Задача A1. Временная сложность рекурсии

Ниже приведены два рекурсивных алгоритма обработки целочисленного массива A размера n:

```
algorithm1(A, n)
    if n <= 20
        return A[n]
    x = algorithm1(A, n - 5)

    for i = 1 to ⌊n / 2⌋
        for j = 1 to ⌊n / 2⌋
            A[i]= A[i] - A[j]
    x = x + algorithm1(A, n - 8)

    return x
```

```
algorithm2(A, n):
    if n <= 50
        return A[n]
    x = algorithm2(A, ⌊n / 4⌋)

    for i = 1 to ⌊n / 3⌋
      A[i] = A[n - i] - A[i]

    x = x + algorithm2(A, ⌊n / 4⌋)

    return x
```

1. Для каждого из представленных алгоритмов составить рекуррентное соотношение, которое выражает их временную сложность T(n). Обратите внимание, что рекуррентное соотношение должно давать полное представление о сложности алгоритма, т.е., охватывать как рекурсивную, так и нерекурсивную ветку вычислений. Предполагается, что все арифметические операции выполняются за постоянное время.

Алгоритм 1

Подробно разберём временную сложность алгоритма.

Базовый случай:
Если $n \leq 20$, алгоритм просто возвращает $A[n]$. Это выполняется за константное время:

$$T(n) = O(1), \quad \text{при } n \leq 20$$

Рекурсивный случай:
Если $n > 20$:
1. Рекурсивный вызов: $x = \text{algorithm1}(A, n - 5)$
Время: $T(n - 5)$

2. Вложенные циклы:

```
for i = 1 to ⌊n / 2⌋
    for j = 1 to ⌊n / 2⌋
        A[i]= A[i] - A[j]
```

Количество итераций внешнего цикла: $\left\lfloor \frac{n}{2} \right\rfloor \approx \frac{n}{2}$
Количество итераций внутреннего цикла: $\left\lfloor \frac{n}{2} \right\rfloor \approx \frac{n}{2}$
Общее количество операций: $\left(\frac{n}{2}\right) \times \left(\frac{n}{2}\right) = \frac{n^2}{4}$
Таким образом, время на выполнение циклов: $O(n^2)$

3. Второй рекурсивный вызов: $x = x + \text{algorithm1}(A, n - 8)$
Время: $T(n - 8)$

Итог: Время для $n > 20$ составляет сумму времен рекурсивных вызовов и времени на выполнение циклов:

$$T(n) = T(n - 5) + T(n - 8) + O(n^2)$$

Рекуррентное соотношение для алгоритма 1:

$$T(n) = \begin{cases} O(1), & \text{если } n \leq 20 \\ T(n - 5) + T(n - 8) + O(n^2), & \text{если } n > 20 \end{cases}$$

Алгоритм 2

Подробно разберём временную сложность алгоритма.

Базовый случай:
Если $n \leq 50$, алгоритм возвращает $A[n]$. Это выполняется за константное время.

$$T(n) = O(1), \quad \text{при } n \leq 50$$

Рекурсивный случай:
Если $n > 50$:
1. Первый рекурсивный вызов: $x = \text{algorithm2}(A, \left\lfloor \frac{n}{4} \right\rfloor)$
Время: $T\left(\frac{n}{4}\right)$

2. Цикл:

```
for i = 1 to ⌊n / 3⌋
  A[i] = A[n - i] - A[i]
```

Количество итераций: $\left\lfloor \frac{n}{3} \right\rfloor \approx \frac{n}{3}$

Время на выполнение цикла: $O(n)$

3. Второй рекурсивный вызов: $x = x + \text{algorithm2}(A, \left\lfloor \frac{n}{4} \right\rfloor)$

Время: $T\left(\frac{n}{4}\right)$

Итог: Время для $n > 50$ составляет:

$$T(n) = 2 \times T\left(\frac{n}{4}\right) + O(n)$$

Рекуррентное соотношение для алгоритма 2:

$$T(n) = \begin{cases} O(1), & \text{если } n \leq 50 \\ 2 \times T\left(\frac{n}{4}\right) + O(n), & \text{если } n > 50 \end{cases}$$

2. Вычислите асимптотическую точную границу Θ(f(n)) временной сложности для каждого из представленных алгоритмов, если это возможно. В случае невозможности формирования асимптотической точной границы, представить отдельно верхнюю и нижнюю границы. Обоснуйте свой ответ с помощью метода подстановки, дерева рекурсии, или индукции.

