Определить сложность следующих алгоритмов:

-. Поиск элемента массива с известным индексом

O(1)  
-. Дублирование одномерного массива через foreach

O(n)  
-. Удаление элемента массива с известным индексом без сдвига

O(1)  
-. Удаление элемента массива с неизвестным индексом без сдвига

O(n)  
-. Удаление элемента массива с неизвестным индексом со сдвигом

O(n).

3. Определить сложность следующих алгоритмов. Сколько произойдет итераций?

a)

int n = 10000;

List<Integer> arrayList = new ArrayList<>();

for (int i = 0; i < n; i++) {

for (int j = 1; j < n; j \*= 2) {

arrayList.add(i \* j);

}

}

Ответ: O(n)\*O(log n)=O(n log n)

b)

int n = 10000;

List<Integer> arrayList = new ArrayList<>();

for (int i = 0; i < n; i += 2) {

for (int j = i; j < n; j++) {

arrayList.add(i \* j);

}

}

Ответ: O(n/2)\*O(n)=O(n^2/2)= O(n^2)

с)

int n = 10000;

List<Integer> arrayList = new ArrayList<>();

for (int i = 0; i < n; i ++) {

for (int j = 0; j < n; j++) {

arrayList.add(i \* j);

n--;

}

}

Ответ: O(n/2)+O(n/2) = O(n)

d)  
```

factorial(BigInteger.valueOf(10000))

public static BigInteger factorial(BigInteger n) {

if (n.equals(BigInteger.ONE)) {

return n;

}

return n.multiply(factorial(n.subtract(BigInteger.valueOf(1))));

}

Ответ:O(n)c памятью O(n)

e)

fib(BigInteger.valueOf(50));

public static BigInteger fib(BigInteger n) {

if (n.equals(BigInteger.ONE)) {

return BigInteger.ONE;

}

if (n.equals(BigInteger.TWO)) {

return BigInteger.ONE;

}

return fib(n.subtract(BigInteger.ONE)).add(fib(n.subtract(BigInteger.TWO)));

}

Ответ: O(1.7^n).