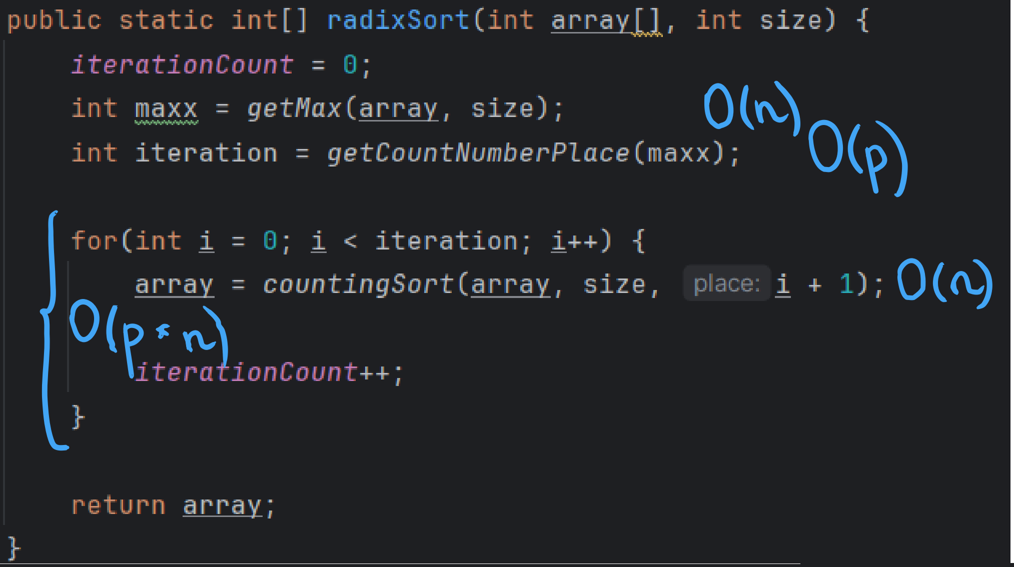


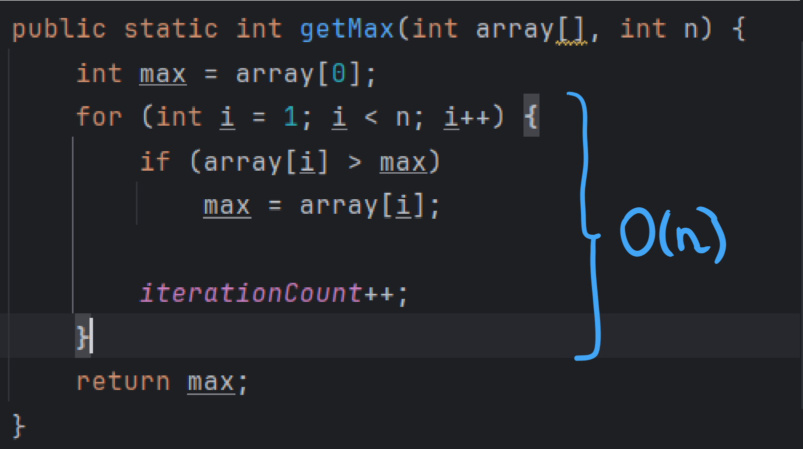
```
public static int[] radixSort(int array[], int size) {
    iterationCount = 0;
    int maxx = getMax(array, size);
    int iteration = getCountNumberPlace(maxx);

    for(int i = 0; i < iteration; i++) {
        array = countingSort(array, size, place: i + 1);

        iterationCount++;
    }

    return array;
}
```

$O(n)$ $O(p)$

$O(n)$

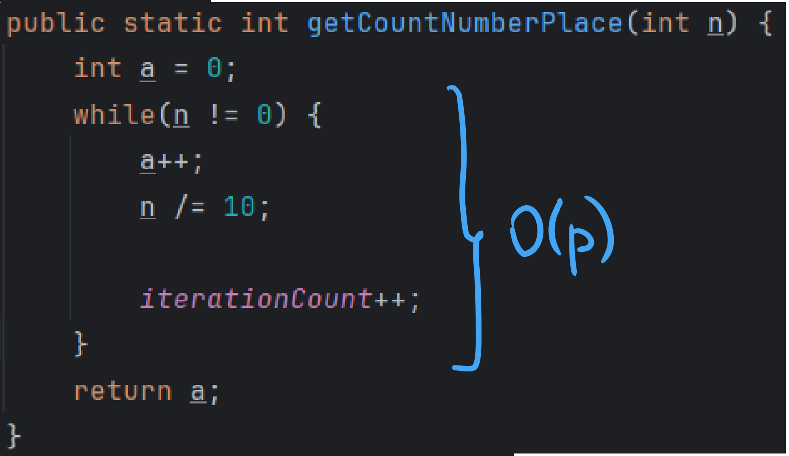
$O(p*n)$

n - кол-во элементов
p - кол-во бит самого длинного числа
Сложность по времени:
(в худшем случае)
$O(p*n)$
Сложность по памяти:
$O(p+n)$



