Двоичный поиск: границы и позиции вставки

1. Для bisect_left([1,3,3,7],3) результат какой?

1. 0
2. 1
3. 2
4. 3

Правильный ответ: 1

2. Для bisect_right([1,3,3,7],3) результат какой?

1. 1
2. 2
3. 3
4. 4

Правильный ответ: 3

3. Для bisect_left([2,5,9],1) результат какой?

1. 0
2. 3
3. −1
4. 1

Правильный ответ: 0

4. Для bisect_left([2,5,9],10) результат какой?

1. 10
2. 3
3. 2
4. −1

Правильный ответ: 3

5. Какое предусловие нужно обычному bisect для корректной позиции?

1. Список отсортирован по тому же правилу сравнения
2. Длина обязательно степень двойки
3. Все элементы обязательно уникальны
4. Список содержит только строки

Правильный ответ: Список отсортирован по тому же правилу сравнения

6. После insort([1,4,8],5) каков порядок значений в изменяемом списке?

1. 5,1,4,8
2. 1,5,4,8
3. 1,4,8,5
4. 1,4,5,8

Правильный ответ: 1,4,5,8

7. Вернул bisect_left(a,x) позицию i. Этого достаточно, чтобы утверждать наличие x?

1. Да, если i положительно
2. Нет, нужно проверить i<len(a) и a[i]==x
3. Да, любая позиция означает совпадение
4. Нет, bisect всегда возвращает None

Правильный ответ: Нет, нужно проверить i<len(a) и a[i]==x

8. Сколько раз примерно нужно делить пополам диапазон из 1024 элементов, чтобы оставить один?

1. 1024
2. 10
3. 512
4. 32

Правильный ответ: 10

9. Почему полная insort в списке имеет линейную стоимость, несмотря на двоичный поиск?

1. Потому что все числа переводятся в строки
2. Список автоматически копируется в словарь
3. Поиск всегда просматривает все пары
4. Элементы после позиции нужно сдвинуть

Правильный ответ: Элементы после позиции нужно сдвинуть

10. Число вхождений x в отсортированный список можно получить как?

1. bisect_left(a,x)−bisect_right(a,x)
2. bisect_right(a,x)+bisect_left(a,x)
3. bisect_right(a,x)−bisect_left(a,x)
4. len(a) всегда

Правильный ответ: bisect_right(a,x)−bisect_left(a,x)