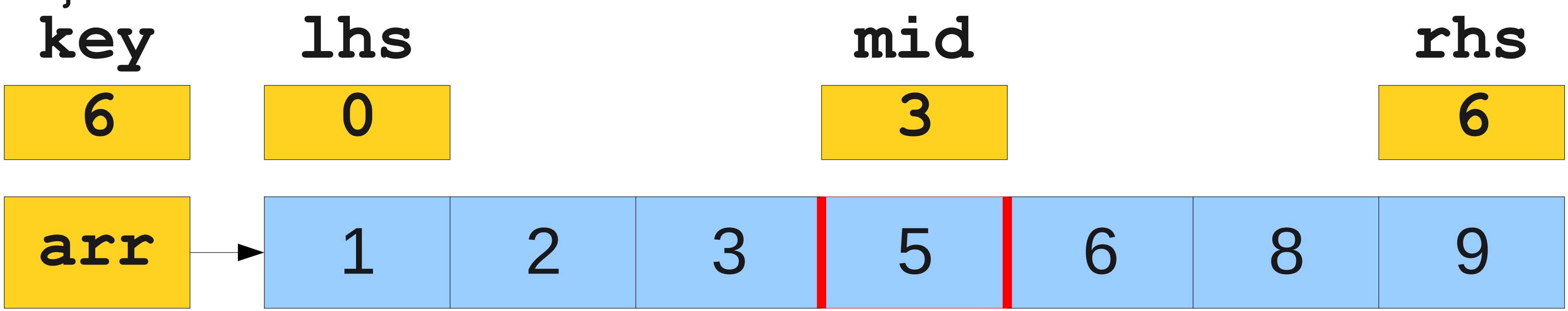
Binary Search

```
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;
}
```



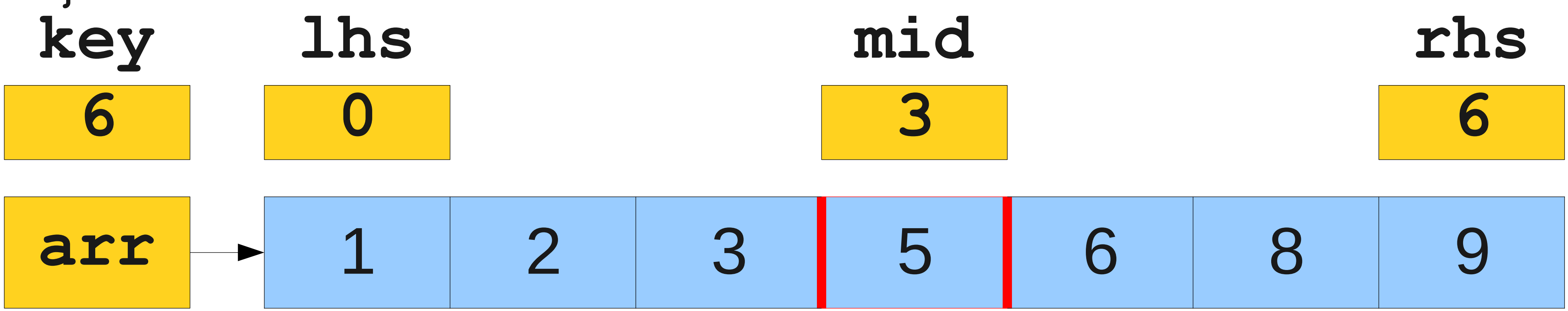
Binary Search

```
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;
}
```



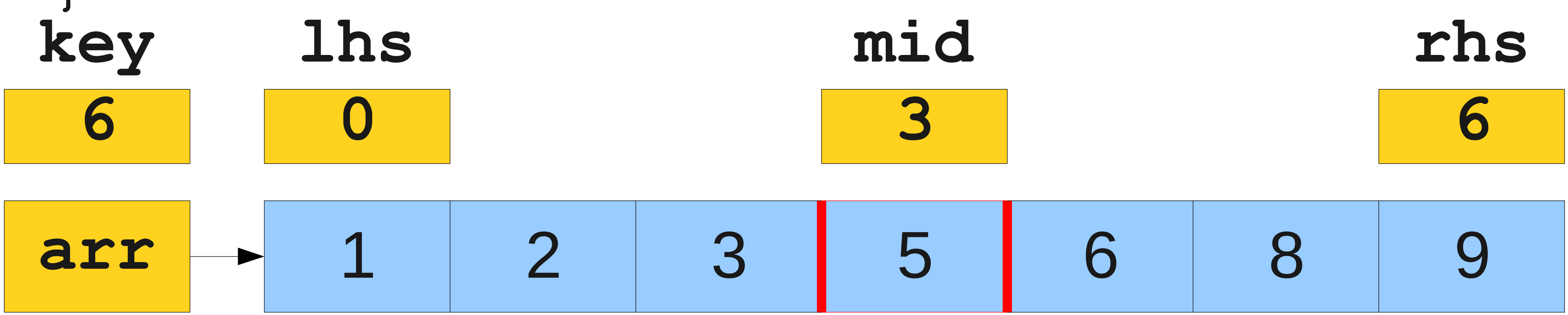
Binary Search

```
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;
}
```



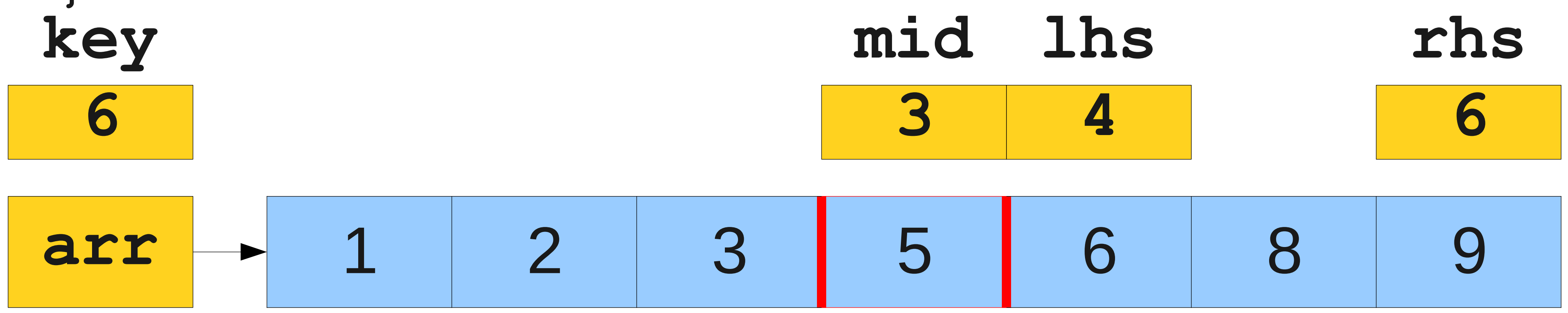
Binary Search

```
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;
}
```



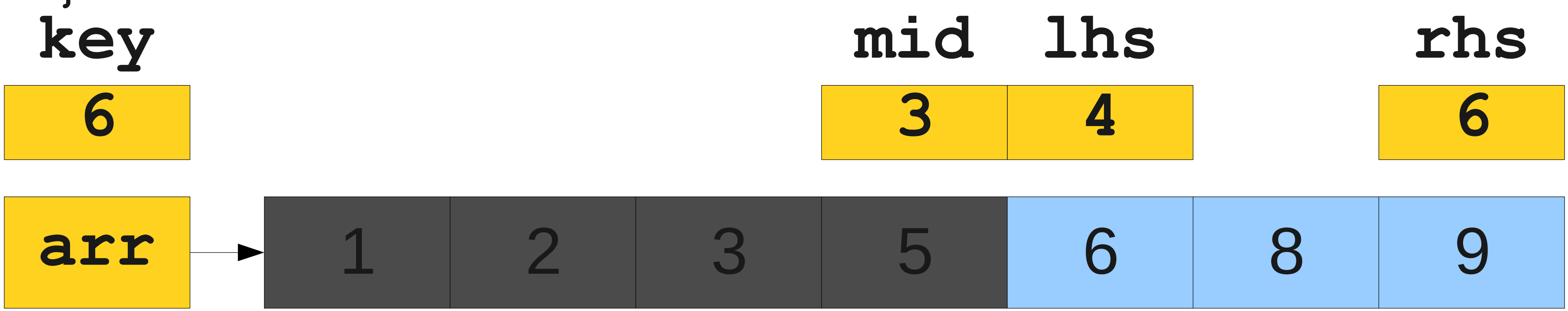
Binary Search

```
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;
}
```



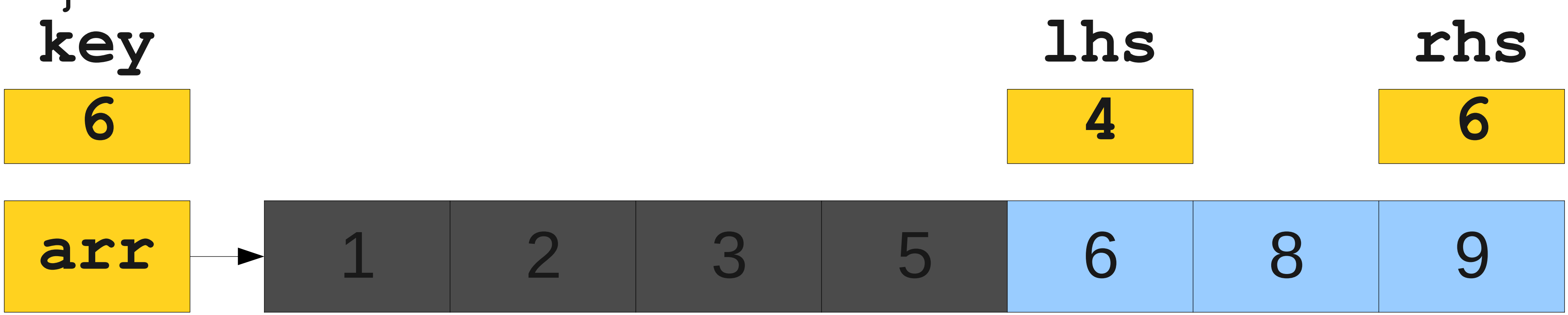
Binary Search

```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;
}
```



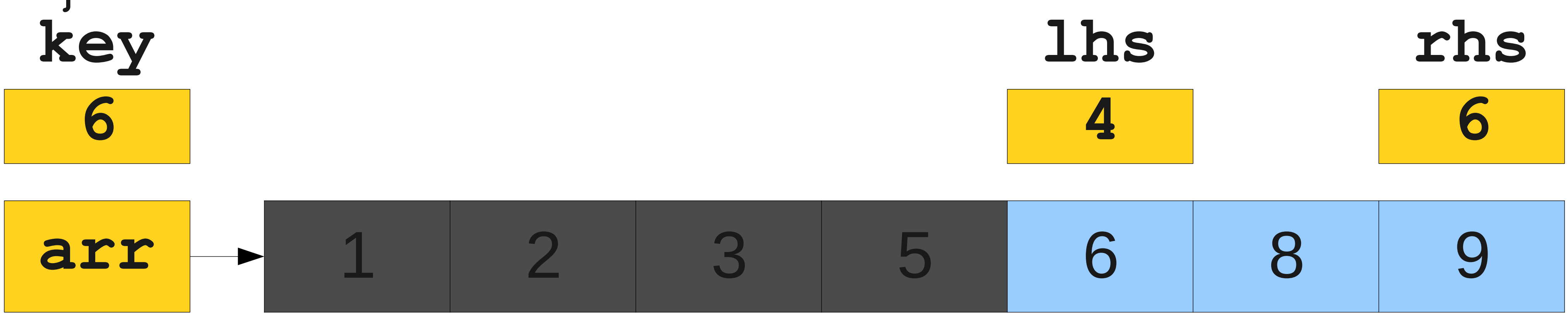
Binary Search

```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;
}
```



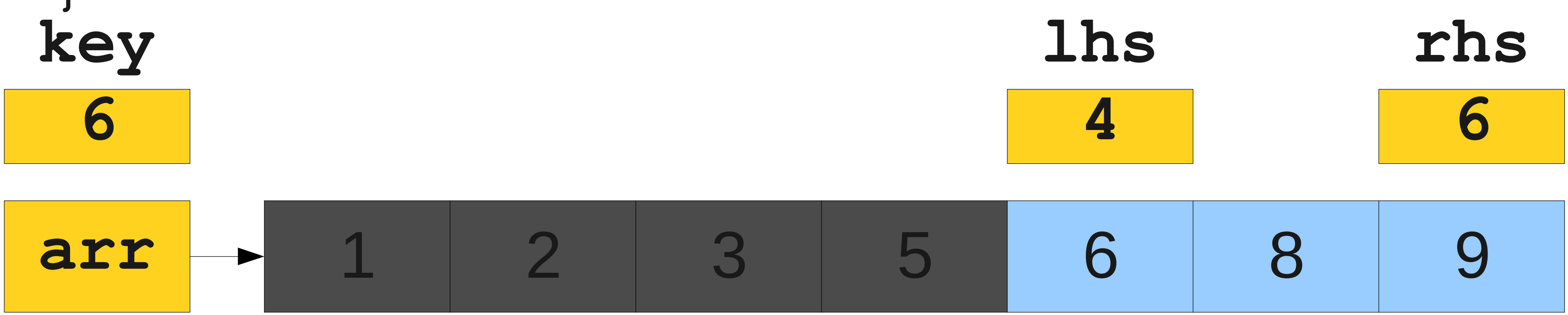
Binary Search

```
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;
}
```



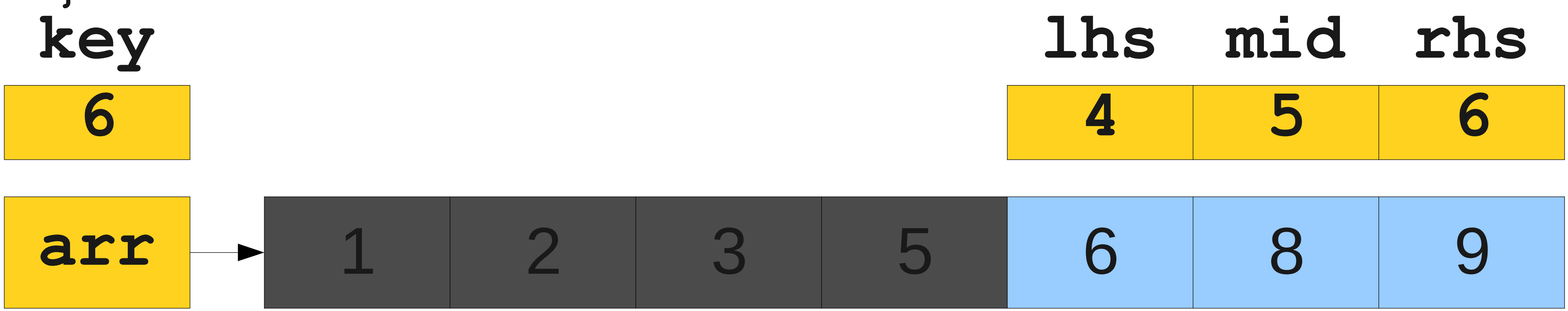
Binary Search

```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;
}
```



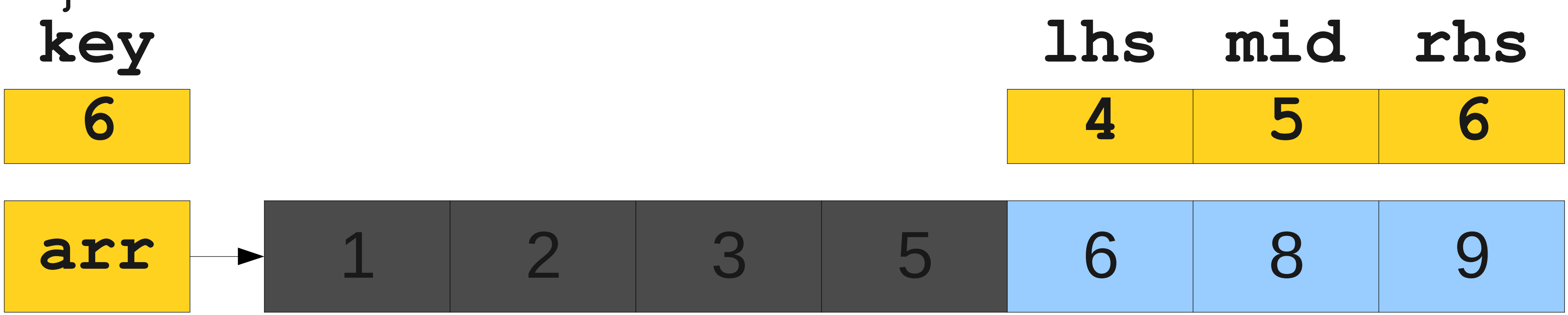
Binary Search

```
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;
}
```



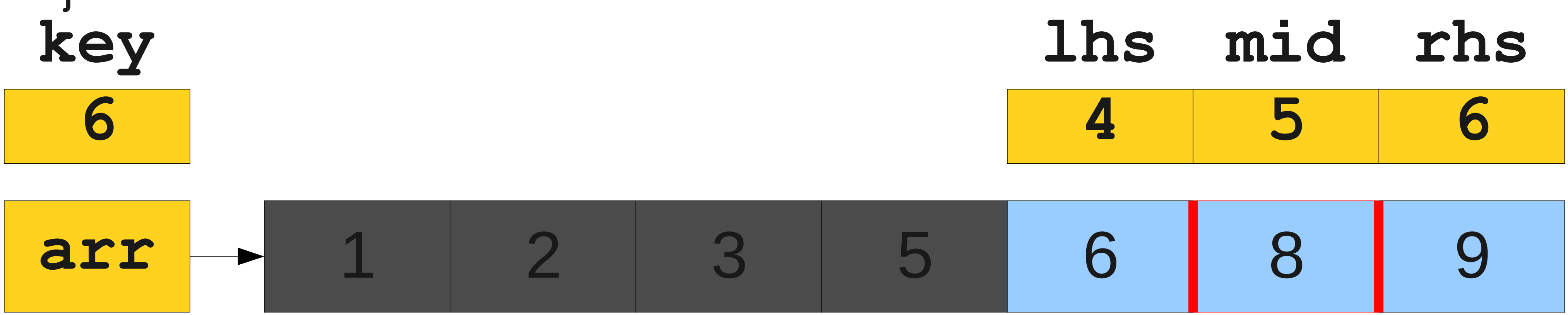
Binary Search

```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;
}
```



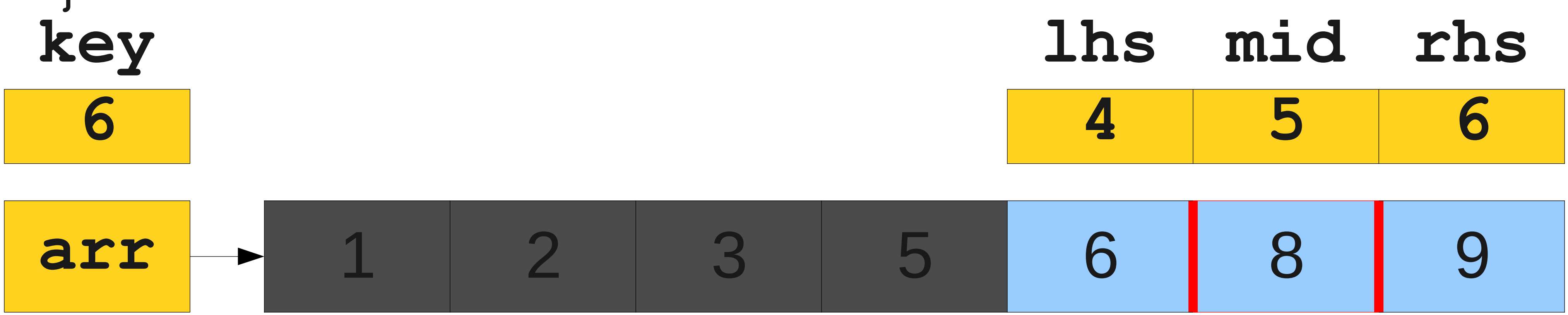
Binary Search

```
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;
}
```



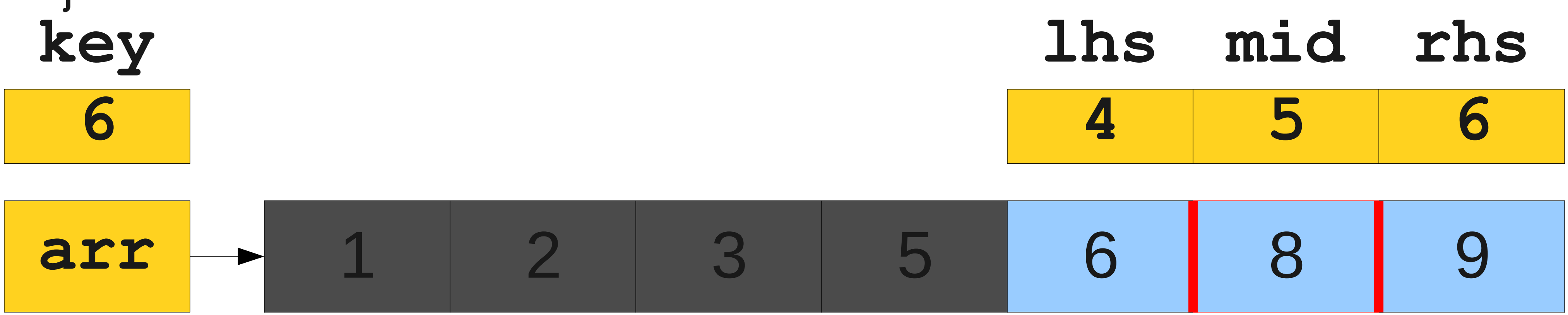
Binary Search

```
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;
}
```



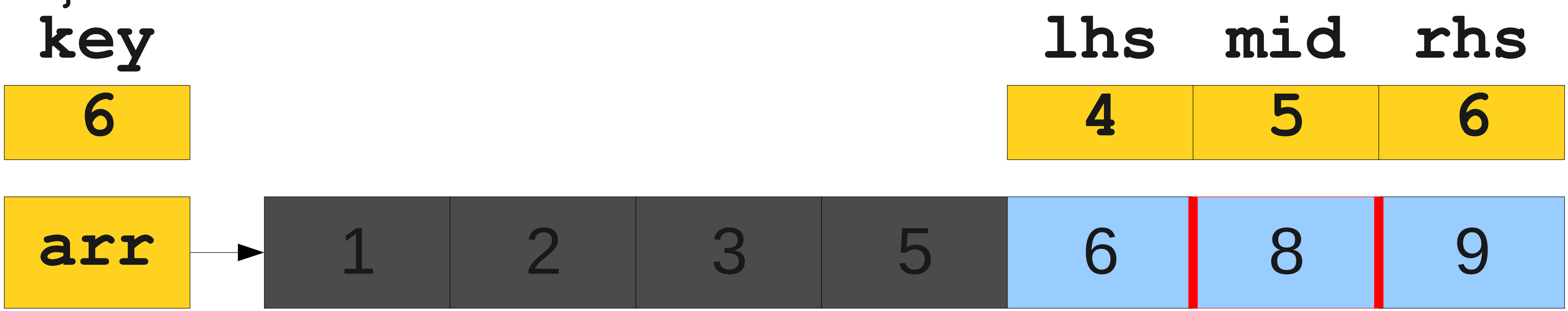
Binary Search

```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;
}
```