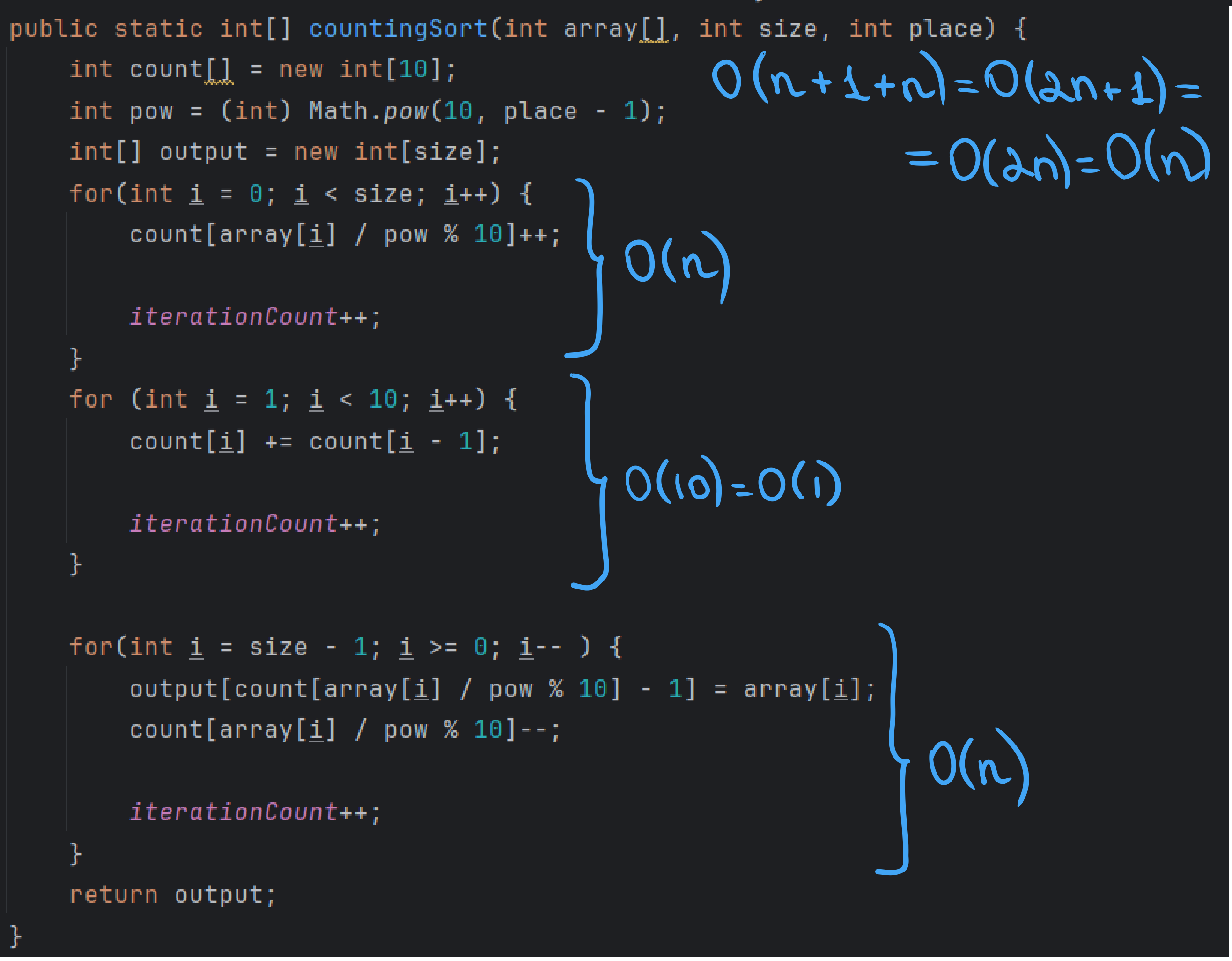
```
public static int getMax(int array[], int n) {
    int max = array[0];
    for (int i = 1; i < n; i++) {
        if (array[i] > max)
            max = array[i];

        iterationCount++;
    }
    return max;
}
```

$O(n)$



```
public static int getCountNumberPlace(int n) {
    int a = 0;
    while(n != 0) {
        a++;
        n /= 10;

        iterationCount++;
    }
    return a;
}
```

$O(p)$



```
public static int[] countingSort(int array[], int size, int place) {
    int count[] = new int[10];
    int pow = (int) Math.pow(10, place - 1);
    int[] output = new int[size];
    for(int i = 0; i < size; i++) {
        count[array[i] / pow % 10]++;

        iterationCount++;
    }
    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];

        iterationCount++;
    }

    for(int i = size - 1; i >= 0; i-- ) {
        output[count[array[i] / pow % 10] - 1] = array[i];
        count[array[i] / pow % 10]--;

        iterationCount++;
    }
    return output;
}
```

$O(n+1+n)=O(2n+1)=$
$=O(2n)=O(n)$

$O(n)$

$O(10)=O(1)$

$O(n)$