Алгоритм 1

Найти асимптотическую границу для $T(n)$, заданного рекуррентным соотношением:

$$T(n) = T(n-5) + T(n-8) + O(n^2)$$

Анализ асимптотической временной сложности

Структура рекурсии.
При каждом вызове для $n > 20$ происходит два рекурсивных вызова:
algorithm1(A, n - 5)
algorithm1(A, n - 8)

Таким образом, каждый уровень рекурсии удваивает количество вызовов, формируя бинарное дерево рекурсии.

Глубина рекурсии.
Каждый рекурсивный вызов уменьшает n на константу (5 или 8).

Максимальная глубина рекурсии пропорциональна n, то есть $O(n)$.

Количество узлов в рекурсивном дереве.

На каждом уровне количество узлов удваивается.

Общее количество узлов составляет $O(2^{n/5})$ (учитывая минимальное уменьшение на 5).

Нерекурсивная часть.

В каждом вызове выполняется двойной цикл с временной сложностью $\Theta(n^2)$.

Общая временная сложность.

Суммарная временная сложность определяется количеством узлов в рекурсивном дереве и временем работы на каждом узле:

$$T(n) = O\left(2^{n/5} \cdot n^2\right),$$

что приводит к экспоненциальной временной сложности:

$$T(n) = \Theta(2^n)$$

Алгоритм 2

Найти асимптотическую границу для $T(n)$, заданного рекуррентным соотношением:

$$T(n) = 2 \times T\left(\frac{n}{4}\right) + O(n)$$

Дерево рекурсии

Уровень 0: $T(n)$ с затратами $O(n)$.
Уровень 1: $2 \times T\left(\frac{n}{4}\right)$ с общими затратами $2 \times O\left(\frac{n}{4}\right) = O\left(\frac{n}{2}\right)$.
Уровень 2: $4 \times T\left(\frac{n}{4^2}\right)$ с общими затратами $4 \times O\left(\frac{n}{16}\right) = O\left(\frac{n}{4}\right)$.

.
.
.

Уровень k: $2^k \times T\left(\frac{n}{4^k}\right)$ с затратами $O\left(n \times \left(\frac{1}{2}\right)^k\right)$.

Высота дерева

$$\log_4 n$$

Общая сумма затрат по уровням

$$O(n) + O\left(\frac{n}{2}\right) + O\left(\frac{n}{4}\right) + \cdots + O\left(n \times \left(\frac{1}{2}\right)^{\log_4 n}\right)$$

Это геометрическая прогрессия с суммой $O(n)$.

Подстановка

Покажем, что $T(n) = O(n)$ и $T(n) = \Omega(n)$.

Верхняя граница $O(n)$.

Предположим, что $T(n) \leq cn$ для некоторой константы c.

$$T(n) \leq 2 \times \frac{cn}{4} + kn = \frac{cn}{2} + kn$$

Для того чтобы $\frac{c}{2} + k \leq c$, достаточно выбрать $c \geq 2k$.

Нижняя граница $\Omega(n)$.

Предположим, что $T(n) \geq dn$ для некоторой константы d.

$$T(n) \geq 2 \times \frac{dn}{4} + kn = \frac{dn}{2} + kn$$

Для того чтобы $\frac{d}{2} + k \geq d$, достаточно выбрать $d \leq 2k$.

$\Longrightarrow$

Таким образом, асимптотическая точная граница временной сложности: $T(n) = \Theta(n)$.