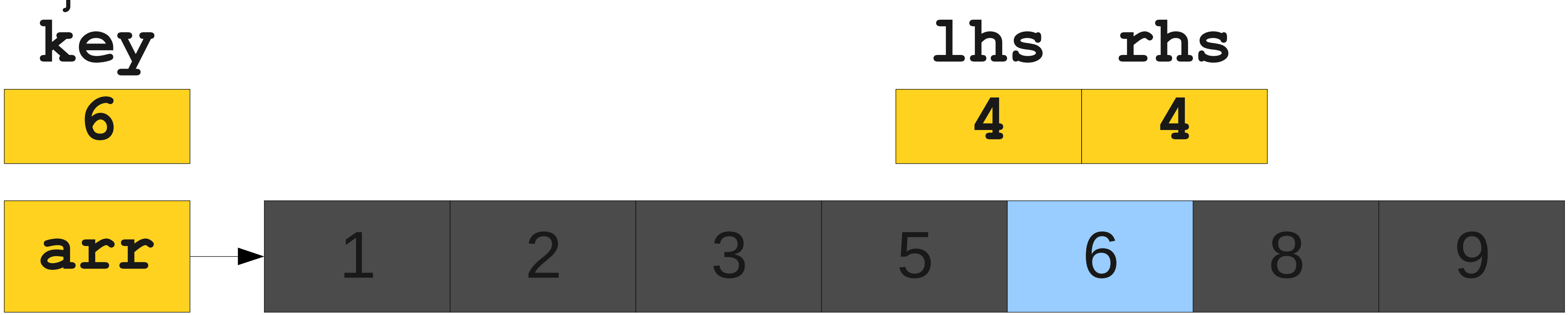
Binary Search

```
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;
}
```



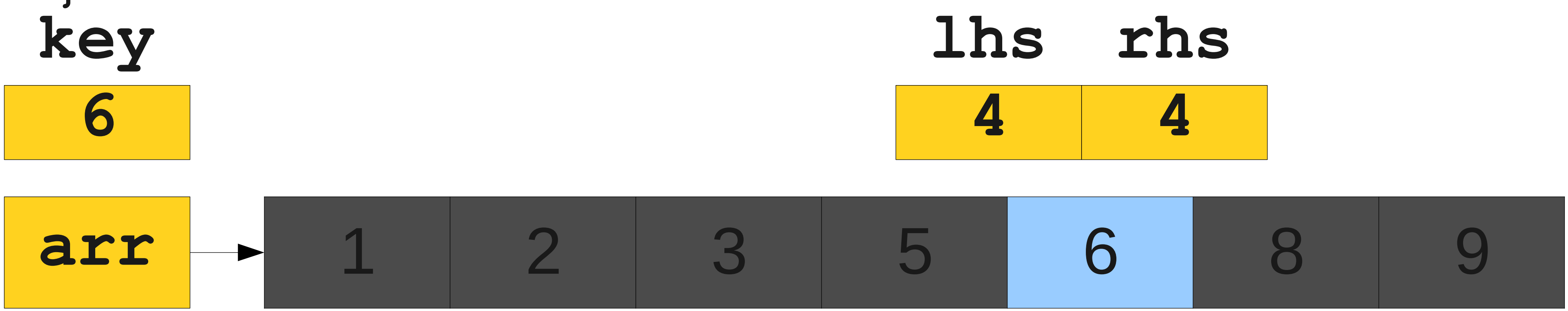
Binary Search

```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;
}
```



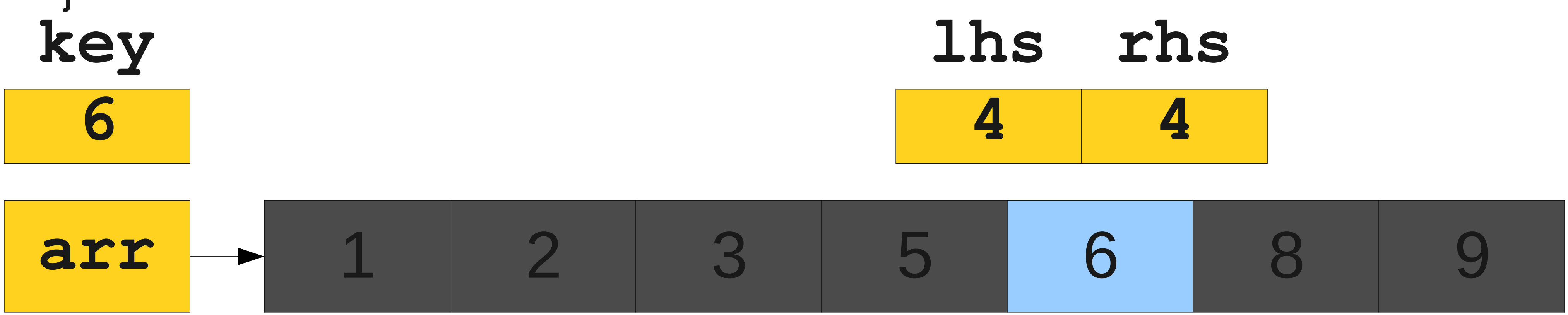
Binary Search

```
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;
}
```



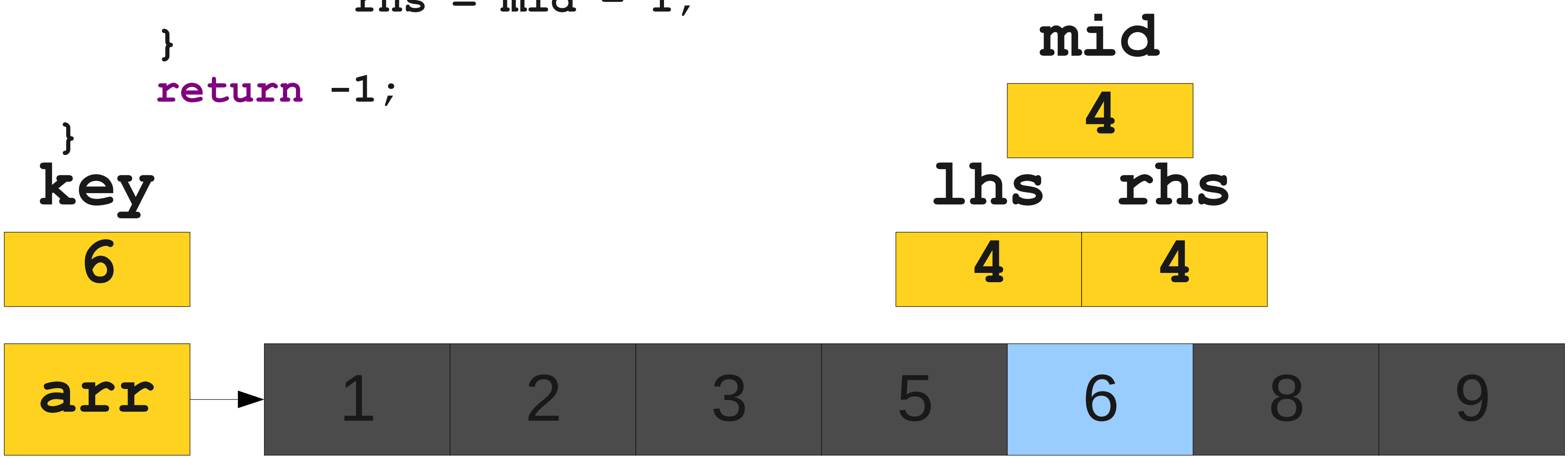
Binary Search

```
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;
}
```



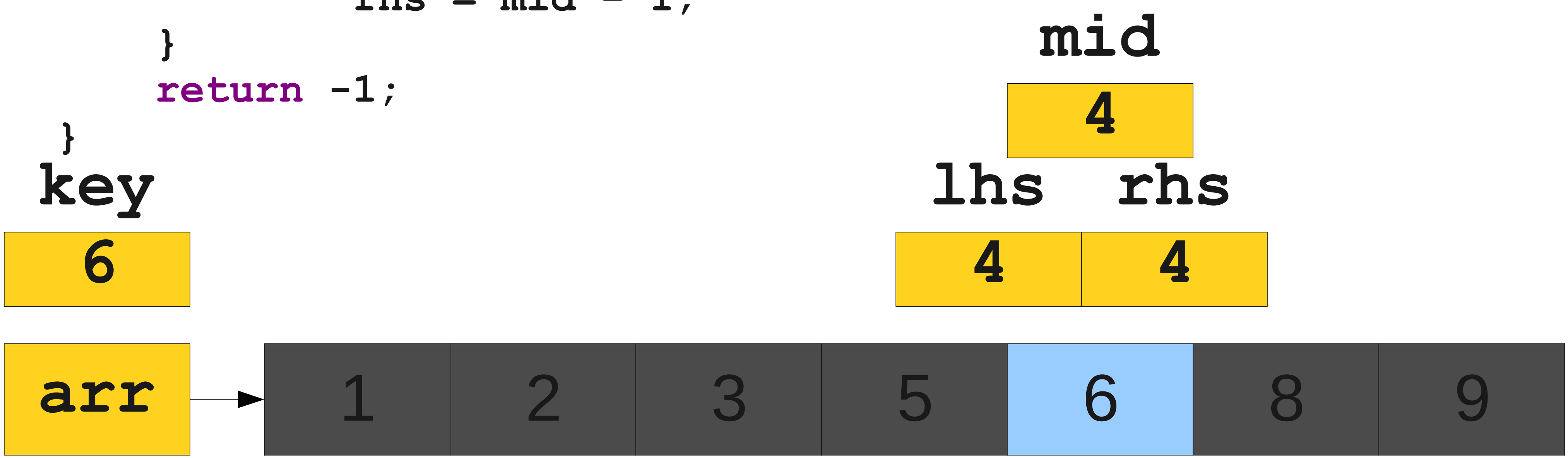
Binary Search

```
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;
}
```



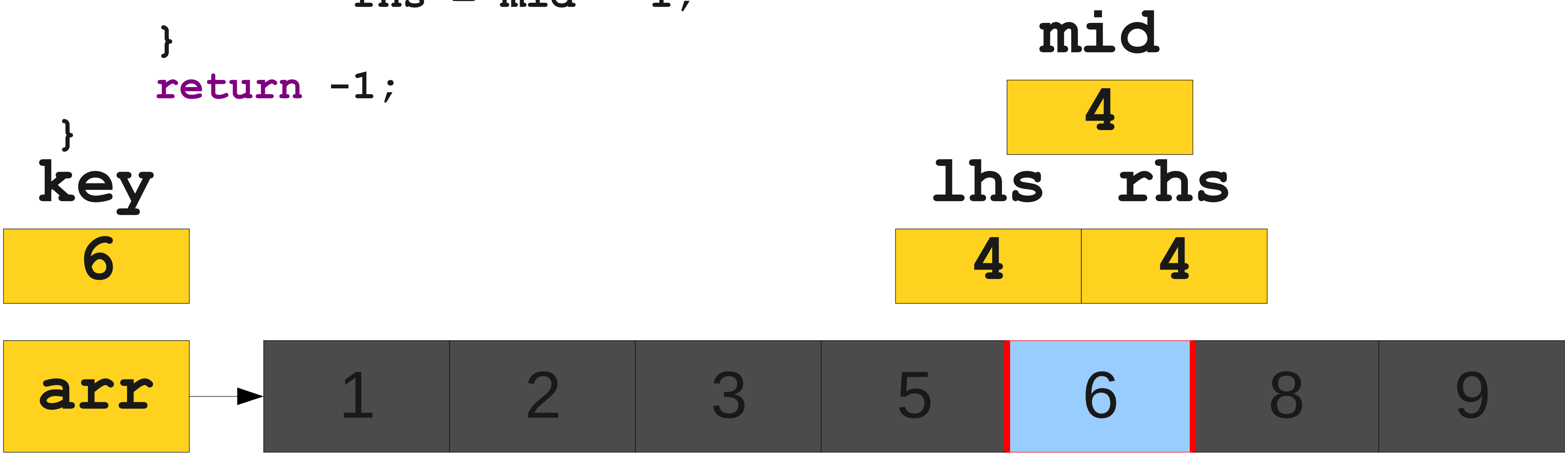
Binary Search

```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;
}
```



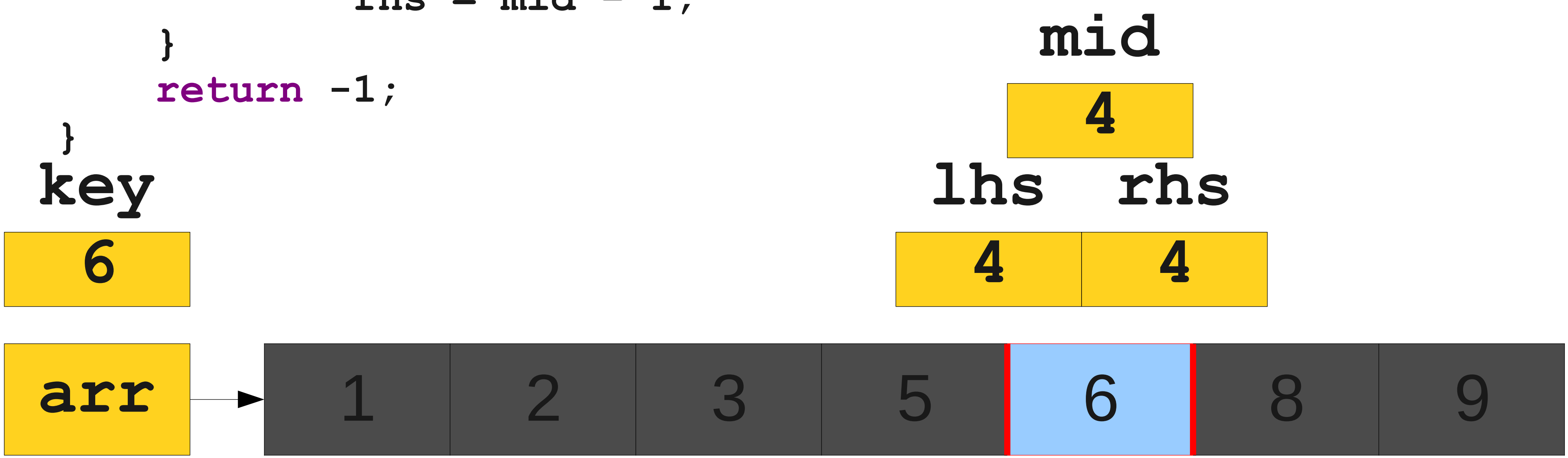
Binary Search

```
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;
}
```



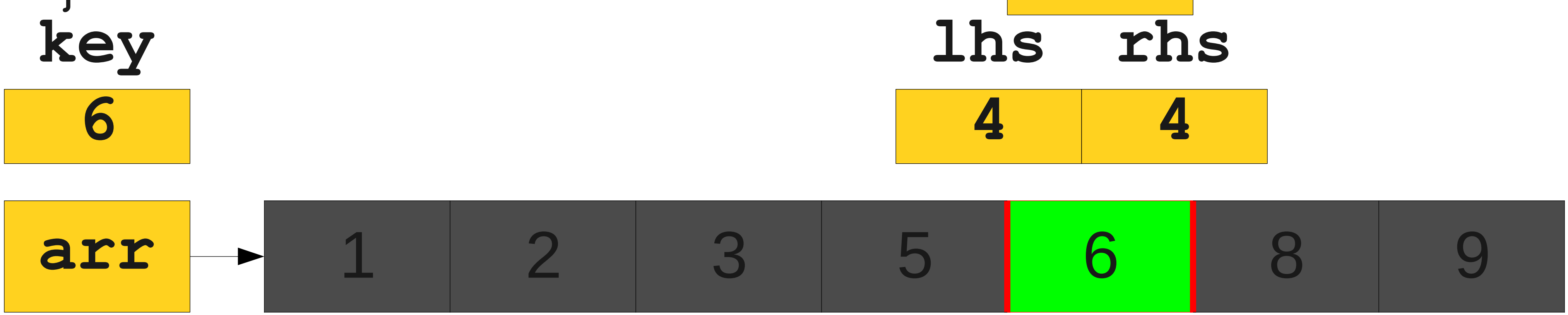
Binary Search

```
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;
```


```
}
```

mid

4

key lhs rhs

6 4 4

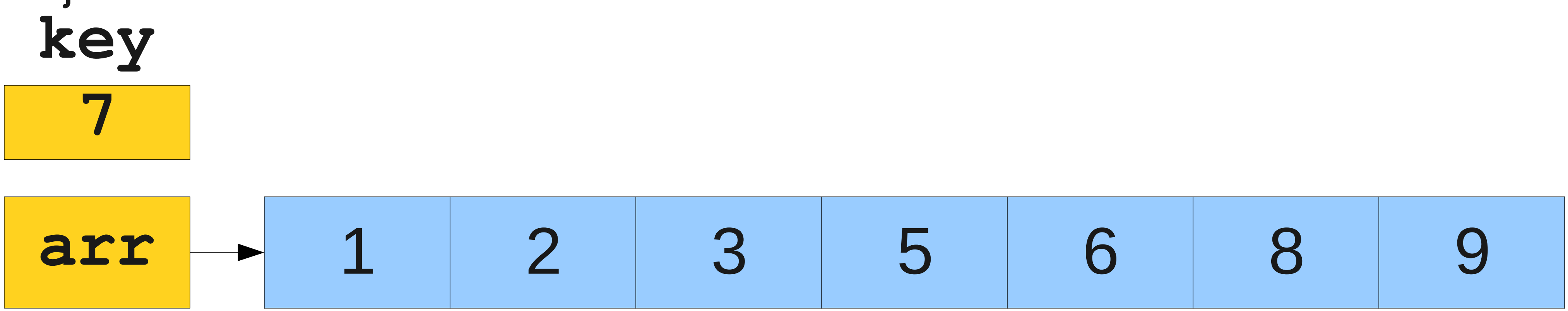
arr → | 1 | 2 | 3 | 5 | 6 | 8 | 9 |

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;
}
```

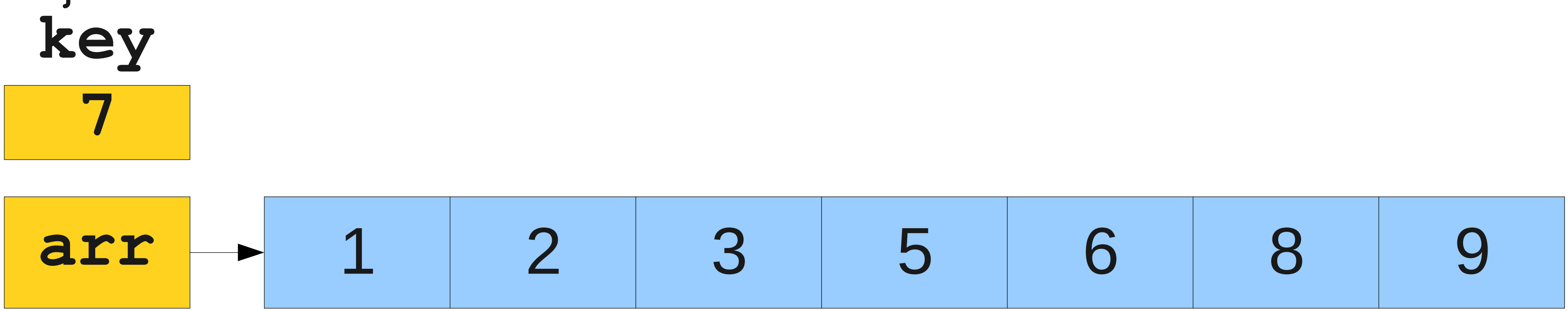
Binary Search

```
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;
}
```



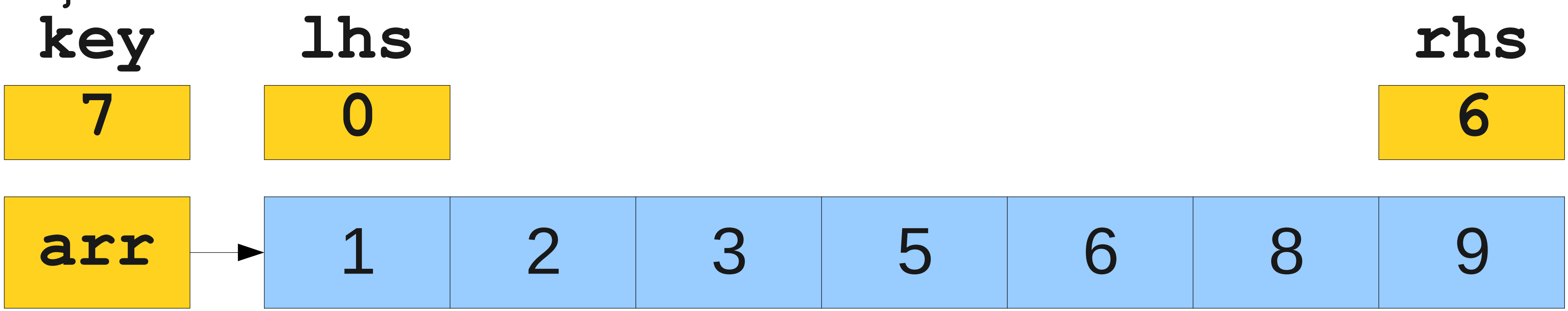
Binary Search

```
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;
}
```



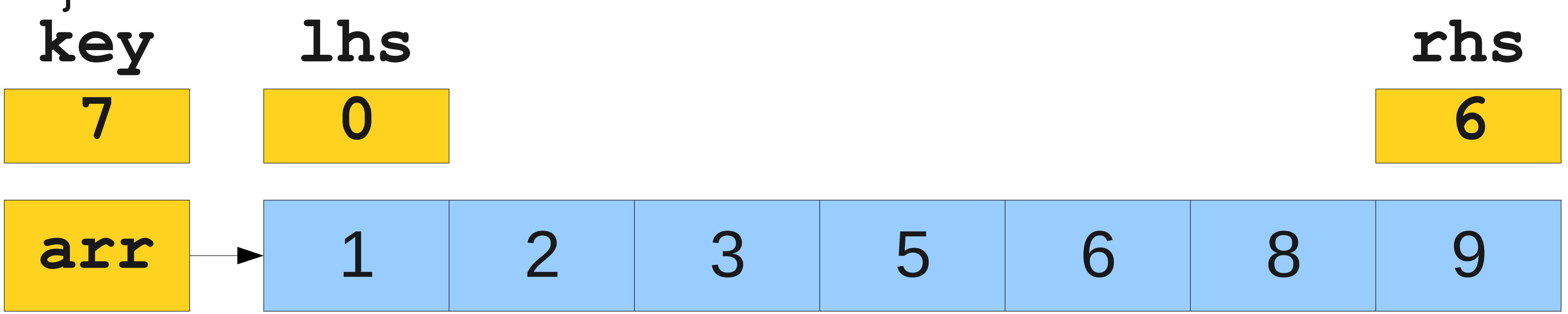
Binary Search

```
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;
}
```



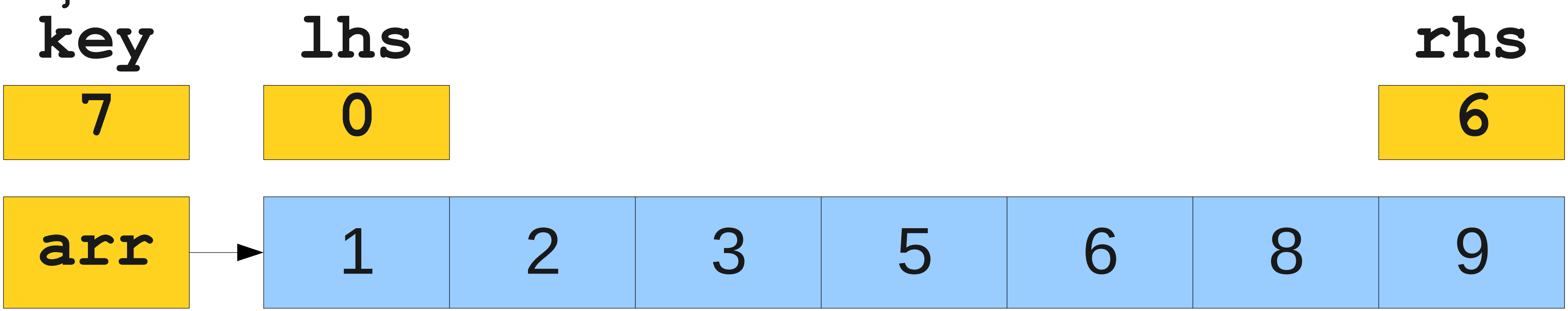
Binary Search

