1 Сортировки

1.1. Даны два отсортированных по неубыванию массива a и b. Определите, есть ли в них одинаковые числа. Время $O(n)$.

1.2. Даны два отсортированных по неубыванию массива a и b. Найдите такие i и j, что разница $|a_i - b_j|$ минимальна. Время $O(n)$.

1.3. Даны два отсортированных по неубыванию массива a и b и число S. Найдите такие i и j, что сумма $a_i + b_j = S$. Время $O(n)$.

1.4. Даны два отсортированных по неубыванию массива a и b. Найдите число пар (i, j), таких, что $a_i = b_j$. Время $O(n)$.

1.5. Даны два отсортированных по неубыванию массива a и b. Найдите число пар (i, j), таких, что $a_i > b_j$. Время $O(n)$.

1.6. Дан массива a. Пара (i, j), такая, что $i < j$ и $a_i > a_j$ называется *инверсией*. Пусть в массиве длины n ровно k инверсий. Докажите, что сортировка вставками работает за $O(n + k)$.

1.7. Дан массива a. Найдите число инверсий в нем. Время $O(n \log n)$

1.8. Покажите, что сортировка слиянием является устойчивой (то есть, не меняет порядок равных элементов).

1.9. Покажите, как сделать сортировку слиянием снизу вверх, без рекурсии.

2 Рекуррентные соотношения

2.1. Докажите по индукции, что если $T(n) = 2T(n/2 + 20) + n$, то $T(n) = O(n \log n)$.

2.2. Докажите по индукции, что если $T(n) = 2T(n/2 + \log n) + n$, то $T(n) = O(n \log n)$.

2.3. Докажите по индукции, что если $T(n) = \log n \cdot T(n/\log n) + n$, то $T(n) = O(n \log n)$.

2.4. Докажите по индукции, что если $T(n) = 2T(n/2) + n$, то $T(n) = \Omega(n \log n)$ (оценка снизу).

2.5. Докажите по индукции, что если $T(n) = 2T(\sqrt{n}) + 1$, то $T(n) = O(\log n)$.

2.6. Найдите асимптотическую оценку на $T(n)$, если $T(n) = T(n - a) + T(a) + n$ (a — константа).

2.7. Для каждой из приведенных программ и функций оцените время ее работы

a)

```
for i in range(n):
  j = 0
  while j * j < i:
    j += 1
```

b)

```
for i in range(n):
  j = i
  while j > 0:
    j = j // 2
```

c)

```
def f(n):
  if n == 0:
    return 1
  else
    return 5 * f(n // 3)
```

d)

```
def f(n):
  if n == 0:
    return 1
  else
    return f(n // 3) + f(n // 3)
```