A4

1a) Если не задумываться о входных данных, то все три алгоритма ищут элемент, который занимает больше половины ячеек массива, то есть по сути решают одну и ту же задачу.

Результаты каждого алгоритма могут отличаться при разных входных данных.

1b)

```
algorithm1(A)
  c = 0
  ind = -1
  for i = 0 to n
      c1 = 0
      for j = 0 to n
          if A[i] = A[j]
              c1 = c1 + 1
      if c1 > c
          c = c1
          ind = i
  if c > n / 2
      return A[ind]
```

```
algorithm2(A)
  c = 1
  ind = 0
  for i = 1 to n
      if A[ind] = A[i]
          c = c + 1
      else
          c = c - 1
      if c = 0
          ind = i
          c = 1

  return A[ind]
```

```
algorithm3(A)
  if n = 1
      return A[0]
  c = 1
  sort(A)
  for i = 1 to n
      if A[i - 1] = A[i]
          c = c + 1
      else
          if c > n / 2
              return A[i - 1]
          c = 1
```

<u>Алгоритм №1</u>

A = [1 1 2]

Должен вернуть в виде ответа число 1.

c	ind	i	c1	j	A[i]	A[j]	c1	c	ind	A[ind]
0	-1	0	0	0	1	1	1	1	0	1
1	0	0	1	1	1	1	2	2	0	1
2	0	0	2	2	1	2	2	2	0	1
2	0	1	...	...	...	...	...	...	...	...

Далее аналогично можно продолжить, но все равно будет возвращено A[0] = 1.

<u>Алгоритм №2</u>

A = [1 1 2]

Должен вернуть в виде ответа число 2.

c	ind	i	A[ind]	A[i]	c	ind	c	A[ind]
1	0	1	1	1	2	0	2	1
2	0	2	1	2	1	0	1	1

Вывод: 1

Алгоритм №3

A = [1 1 2]

Должен вернуть в виде ответа число 2.

c	i	A[i - 1]	A[i]	c	A[i - 1]	c
1	1	1	1	2	1	2
2	2	1	2	2	1	1

Вывод: 1

Все алгоритмы работают верно при A = [1 1 2].

Пусть A = [1], тогда 1 и 3 алгоритмы сработают, а 2 алгоритм нет.

1 и 3 алгоритмы сработают при A = [1], а 2 алгоритм нет.

Пусть A = [1 2 2]

Алгоритмы должны вернуть 2

Алгоритм №1

A = [1 2 2]

c	ind	i	c1	j	A[i]	A[j]	c1	c	ind	A[ind]
0	-1	0	0	0	1	1	1	1	0	1
1	0	0	1	1	1	2	1	1	0	1
аналогично										
1	0	1	0	0	2	1	0	1	0	1
1	0	1	0	1	2	2	1	1	1	2
1	1	1	1	2	2	2	2	2	1	2
аналогично										

Далее будет выведено 2 => алгоритм работает верно.

Алгоритм №2

A = [1 2 2]

c	ind	i	A[ind]	A[i]	c	ind	c	A[ind]
1	0	1	1	2	0	1	1	2
1	1	2	2	2	2	1	2	2

Далее будет выведено 2 => алгоритм работает верно.

Алгоритм №3

A = [1 2 2]

c	i	A[i - 1]	A[i]	c	A[i - 1]
1	1	1	2	1	1

Далее будет выведено 1 => алгоритм работает не верно.

1 и 2 алгоритмы сработают при A = [1 2 2], а 3 алгоритм нет.

ОТВЕТ: Пример №1 с входными данными A = [2 2 1] работает со всеми алгоритмами. Пример №2 с входными данными A = [1] не работает с 2 алгоритмом. Пример №3 с входными данными A = [1 2 2] не работает с 3 алгоритмом. Данный ответ доказывают трассировки выше!

2. Вычисление асимптотической верхней границы

$$O(g(n)) = \{f(n) : \text{существуют положительные константы } c \text{ и } n_0, \text{ такие, что } 0 \le f(n) \le cg(n) \text{ для всех } n \ge n_0\} .$$

Определим O(f(n))

```
algorithm1(A)
1   c = 0                                   O(1)
2   ind = -1
3   for i = 0 to n                          O(n^2) – внешний цикл
4       c1 = 0                              O(1)
5       for j = 0 to n                      O(n) – внутренний цикл
6           if A[i] = A[j]
7               c1 = c1 + 1
8       if c1 > c
9           c = c1                          O(1)
10          ind = i
11  if c > n / 2
12      return A[ind]
```

T(n) = O(n²) для алгоритма №1

продолжение на следующей странице ->

```
algorithm2(A)
  c = 1                          O(1)
  ind = 0
  for i = 1 to n                 O(n) - цикл
      if A[ind] = A[i]
          c = c + 1
      else
          c = c - 1              O(1)
      if c = 0
          ind = i
          c = 1

  return A[ind]
```

T(n) = O(n) для алгоритма №2

```
algorithm3(A)
  if n = 1                       O(1)
      return A[0]
  c = 1
  sort(A)                        O(n^2)
  for i = 1 to n                 O(n) - цикл
      if A[i - 1] = A[i]
          c = c + 1
      else                       O(1)
          if c > n / 2
              return A[i - 1]
          c = 1
```

$O(n^2)$ + O(n) => T(n) = $O(n^2)$ для алгоритма №3

3.

Добавить в алгоритм №2 проверку на массив с одним элементом:

if n = 1

return A[0]

Добавить в алгоритм №3 к условию на строчке 9: **if c > n/2 and i >= n / 2**

4. Доработки не ухудшают верхние границы, так как их верхние границы равны O(1).