```
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;
}
```



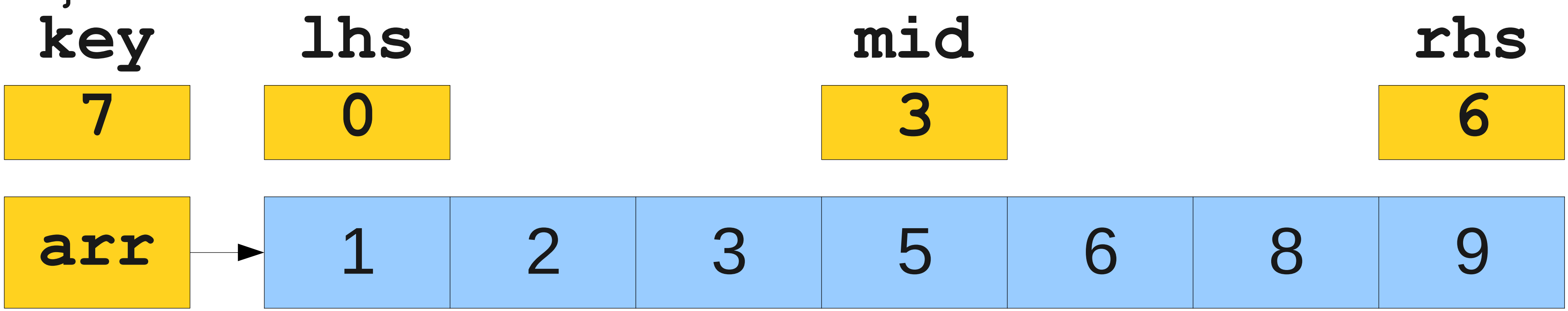
Binary Search

```
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;
}
```



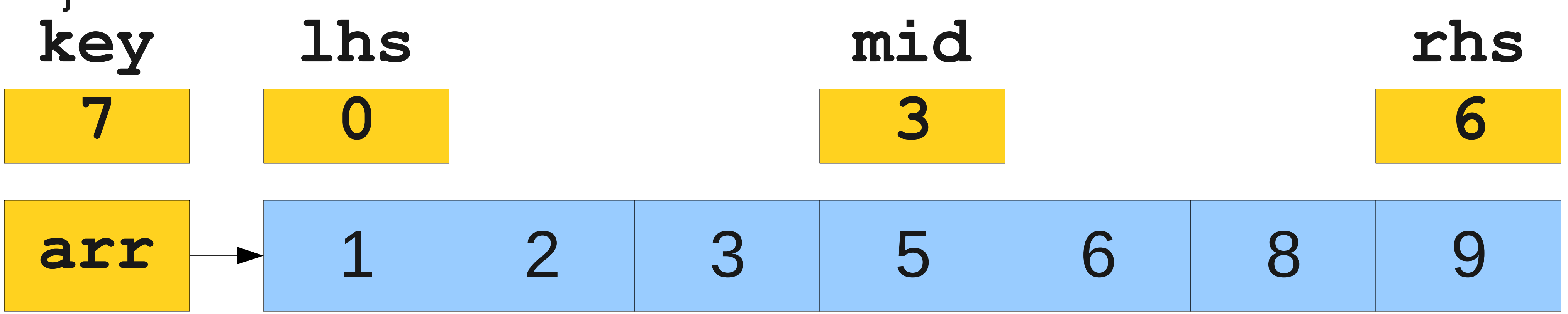
Binary Search

```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;
}
```



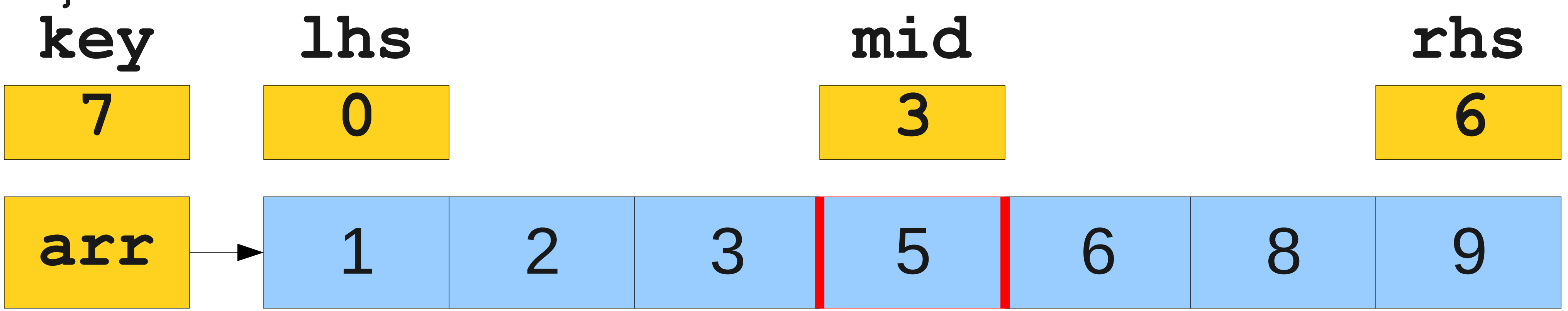
Binary Search

```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;
}
```



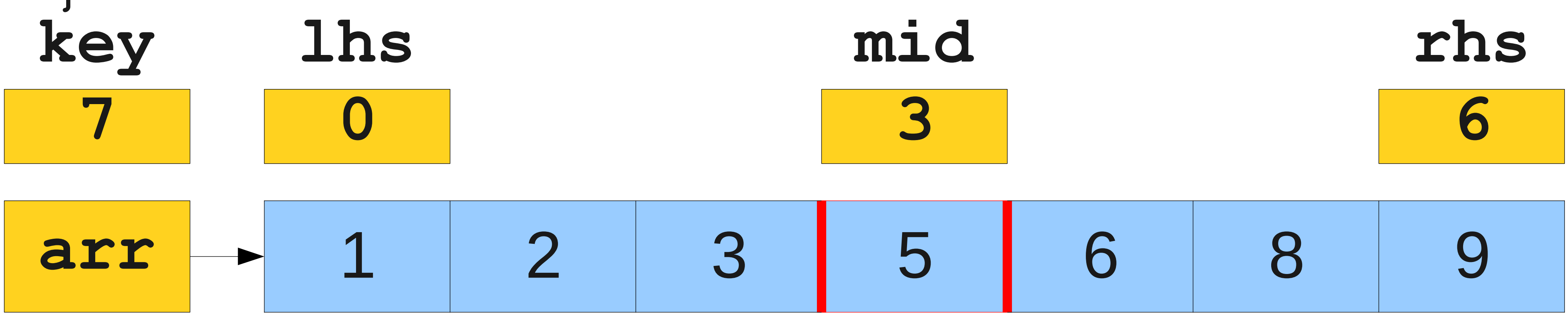
Binary Search

```
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;
}
```



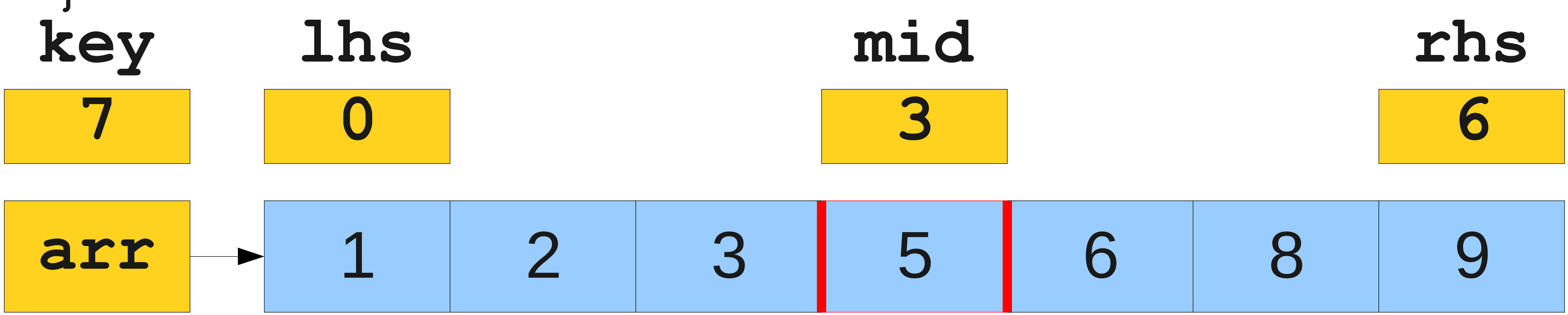
Binary Search

```
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;
}
```



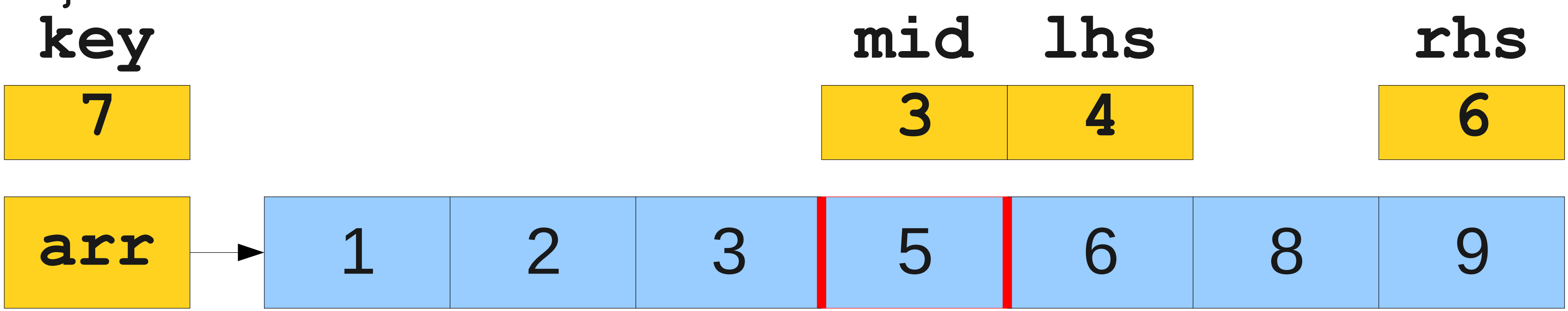
Binary Search

```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;
}
```



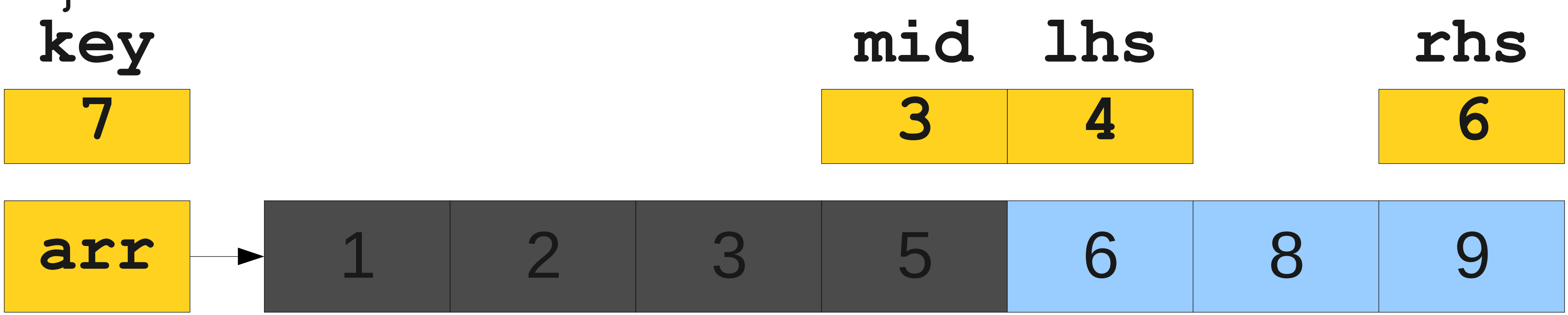
Binary Search

```
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;
}
```



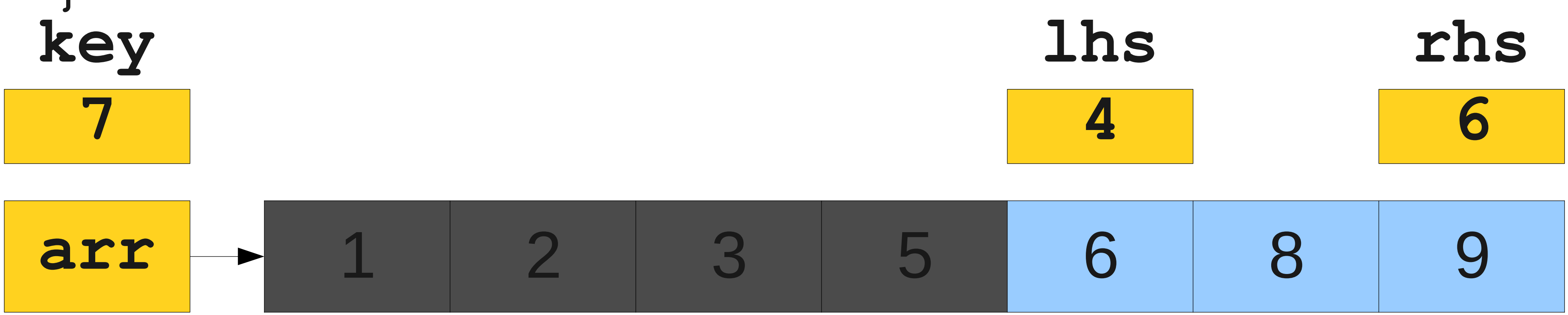
Binary Search

```
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;
}
```



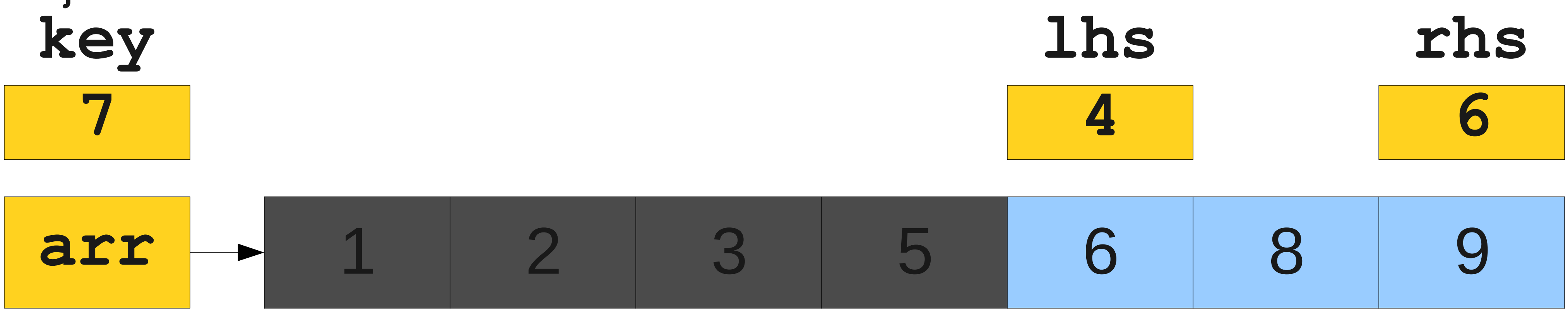
Binary Search

```
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;
}
```



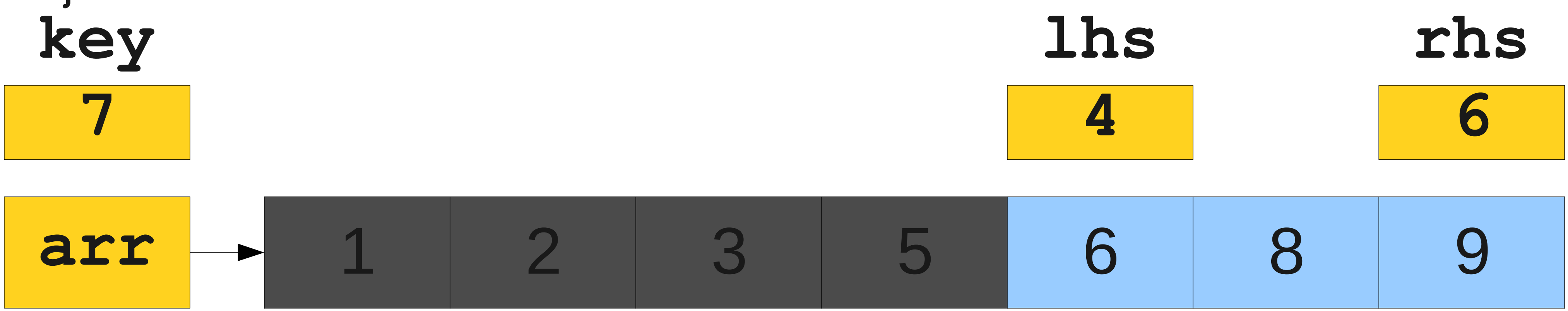
Binary Search

```
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;
}
```



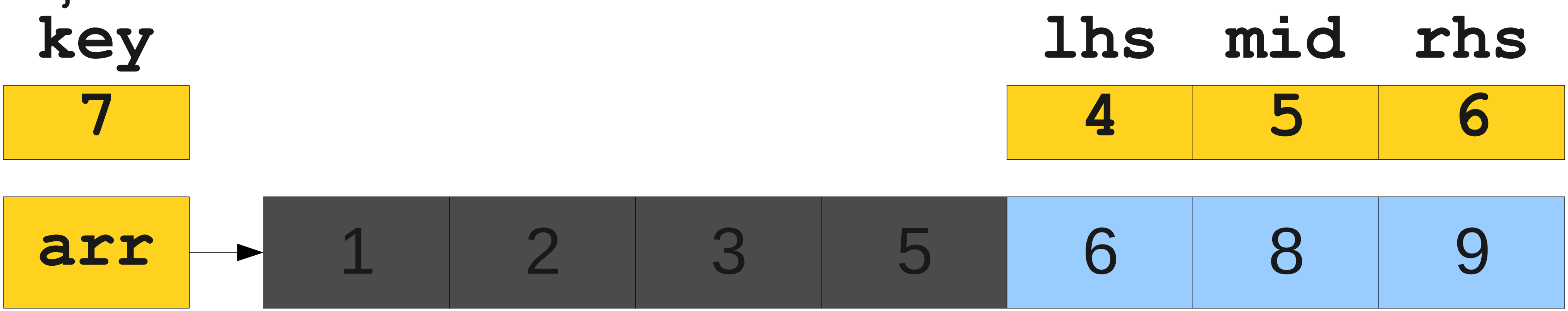
Binary Search

```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;
}
```



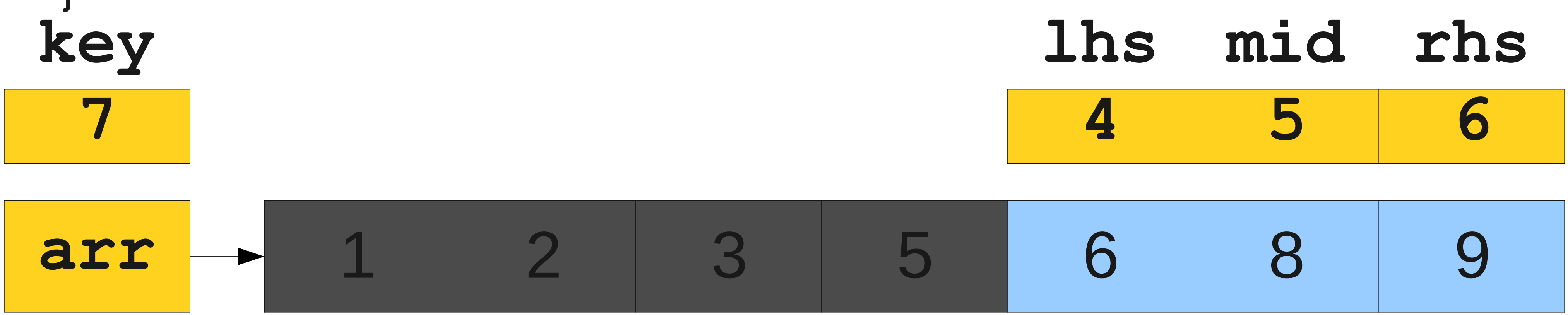
Binary Search

```
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;
}
```



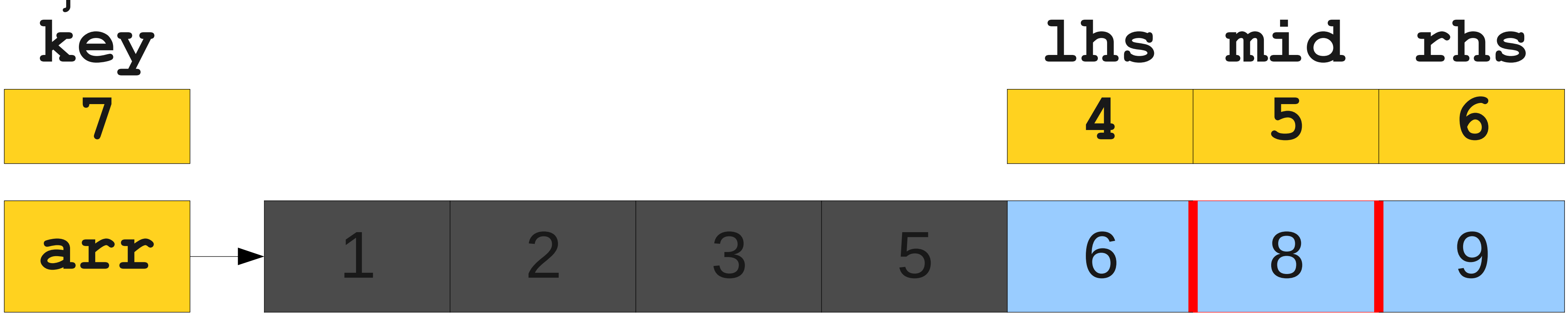
Binary Search

```
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;
}
```



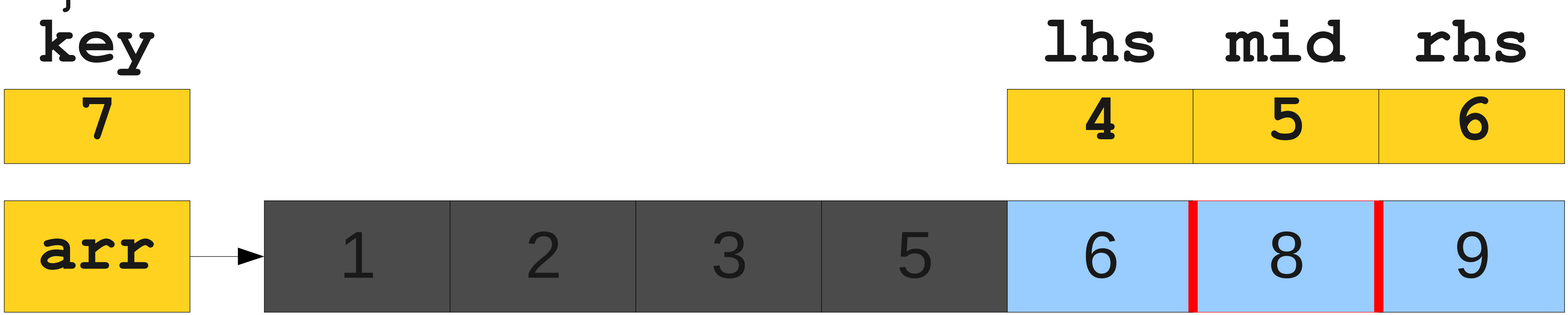
Binary Search

```
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;
}
```



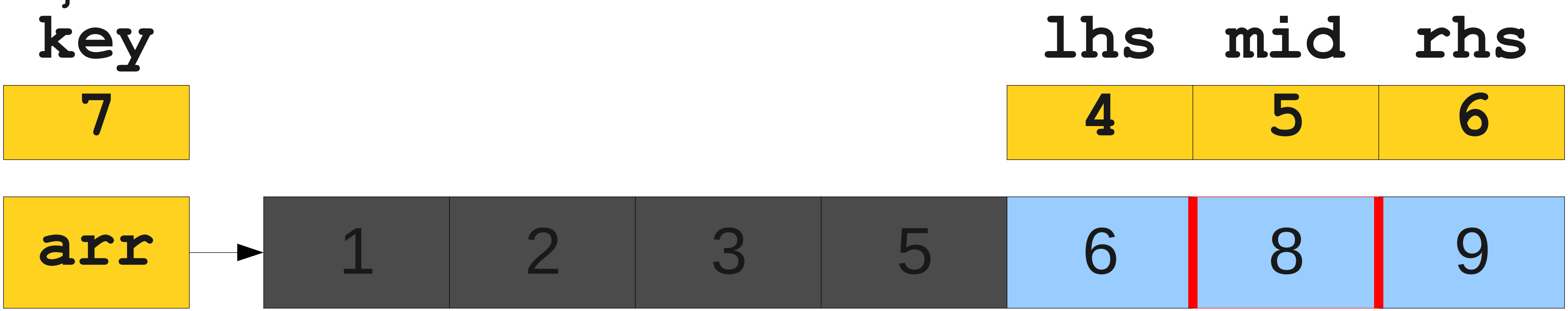
Binary Search

```
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;
}
```



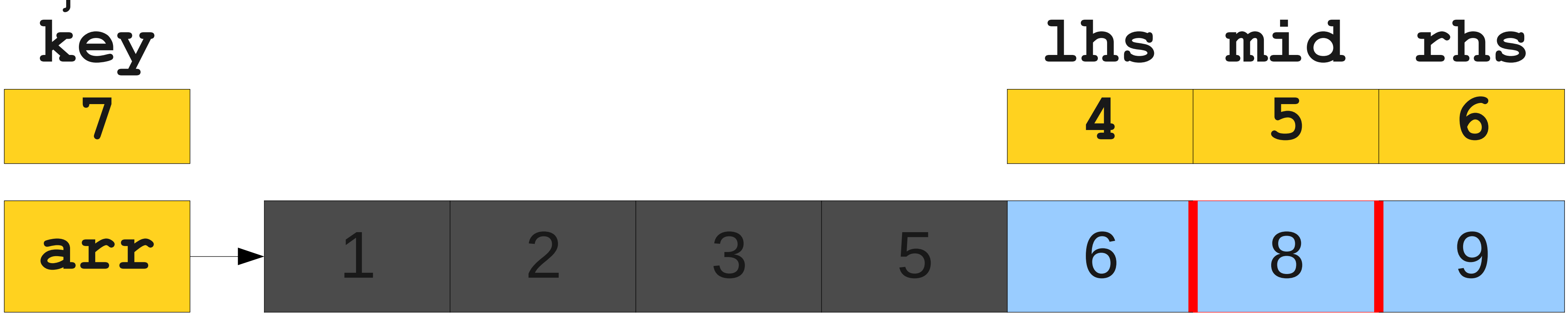
Binary Search

```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;
}
```



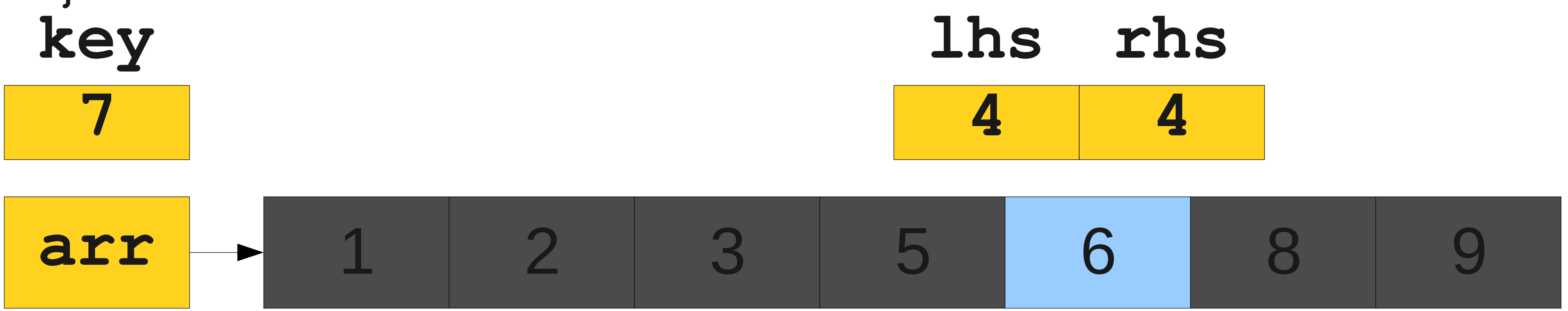
Binary Search

```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;
}
```



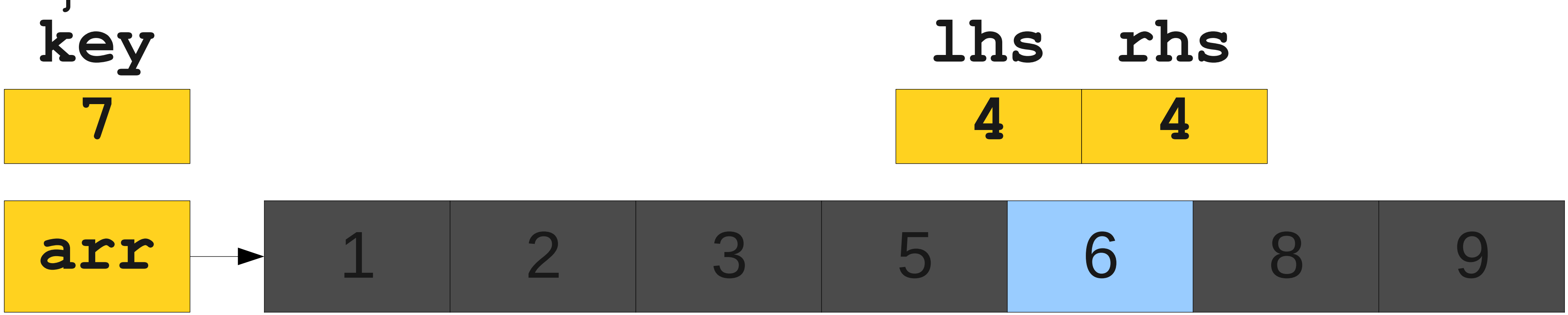
Binary Search

```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;
}
```



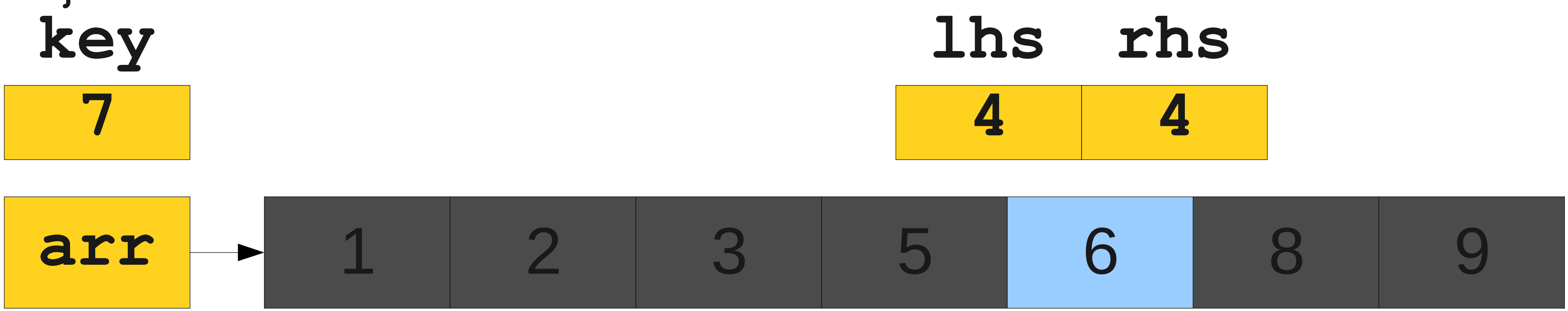
Binary Search

```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;
}
```



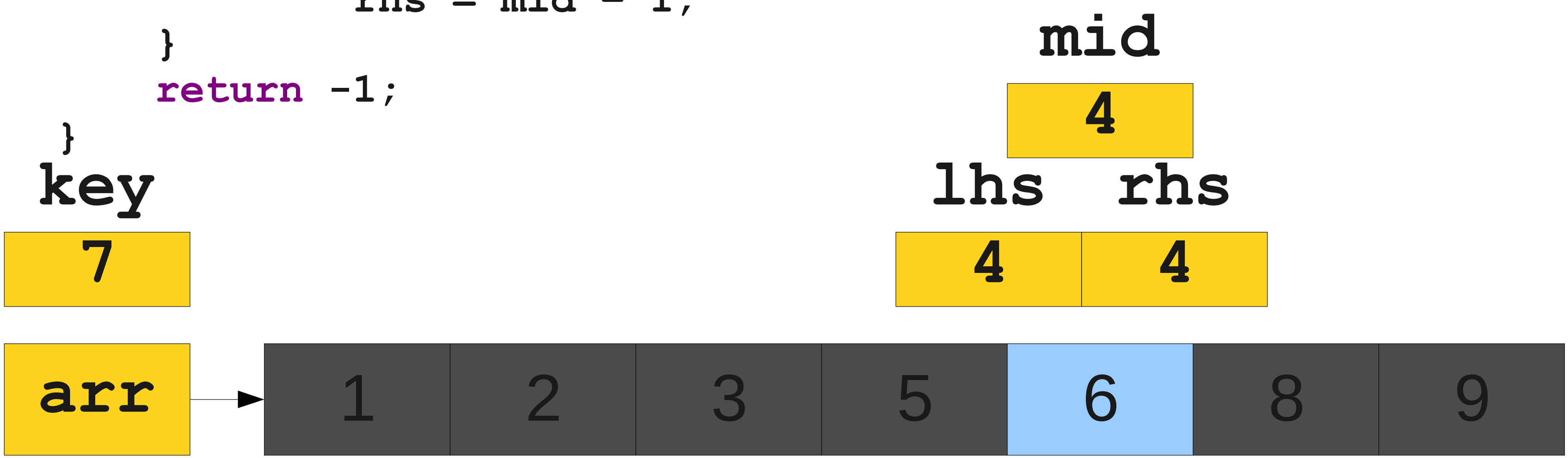
Binary Search

```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;
}
```



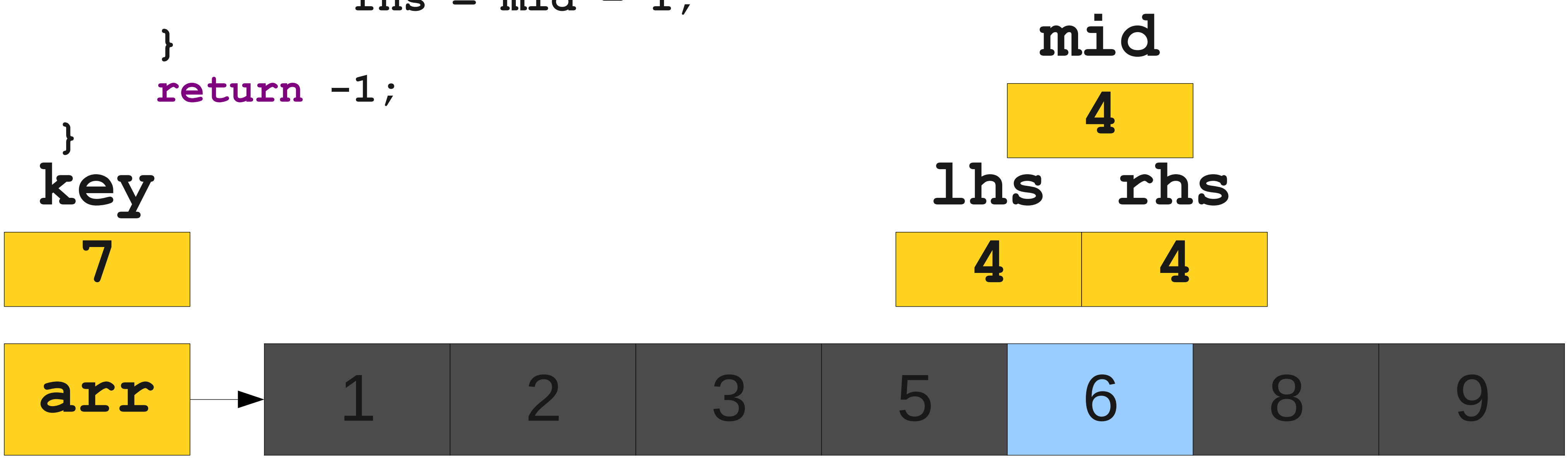
Binary Search

```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;
}
```



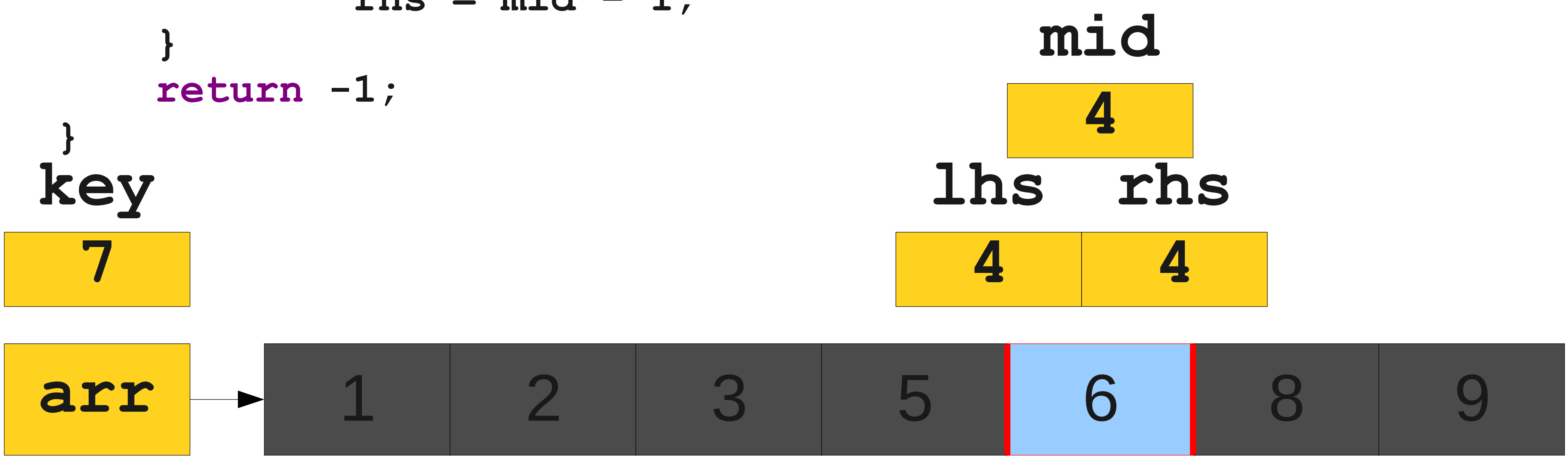
Binary Search

```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;
}
```



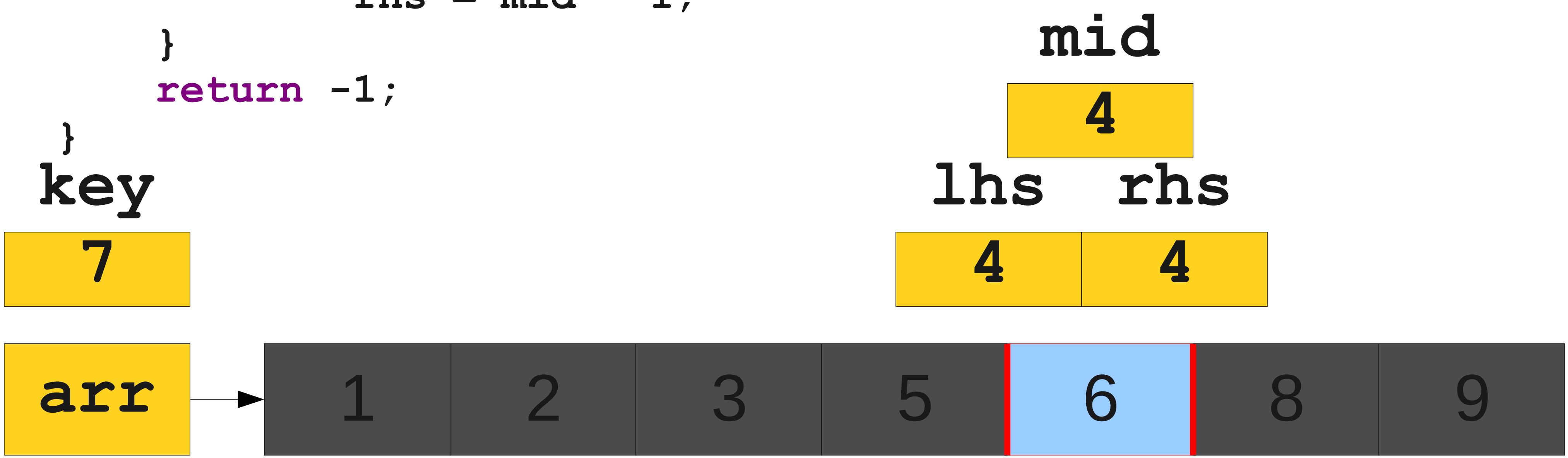
Binary Search

```
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;
}
```



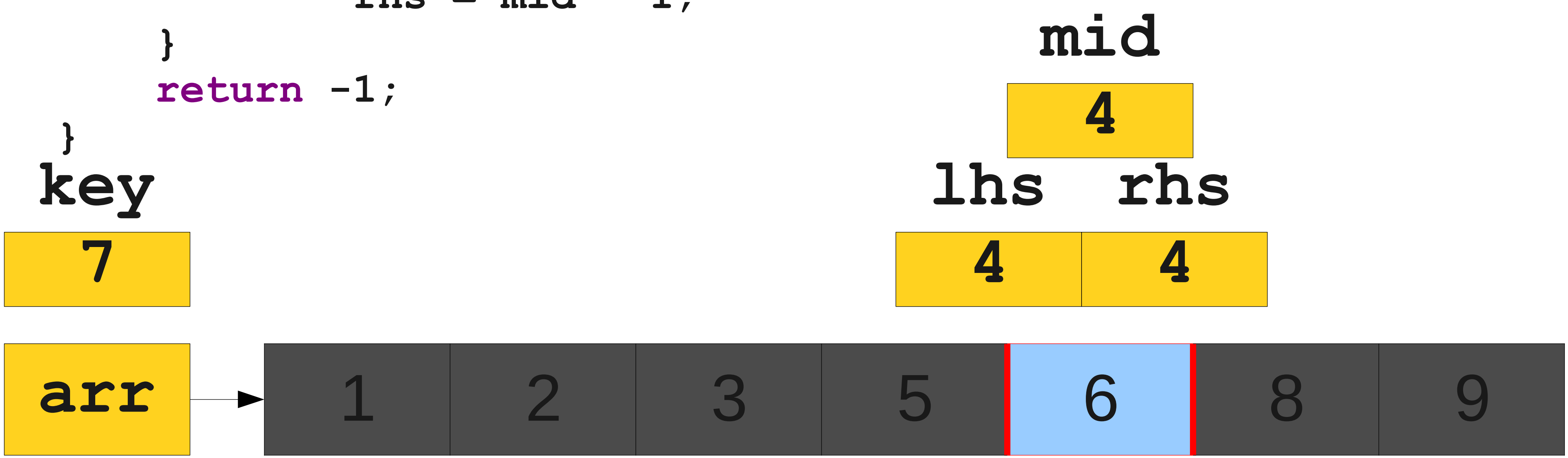
Binary Search

```
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;
}
```



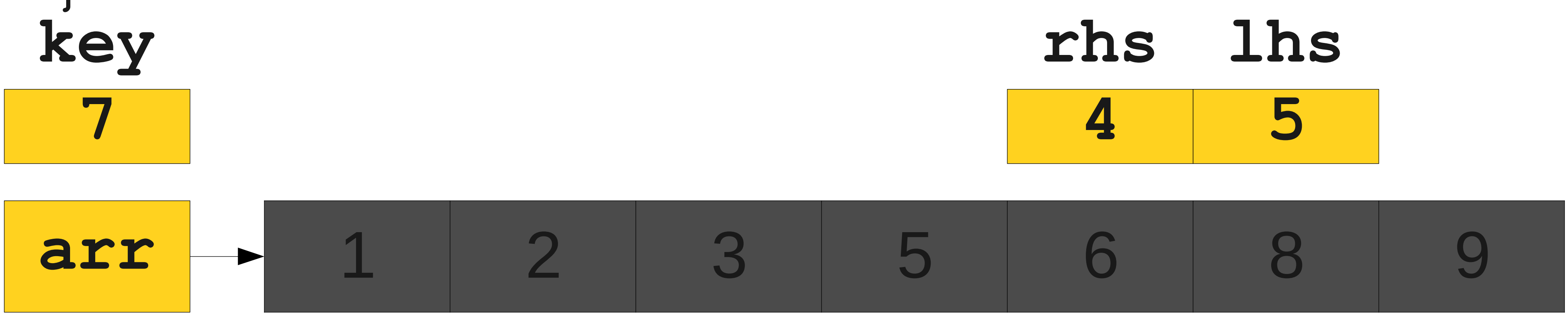
Binary Search

```
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;
}
```



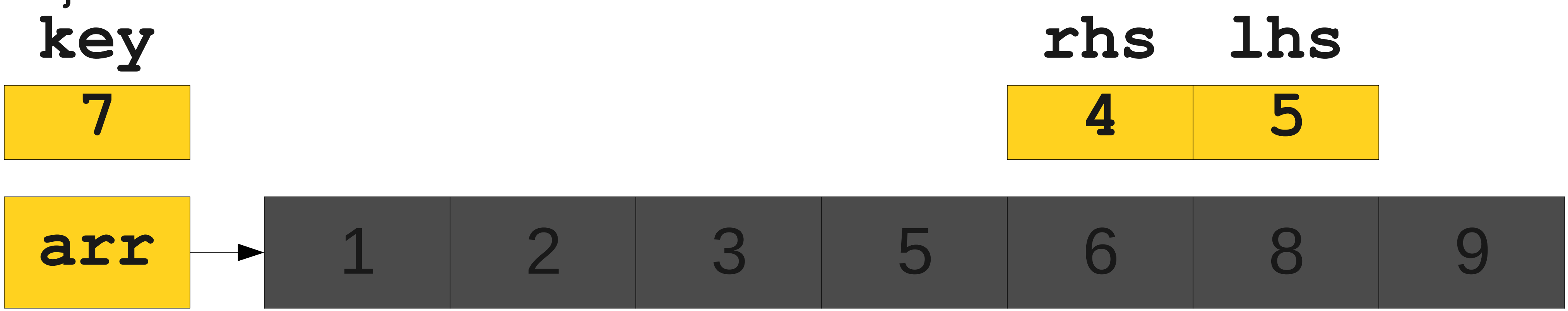
Binary Search

```
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;
}
```



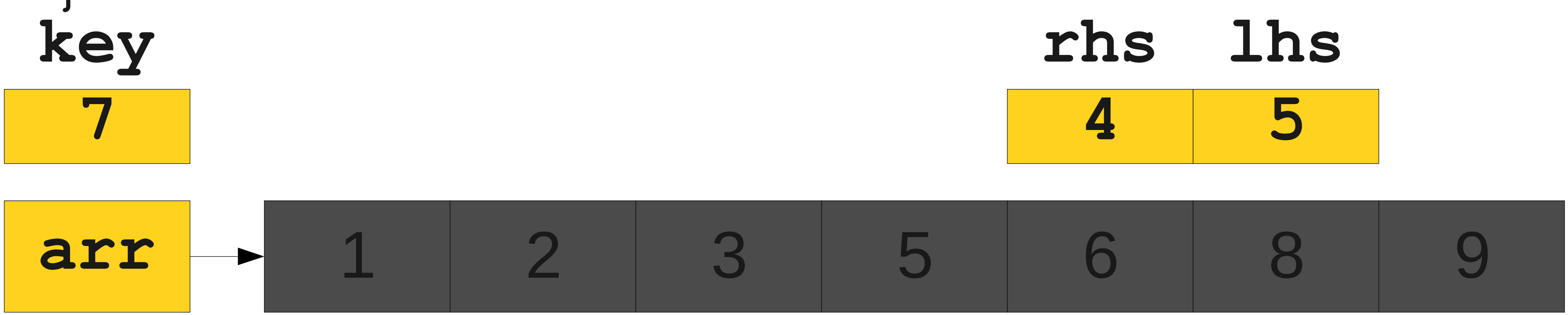
Binary Search

```
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 <=
        int mid =

        if (arr[m
            retur
        else if (
            lhs =
        else
            rhs =
    }
    return -1;
}
```

key

7

rhs lhs

4 5

arr →

1	2	3	5	6	8	9

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 $\mathbf{\log_2 N}$ 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.

Lower bound

- Search space size: N
  Are $log\ N$ queries necessary?
- Yes. When each query is a yes/no type, then the search space gets divided into two parts with each query (some solutions correspond to yes and others to no).
- One of the parts will be at least $N/2$.
- In worst case, with each query, we get the larger of the two parts.
- To reduce the search space size to 1, we need $log\ N$ queries

Lower bound

- Search space size: N
  Are $log\ N$ queries necessary? Another argument.

- Suppose you make k queries.

- There are 2^k possible outcomes.

- If $2^k < N$, then two different elements must lead to same outcomes for all queries.

- We won't be able to say which of the two is correct.

Idea is that the number of the queries is logN is roughly number of digits in N , so asking question add a bit of info that is the bit 0 or 1 , and we need logN such info

Lower bound

- Search space size: N
  Are $\log N$ queries necessary?
- Another argument based on information theory.
- Each yes/no query gives us 1 bit of information.
- The final answer is a number between 1 and N, and thus, requires $\log N$ bits of information.
- Hence, $\log N$ queries are necessary.
- Ignore this argument if it is hard to digest.

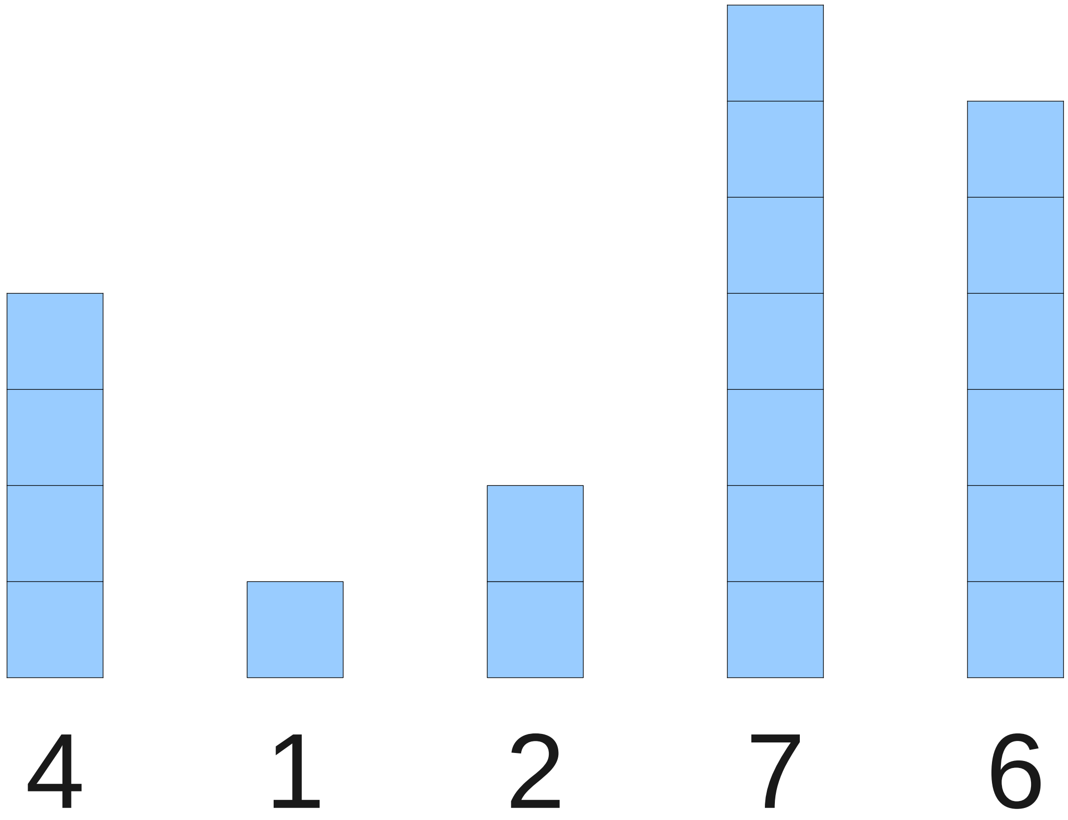
Sorting

Bubble Sort

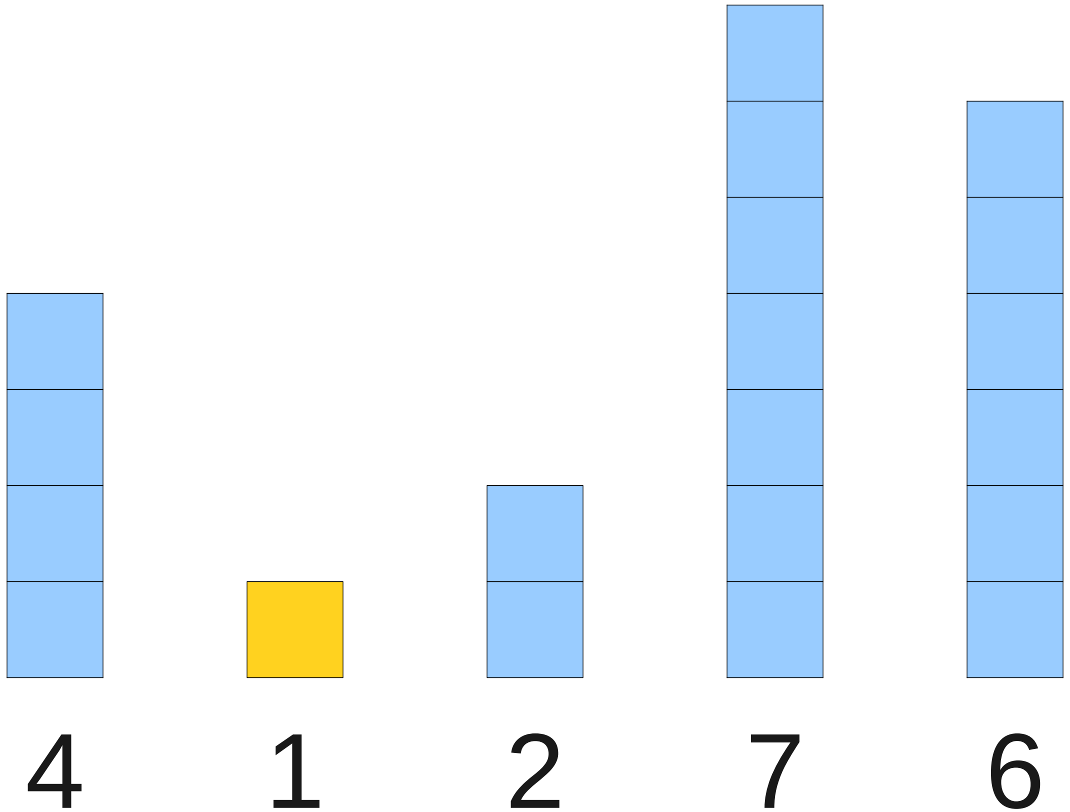
- Until the array is sorted:
  - Look at each adjacent pair of elements.
  - If they are out of order, swap them.

An Second Idea: **Selection Sort**

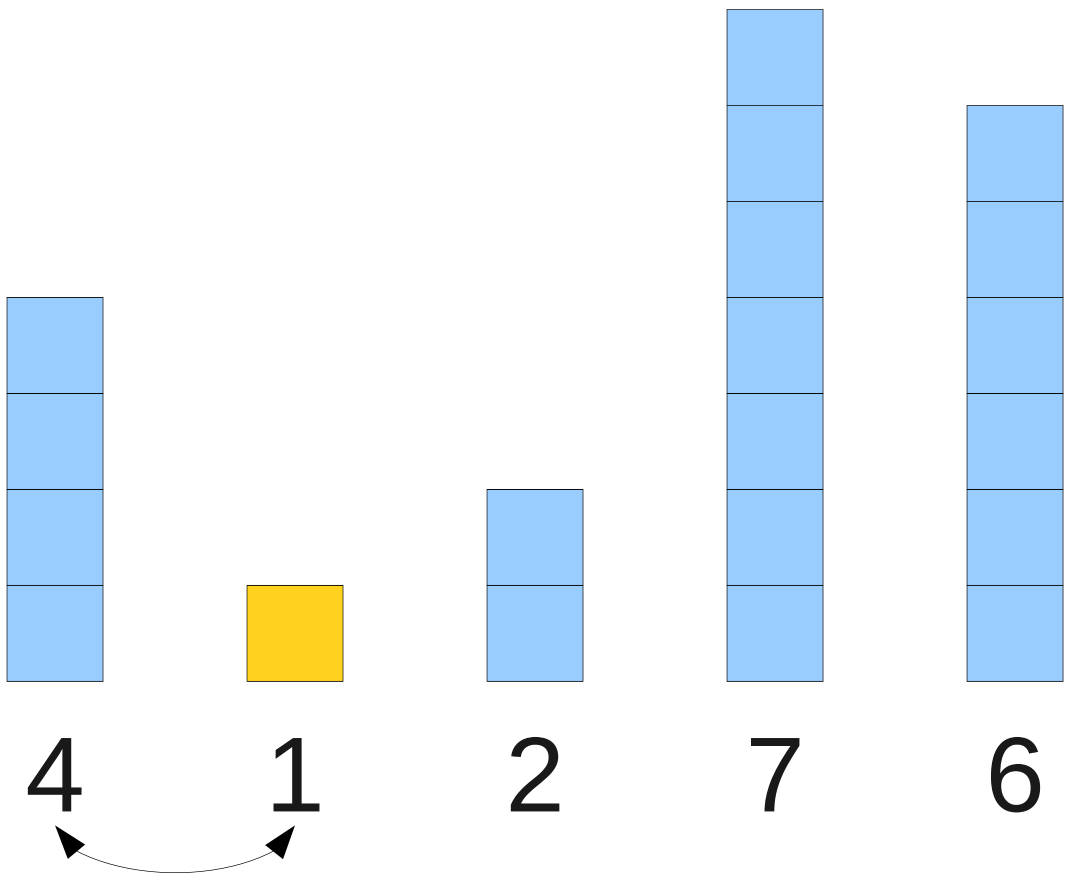
An Second Idea: Selection Sort



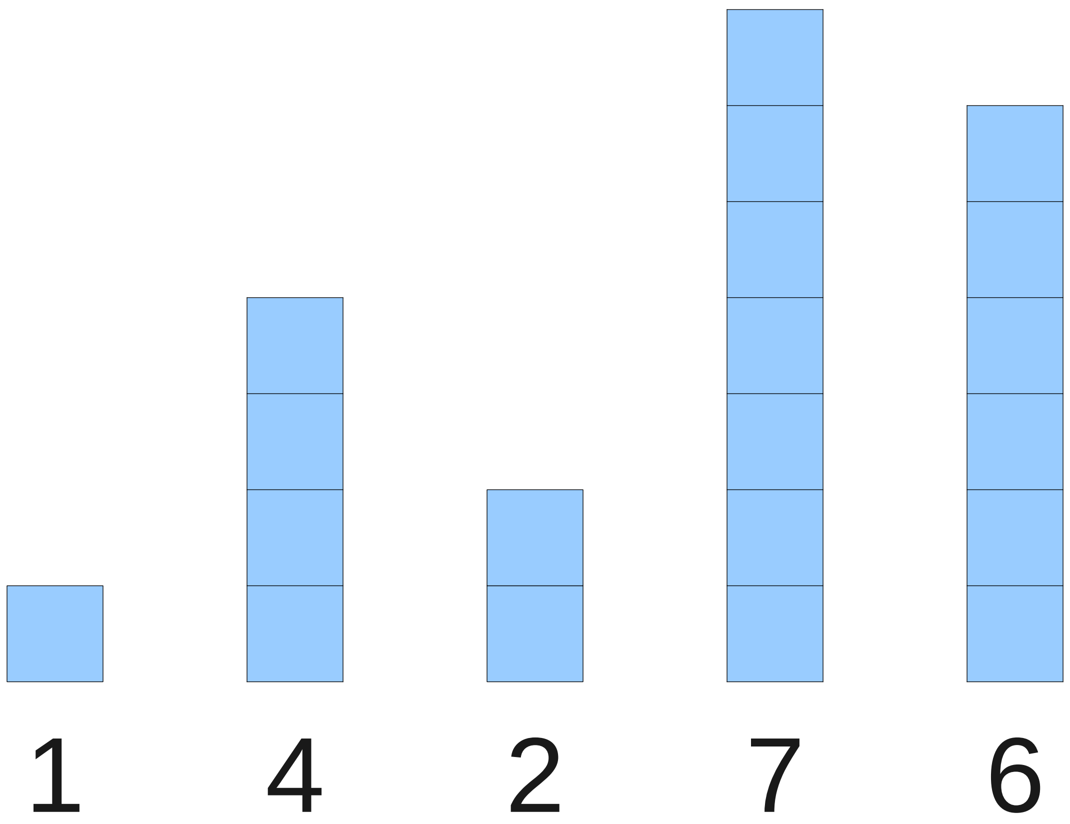
An Second Idea: **Selection Sort**



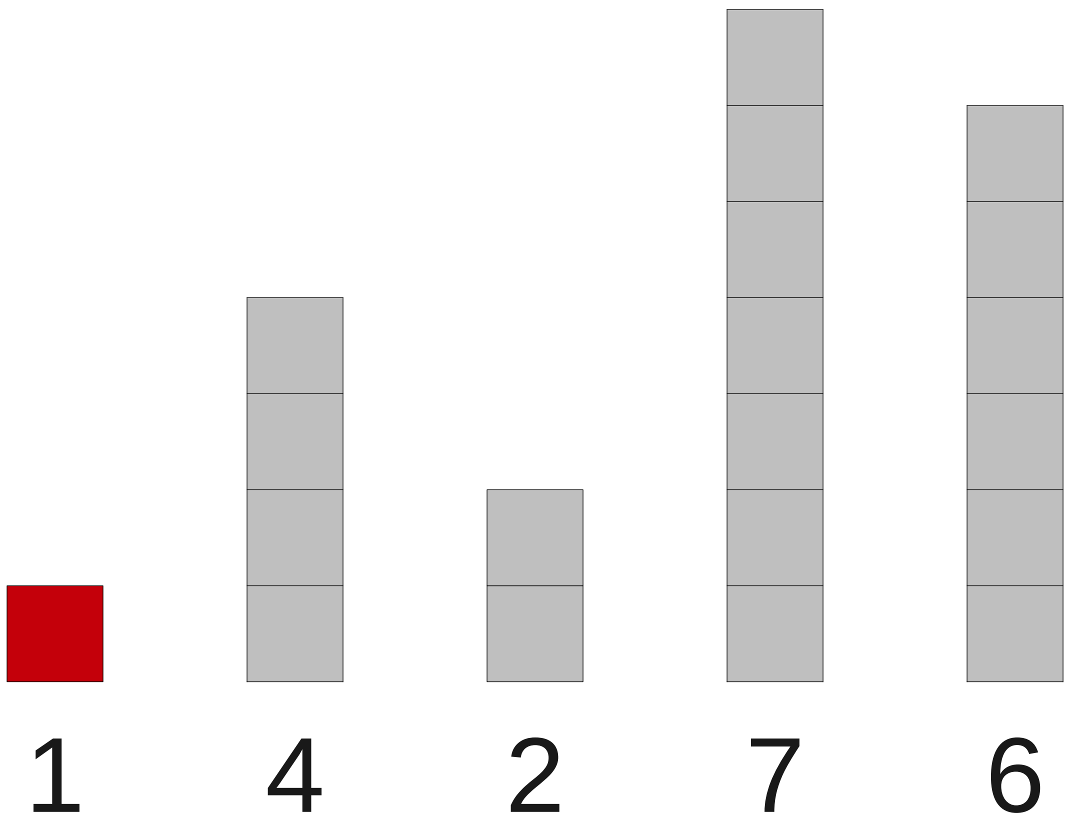
An Second Idea: **Selection Sort**



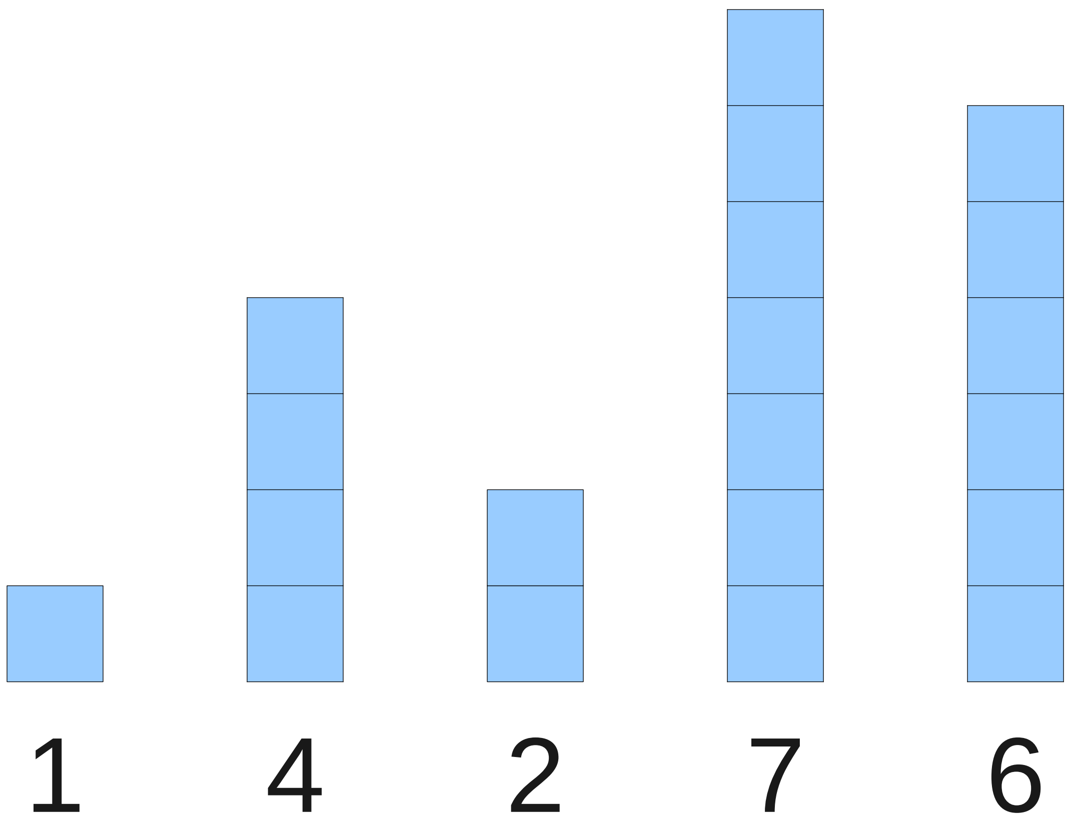
An Second Idea: Selection Sort



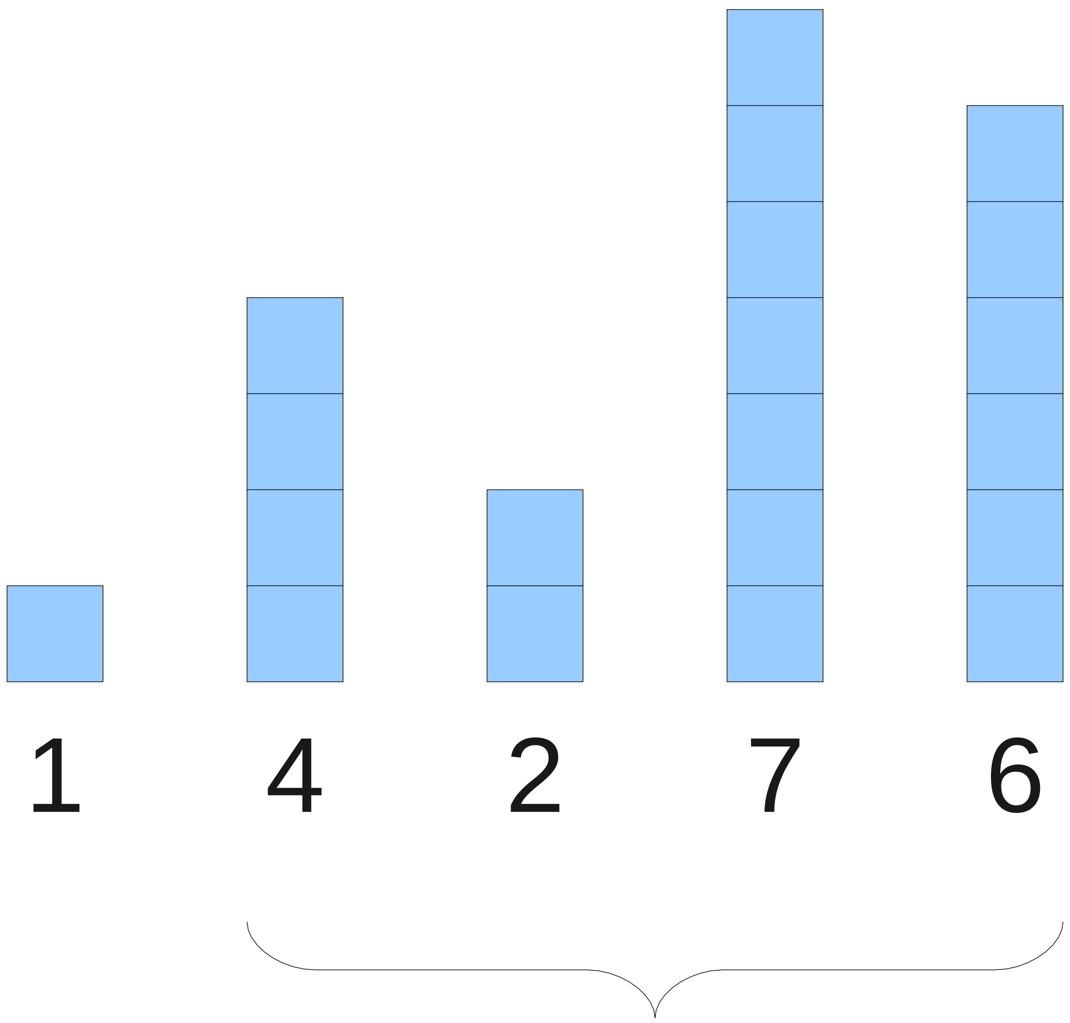
An Second Idea: Selection Sort



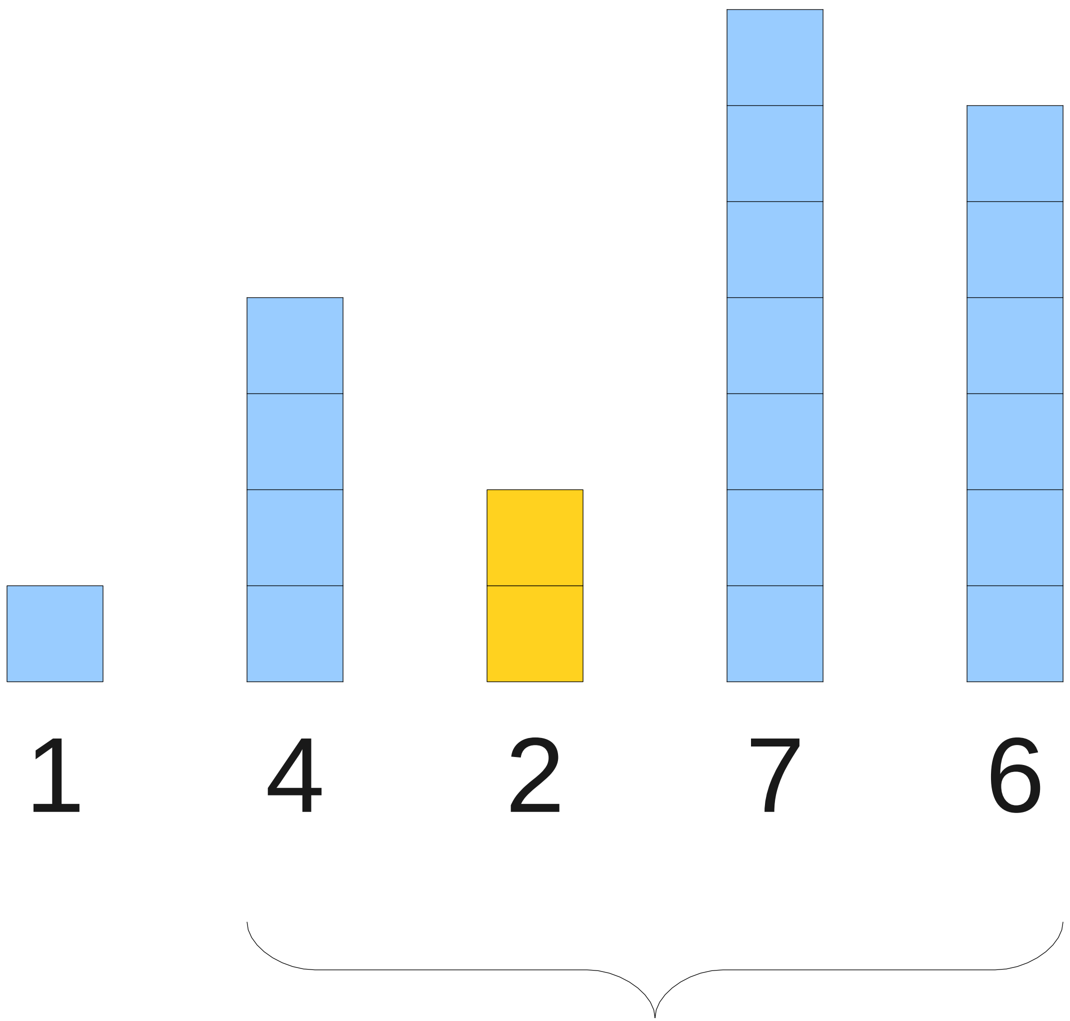
An Second Idea: Selection Sort



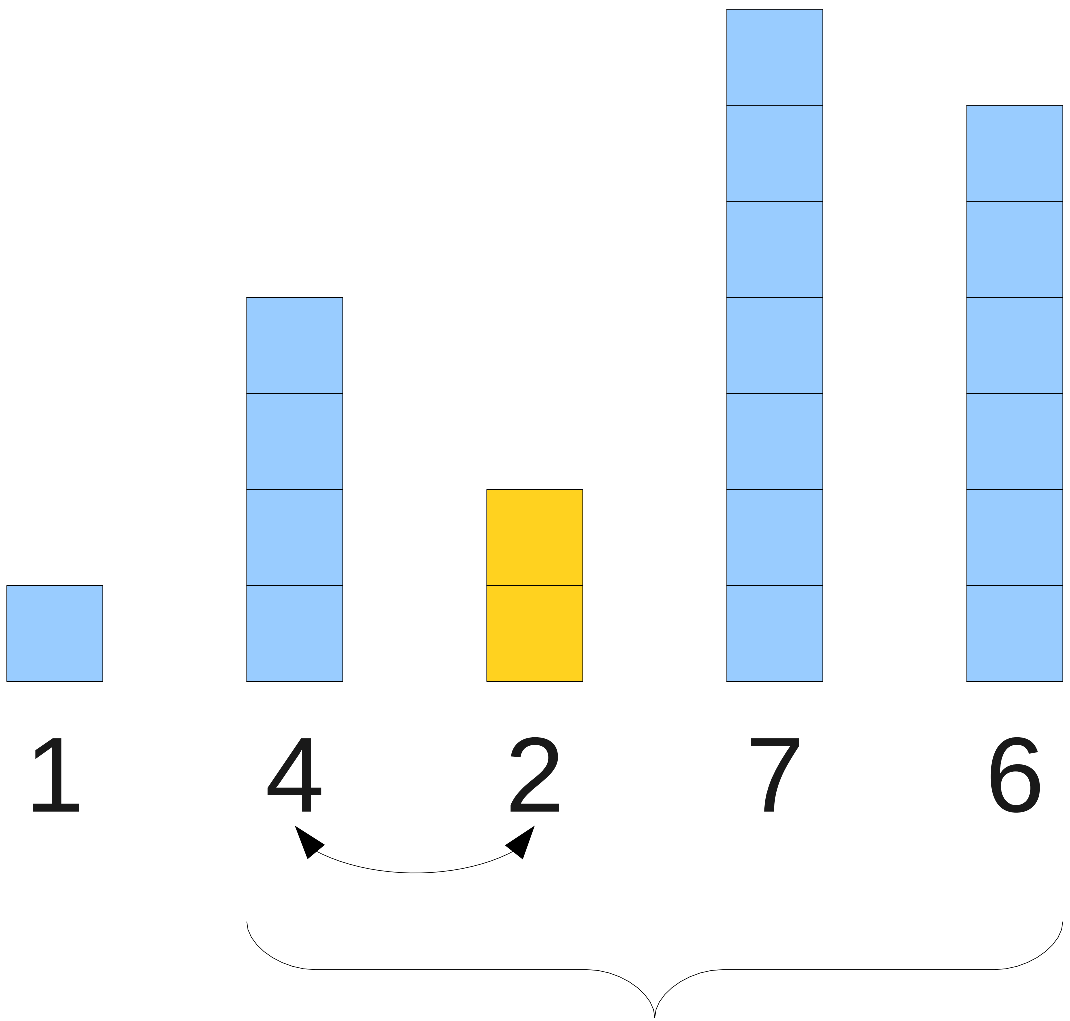
An Second Idea: Selection Sort



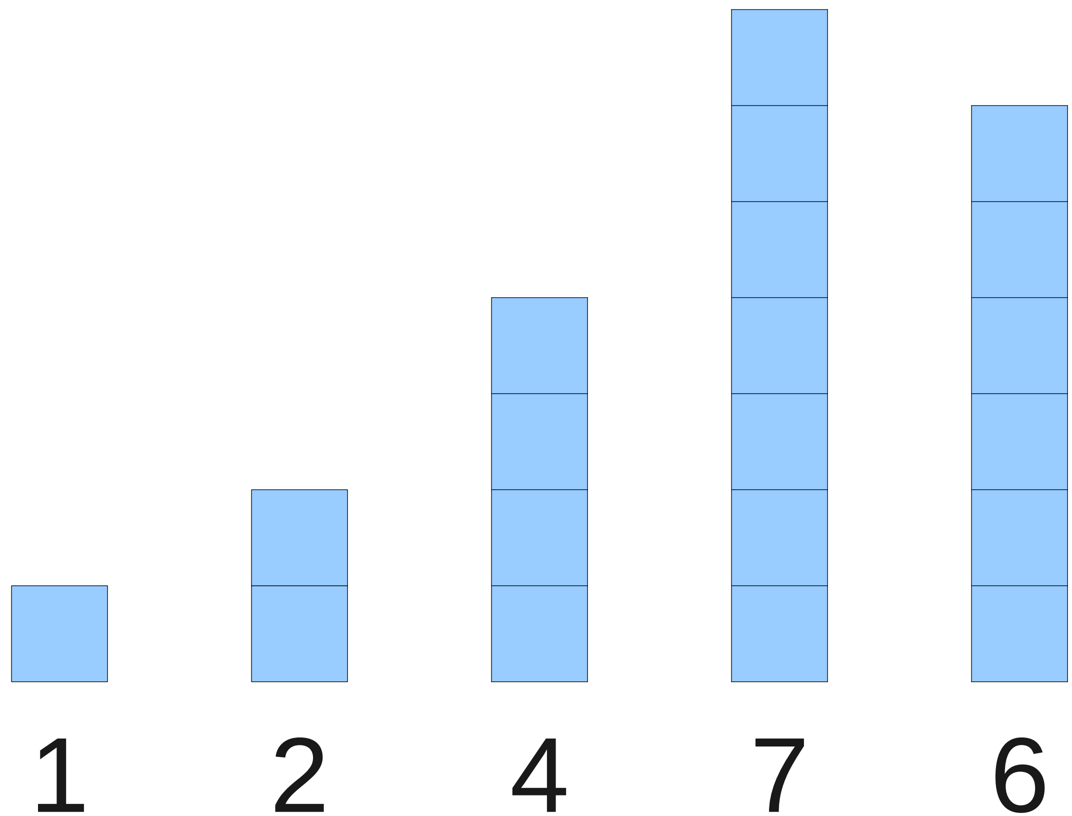
An Second Idea: **Selection Sort**



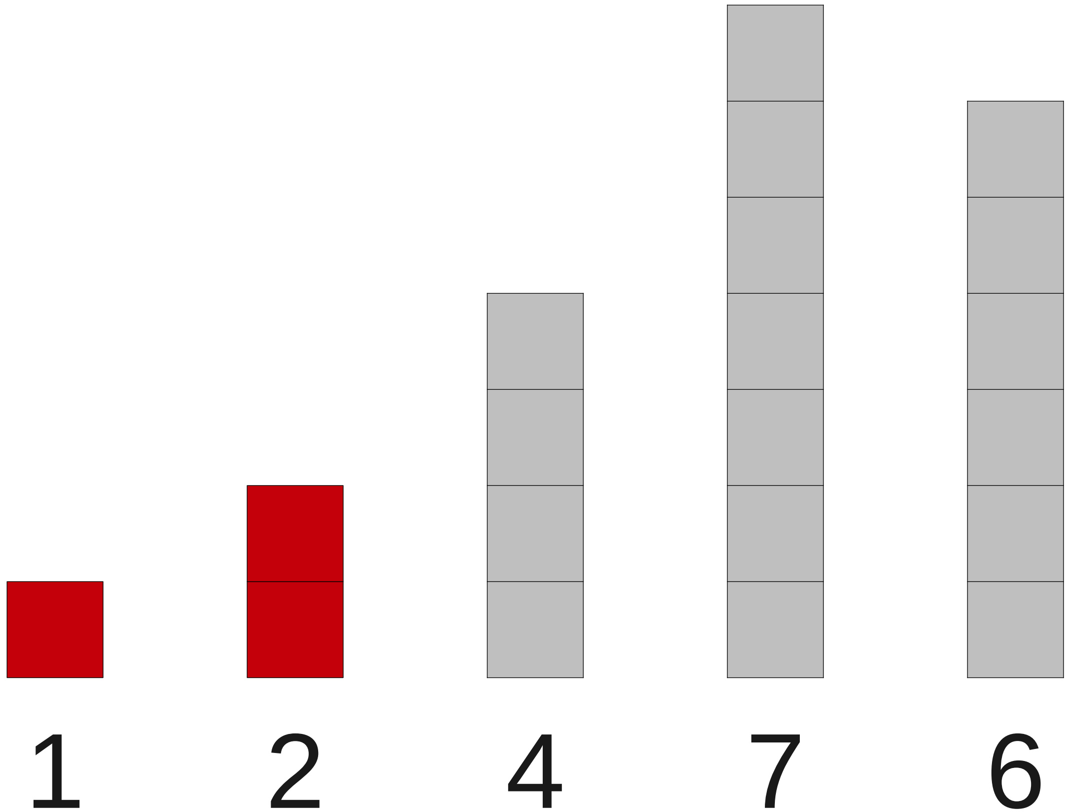
An Second Idea: **Selection Sort**



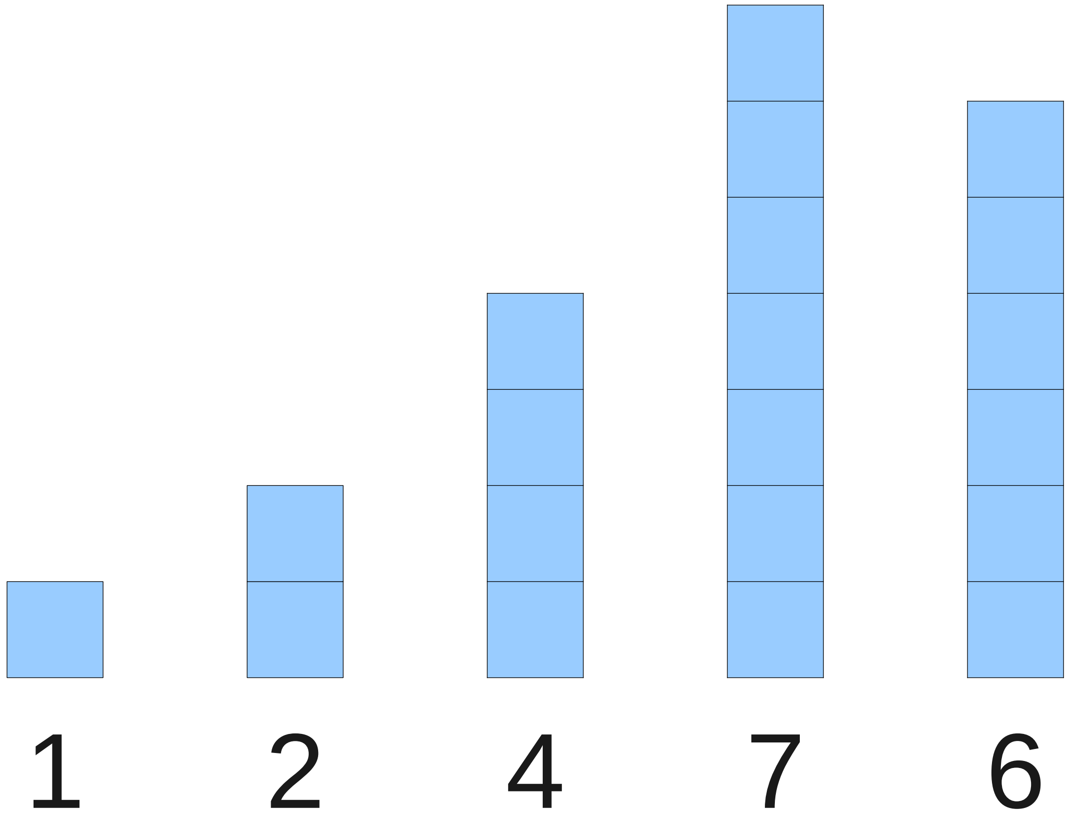
An Second Idea: **Selection Sort**



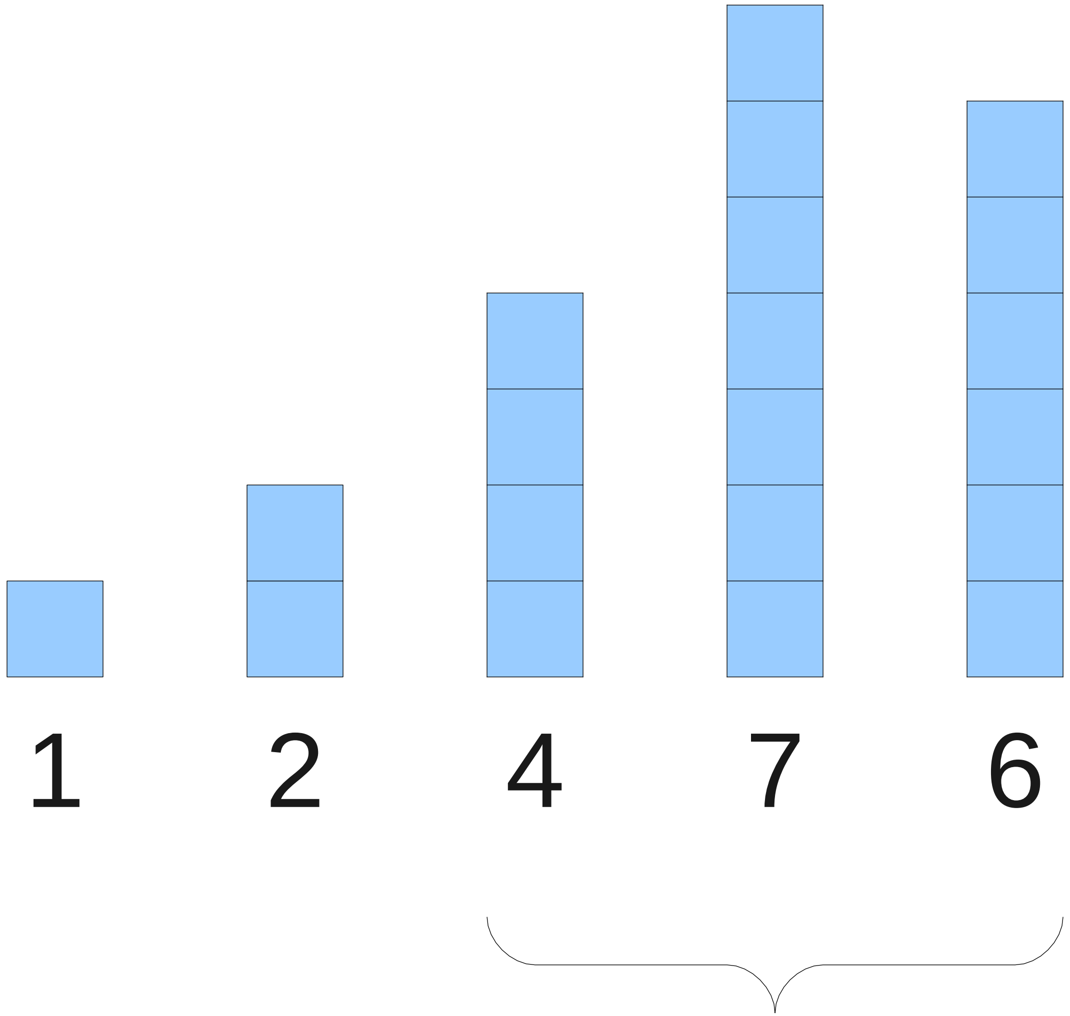
An Second Idea: Selection Sort



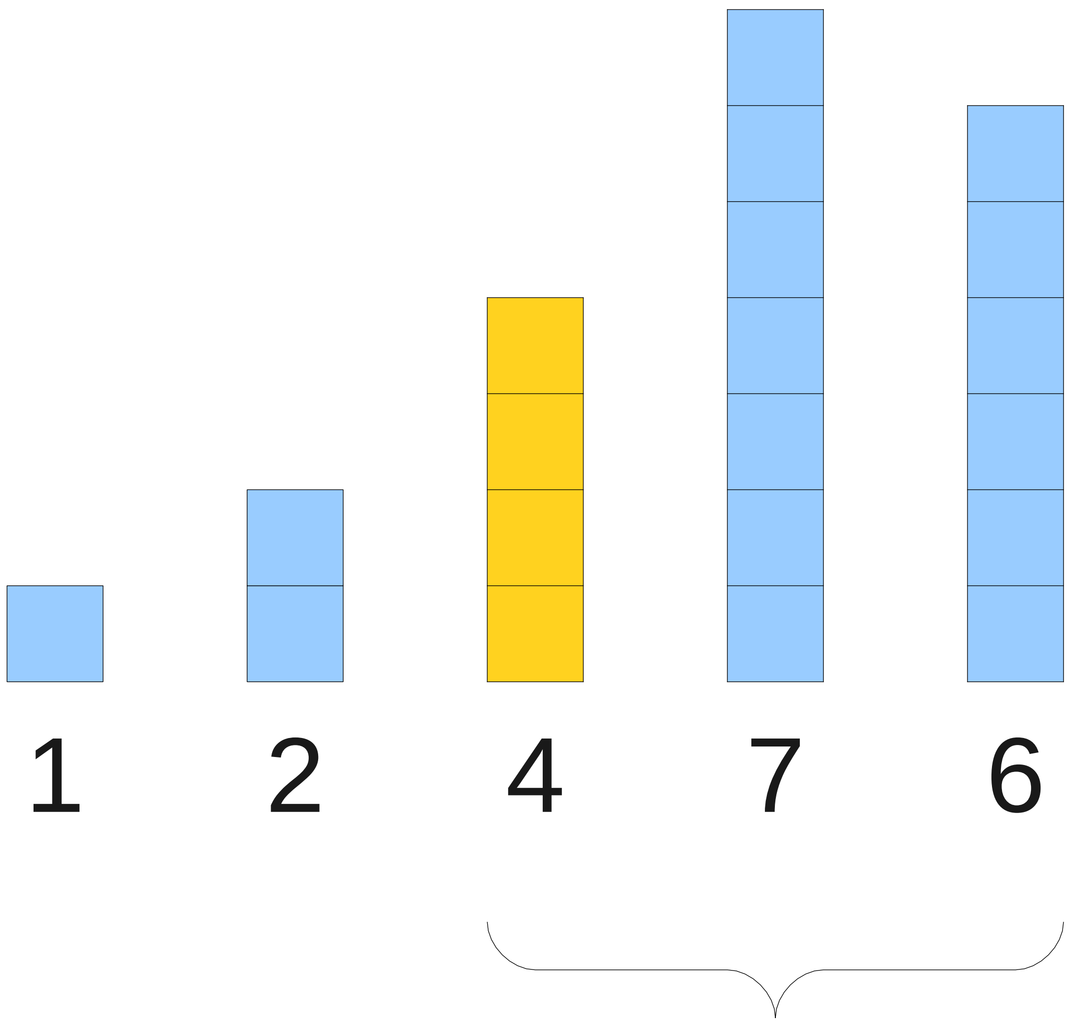
An Second Idea: **Selection Sort**



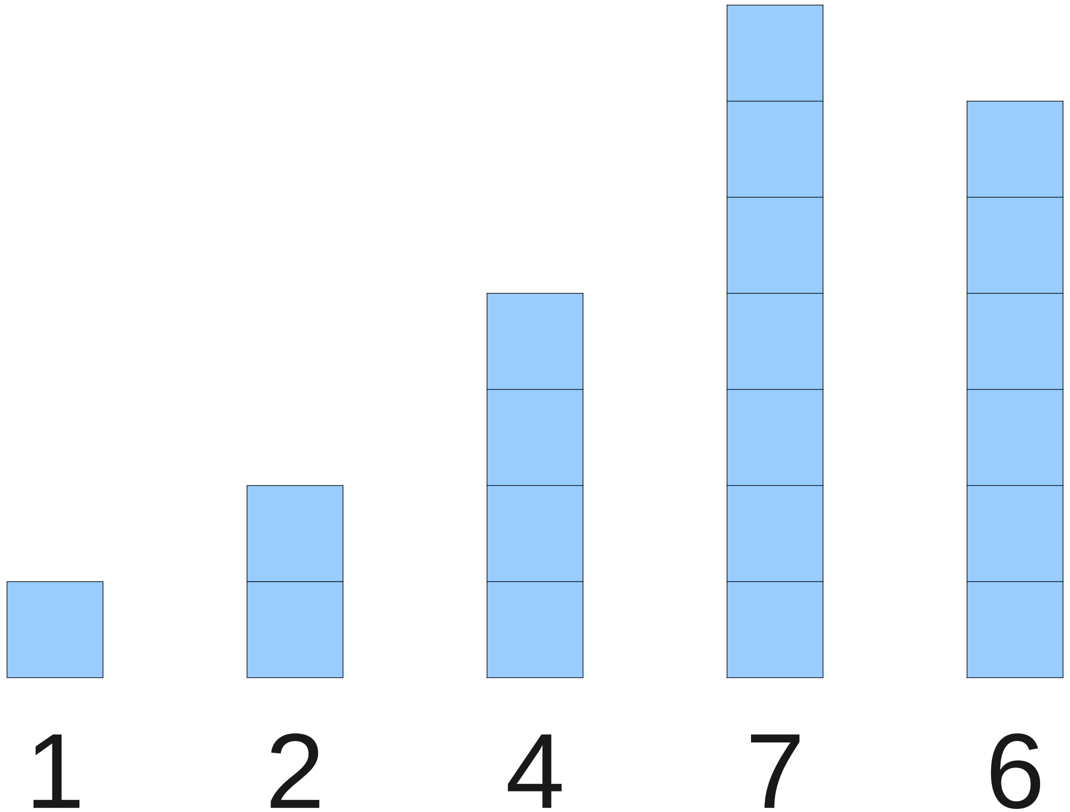
An Second Idea: Selection Sort



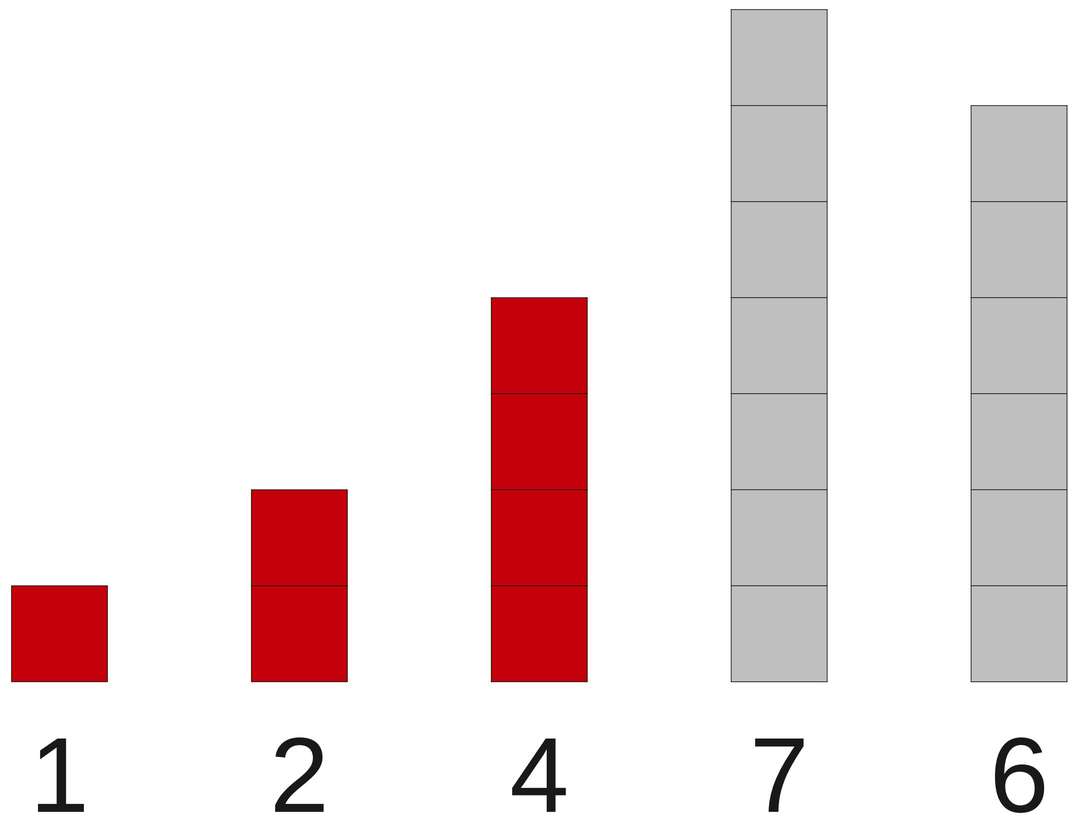
An Second Idea: Selection Sort



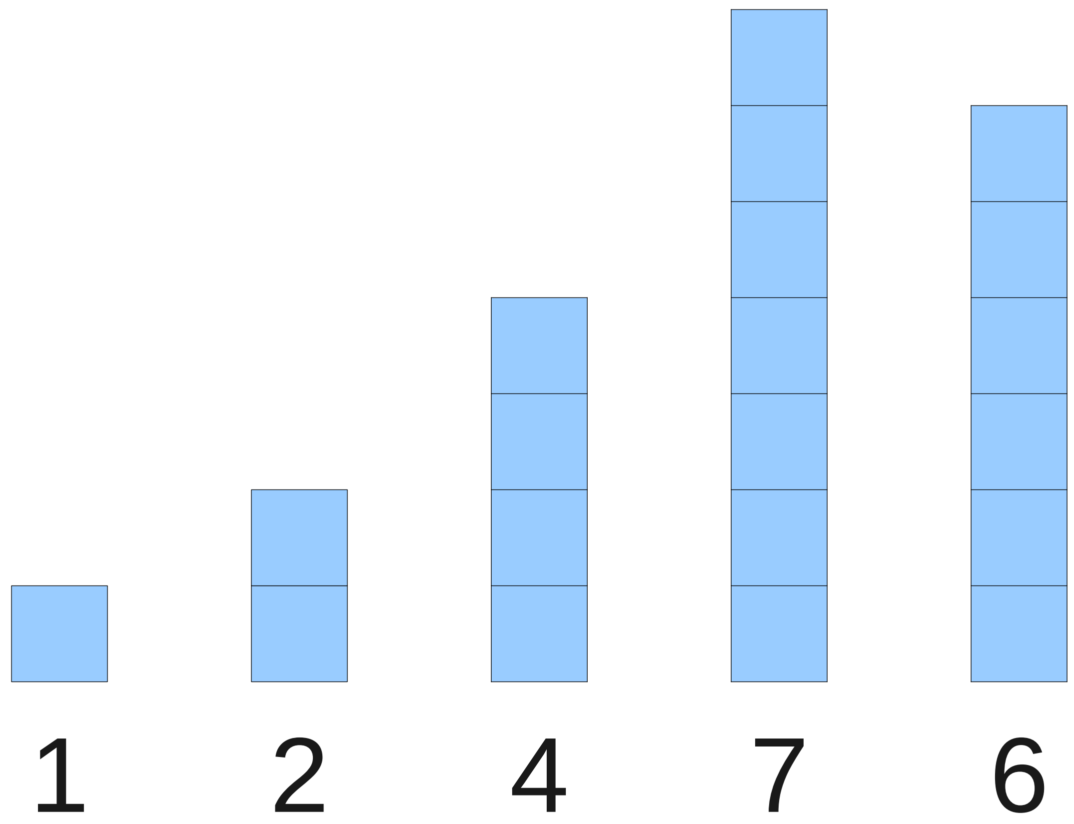
An Second Idea: Selection Sort



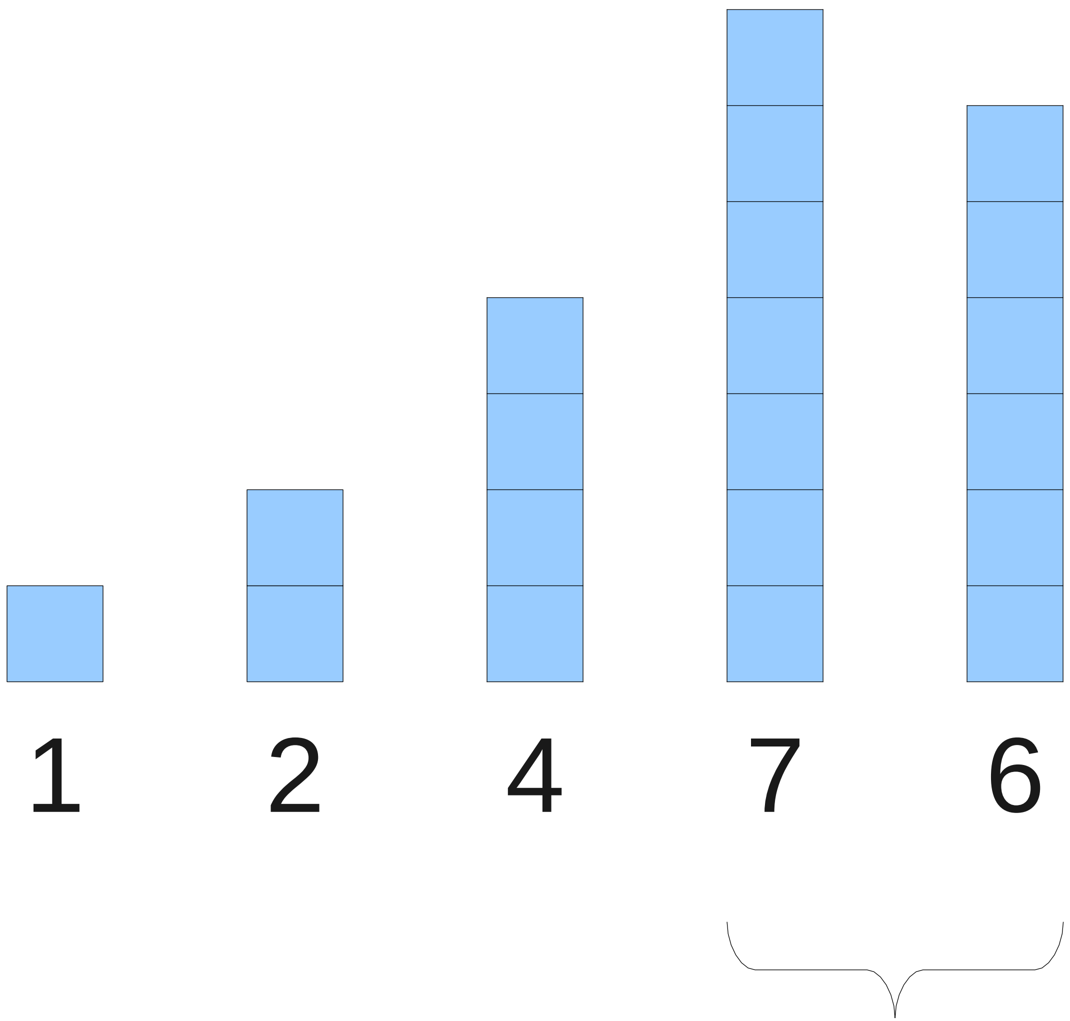
An Second Idea: Selection Sort



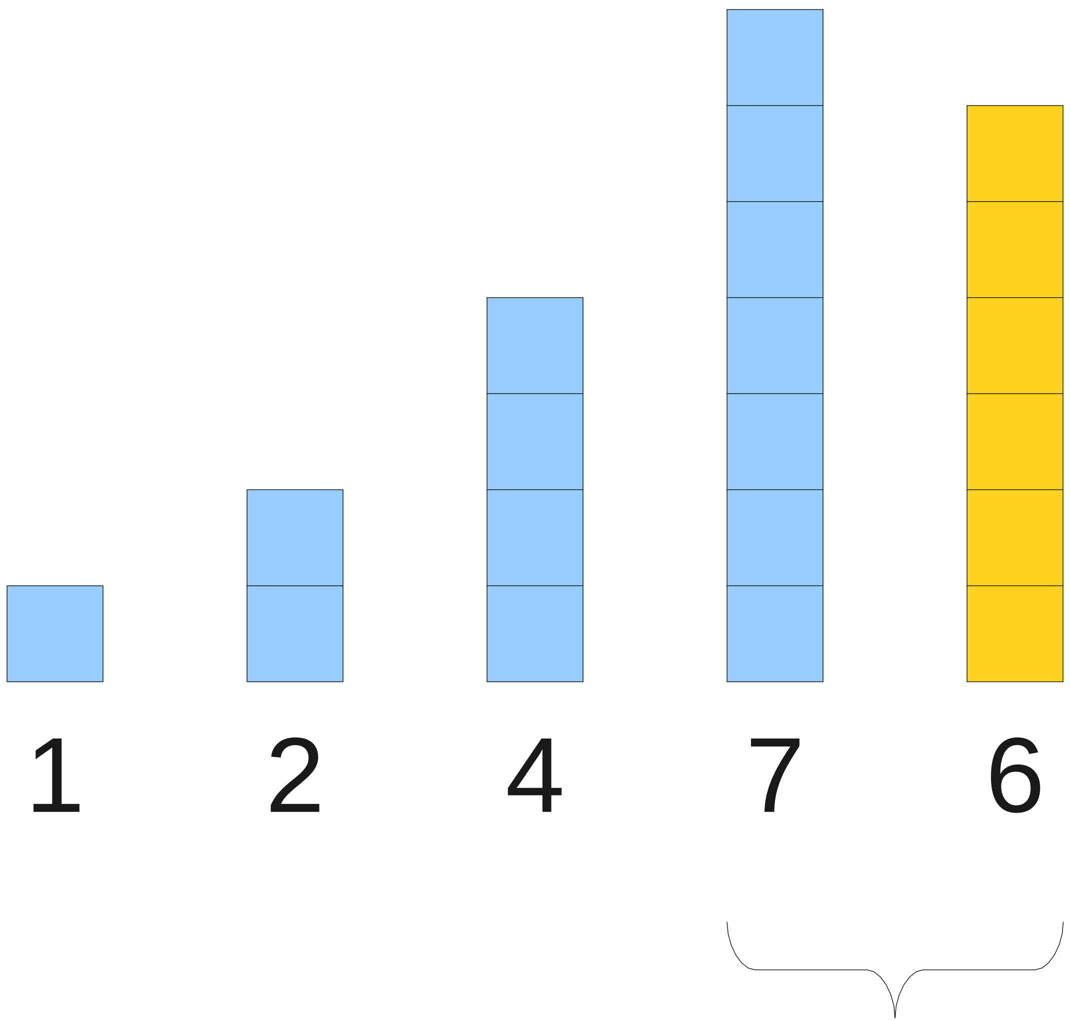
An Second Idea: Selection Sort



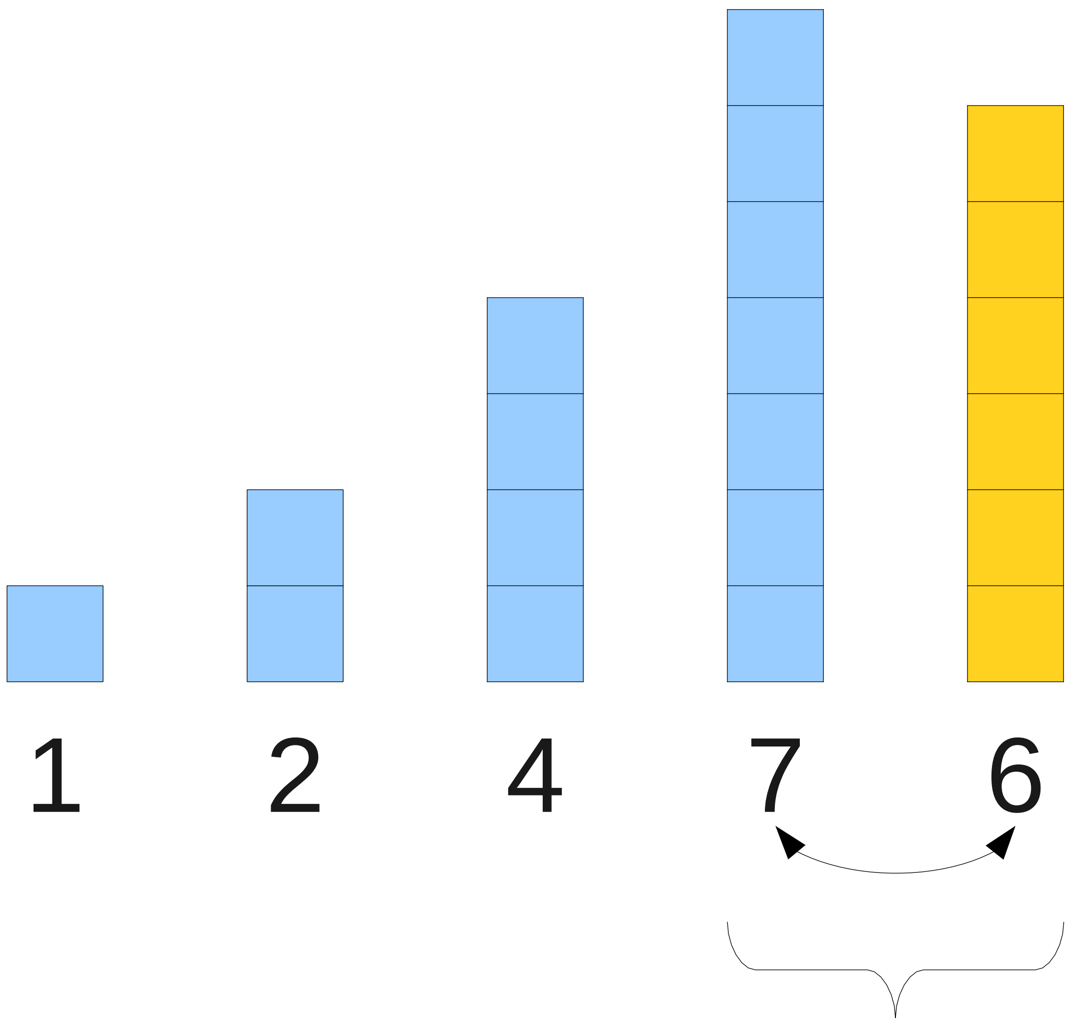
An Second Idea: Selection Sort



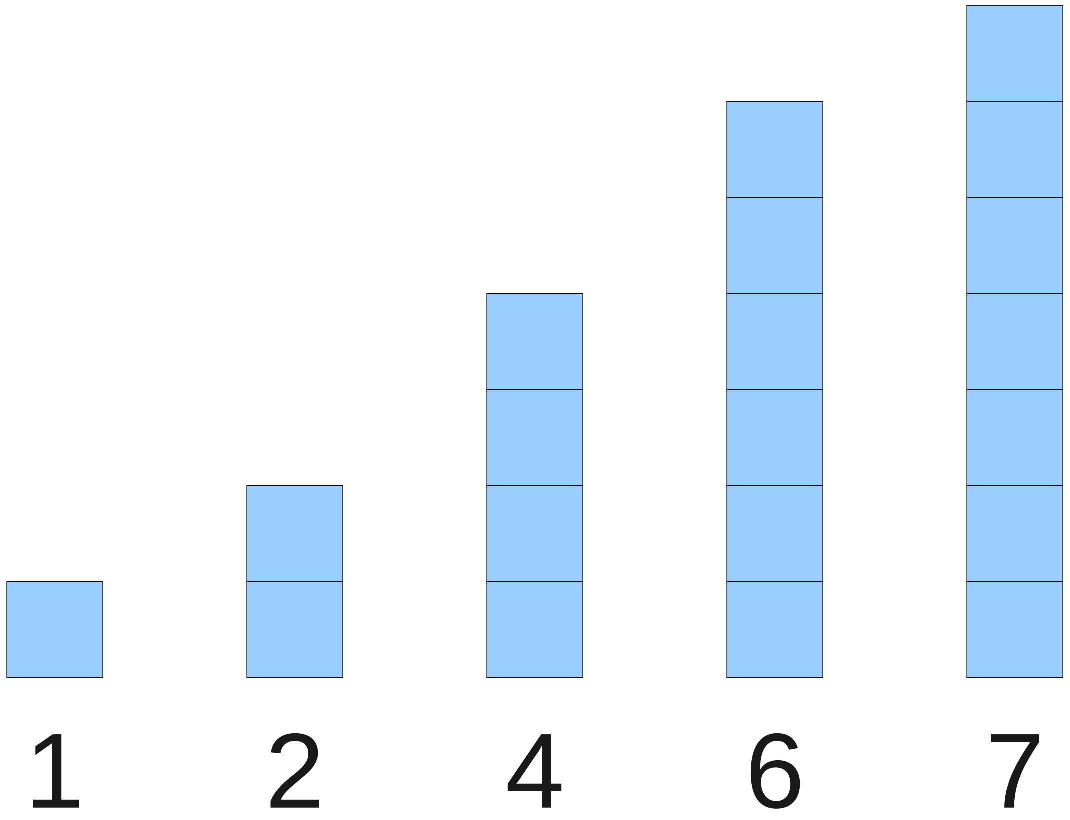
An Second Idea: Selection Sort



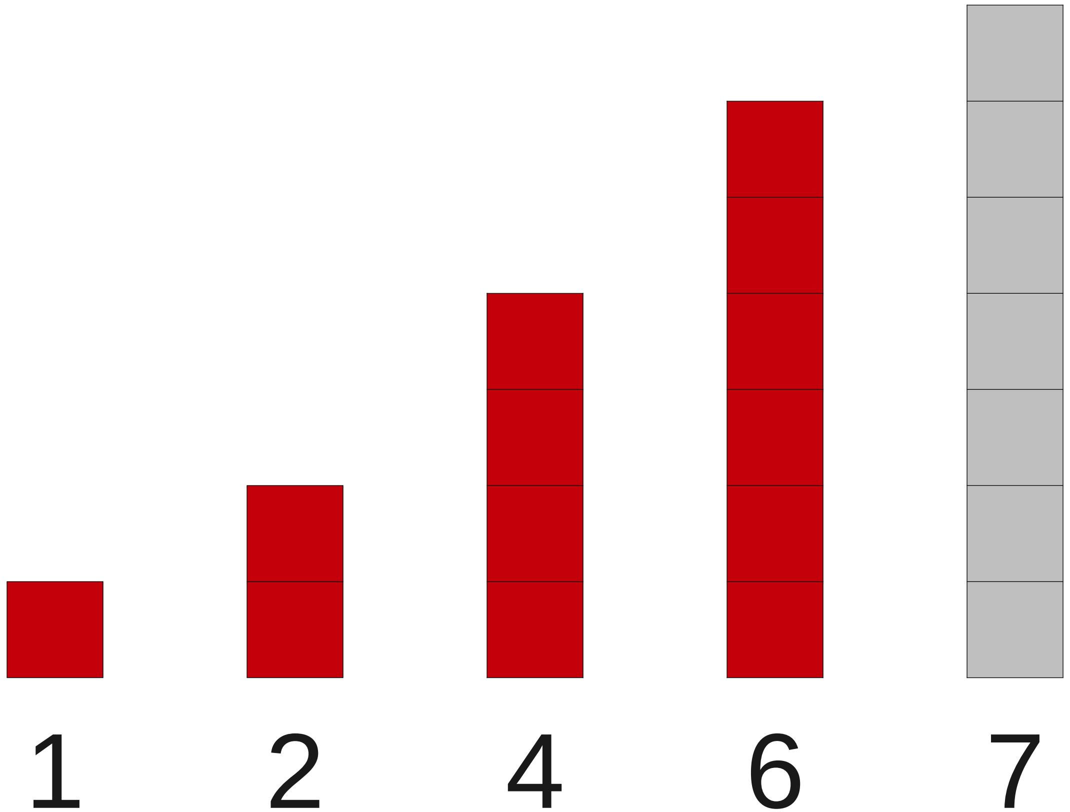
An Second Idea: Selection Sort



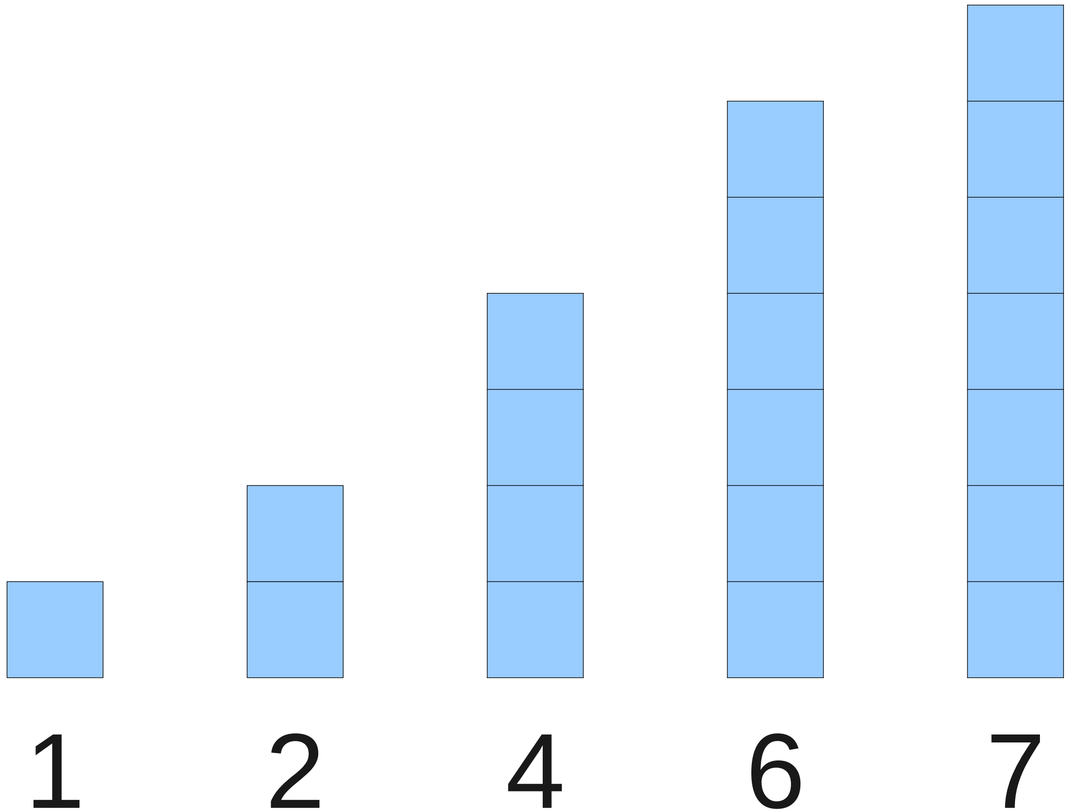
An Second Idea: **Selection Sort**



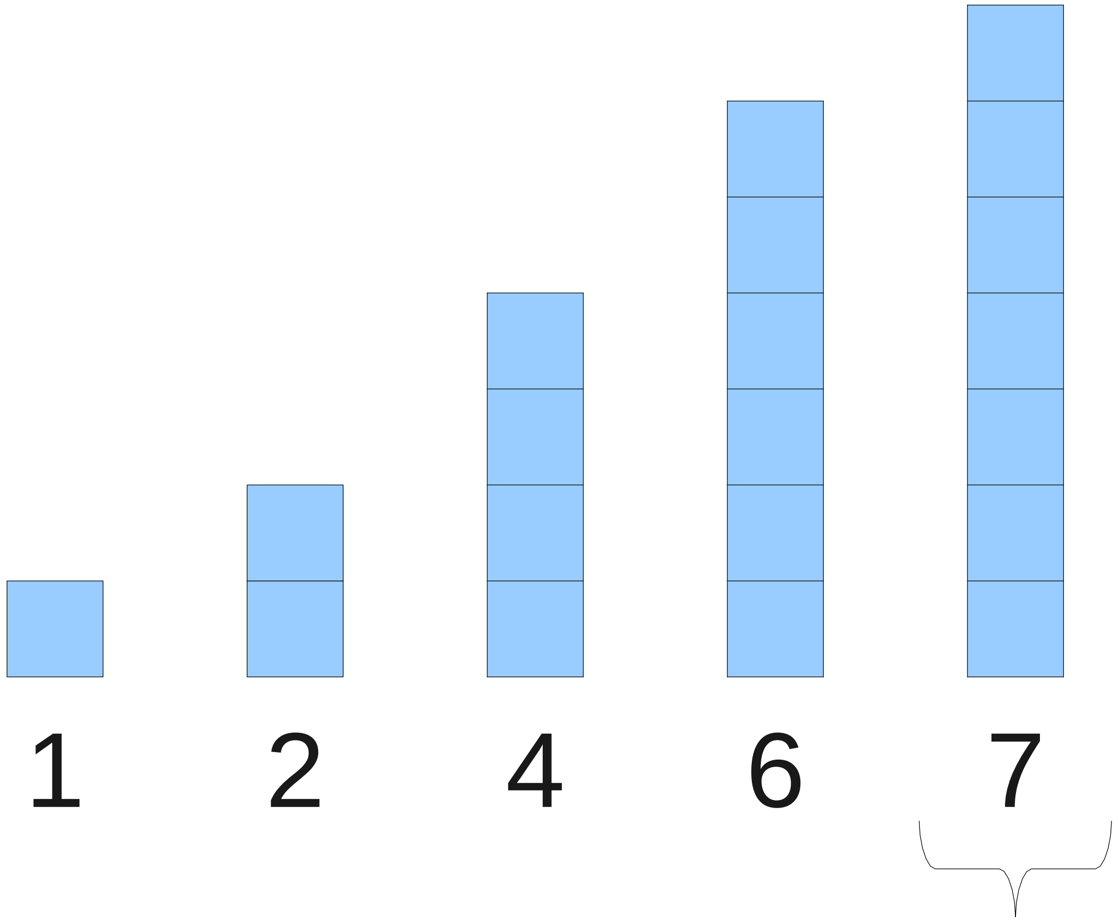
An Second Idea: **Selection Sort**



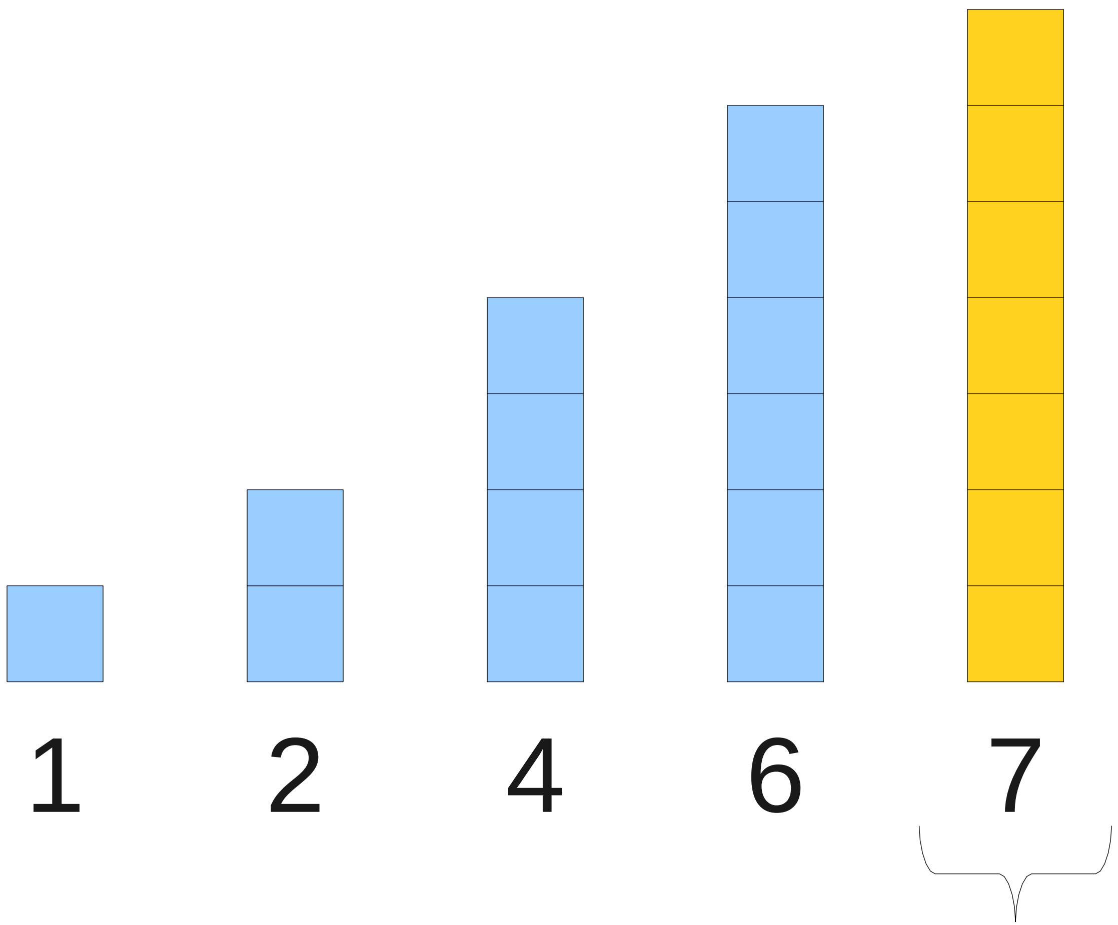
An Second Idea: Selection Sort



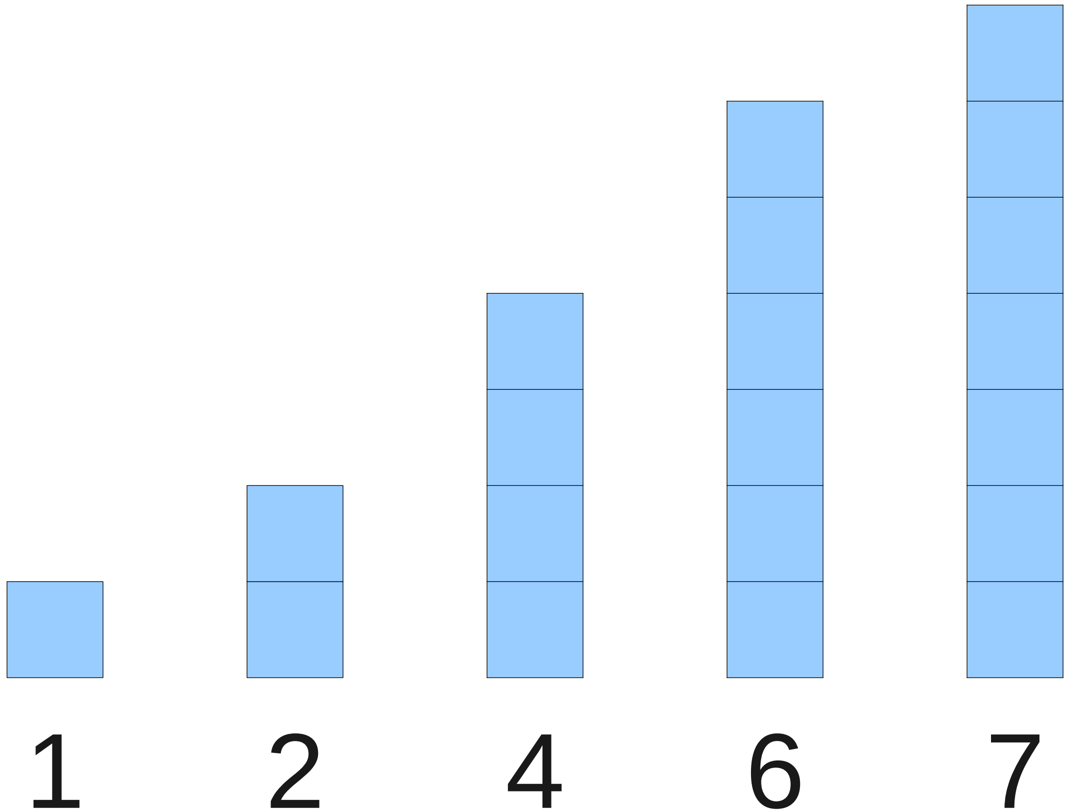
An Second Idea: **Selection Sort**



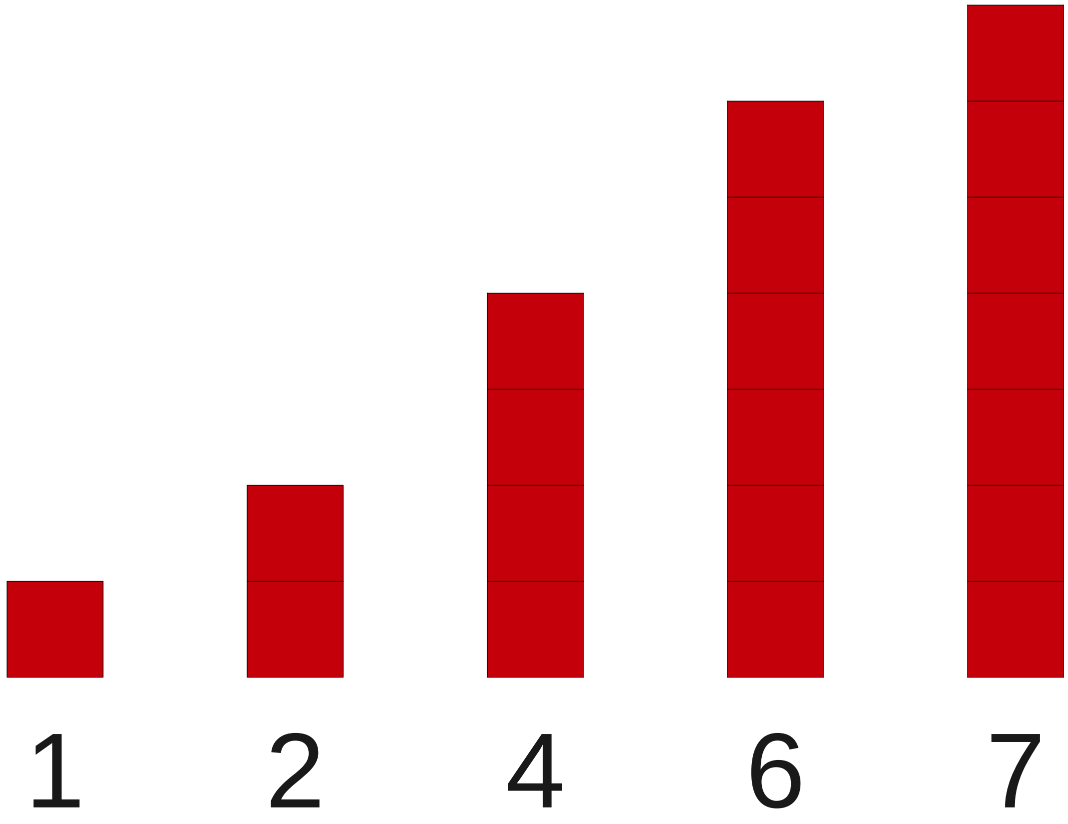
An Second Idea: Selection Sort



An Second Idea: **Selection Sort**



An Second Idea: Selection Sort



Selection Sort

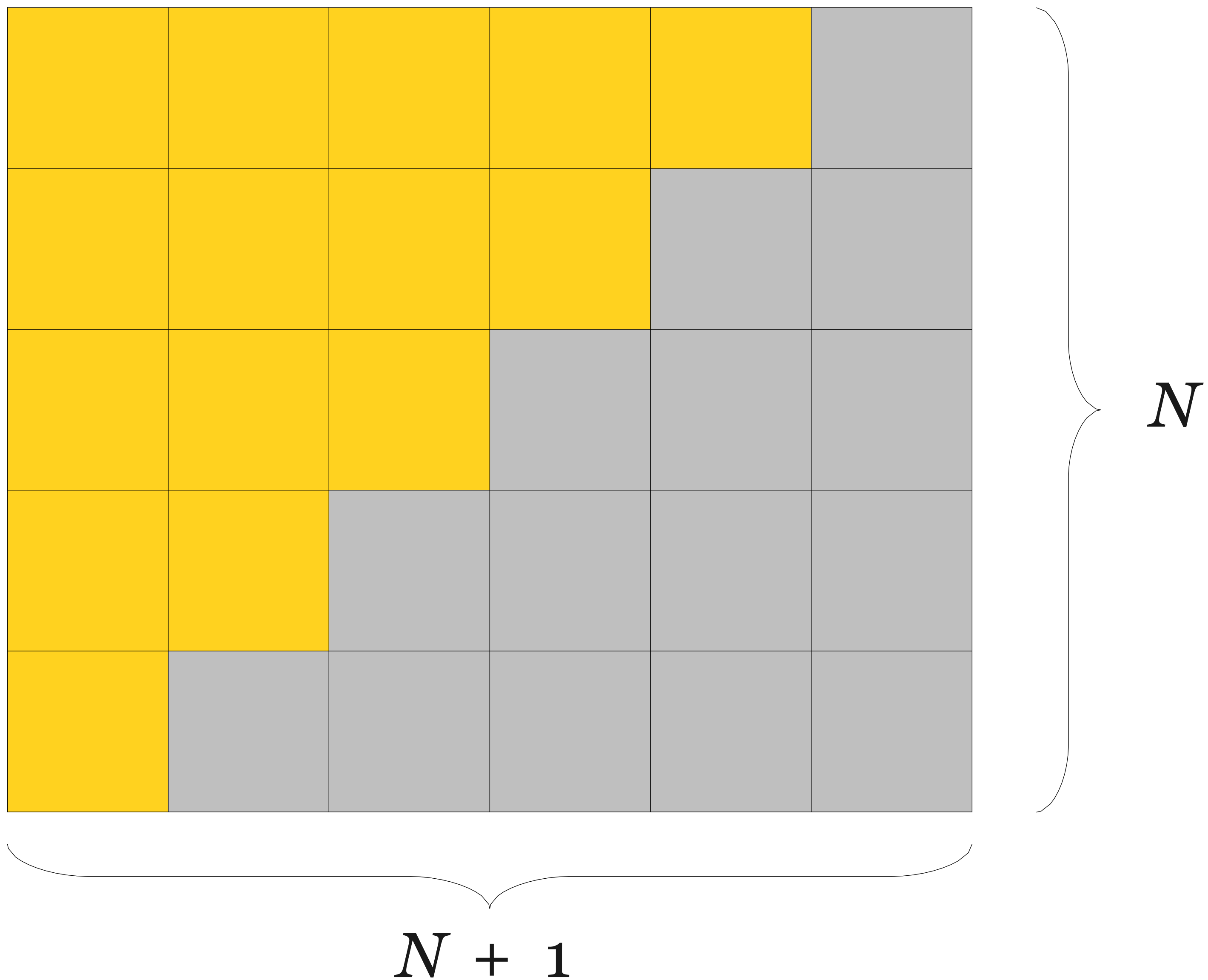
- Find the smallest element and move it to the first position.
- Find the second-smallest element and move it to the second position.
- (etc.)

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


private int indexOfSmallest(int[] elems, int startPoint) {
    int smallestIndex = startPoint;
    for (int i = startPoint + 1; i < elems.size(); i++) {
        if (elems[i] < elems[smallestIndex])
            smallestIndex = i;
    }
    return smallestIndex;
}

private void swap(int[] arr, int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}
```

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 $N + (N - 1) + (N - 2) + \ldots + 1$.

$$1 + 2 + \ldots + (N-1) + N = N(N+1)/2$$



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?

$$
\begin{aligned}
& \mathit{newTime} \,/\, \mathit{oldTime} \\
\approx\ & ((2N)^2 / 2) / (N^2 / 2) \\
\approx\ & (2N)^2 / N^2 \\
\approx\ & 4N^2 / N^2 \\
\approx\ & 4
\end{aligned}
$$

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?

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

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

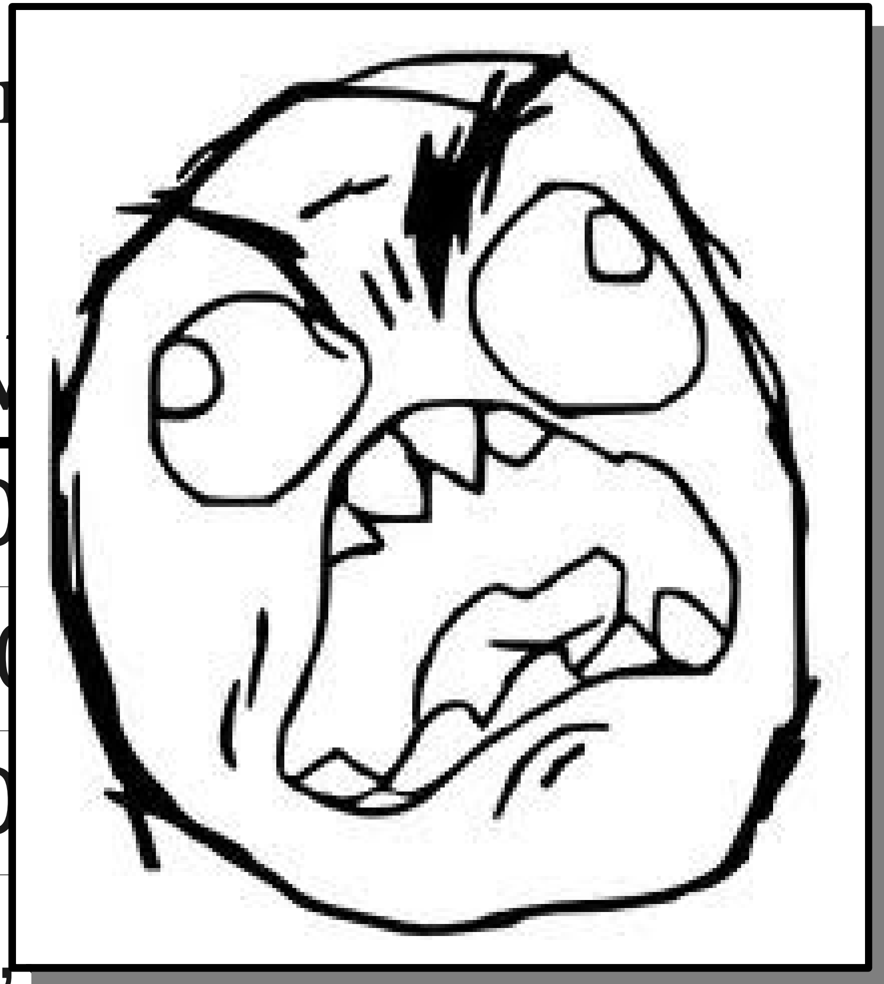
Analyzing Selection Sort

- 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 do



N	
10	
100	
100	
1,000,	,000
1,000,000,000	500,000,000,000,000,000