Алгоритмы и структуры данных-1

SET 2. Задача А1.

Осень 2024. Клычков М. Д.

```
1   algorithm1(A, n)
2       if n <= 20
3           return A[n]
4       x = algorithm1(A, n - 5)
5
6       for i = 1 to ⌊n / 2⌋
7           for j = 1 to ⌊n / 2⌋
8               A[i]= A[i] - A[j]
9       x = x + algorithm1(A, n - 8)
10
11      return x
```

```
1   algorithm2(A, n):
2       if n <= 50
3           return A[n]
4       x = algorithm2(A, ⌊n / 4⌋)
5
6       for i = 1 to ⌊n / 3⌋
7         A[i] = A[n - i] - A[i]
8
9       x = x + algorithm2(A, ⌊n / 4⌋)
10
11      return x
```

Пункт 1. Проанализируем первый алгоритм. В рекурсивную ветку вычислений входят вызовы функции `algorithm1` на 4 и 9 строках кода, причем аргументы передаются разные. В нерекурсивной ветке вычислений, учитывая, что все арифметические операции выполняются за константное время, производится $\Theta\left(\left\lfloor\frac{n}{2}\right\rfloor^2\right) + \Theta(1) = \Theta\left(\left(\frac{n}{2}\right)^2\right) = \Theta(n^2)$ операций. Такое количество обеспечивается вложенным циклом от 1 до $\left\lfloor\frac{n}{2}\right\rfloor$ каждый. Также необходимо учесть, что при $n \leq 20$ алгоритм будет работать за $\Theta(1)$.

Можно еще заметить, что на каждом шаге рекурсии происходит копирование массива A. Не понятно, является ли это одним из допущений псевдокода или задуманным копированием, но в любом случае нерекурсивная ветка даже с учетом копирования: $\Theta(n^2) + \Theta(n) = \Theta(n^2)$

$$T_1(n) = \begin{cases} \Theta(1) & , n \leq 20 \\ T_1(n-5) + T_1(n-8) + \Theta(n^2) & , n > 20 \end{cases} \tag{1}$$

Проанализируем второй алгоритм. В рекурсивную ветку вычислений входят вызовы функции `algorithm2` на 4 и 9 строках кода, аргументы передаются одинаковые. В нерекурсивной ветке вычислений, учитывая, что все арифметические операции выполняются за константное время, производится $\Theta\left(\left\lfloor\frac{n}{3}\right\rfloor\right) + \Theta(1) = \Theta\left(\frac{n}{3}\right) = \Theta(n)$ операций. Такое количество обеспечивается циклом от 1 до $\left\lfloor\frac{n}{3}\right\rfloor$.

Будем считать, что n принимает значения равные степеням четверки, это допущение позволит нам избавиться от округления вниз и в то же время никак не повлияет на асимптотическую оценку. Также необходимо учесть, что при $n \leq 50$ алгоритм будет работать за $\Theta(1)$. Аналогично первому алгоритму тут уместна оговорка о копировании массива.

$$T_2(n) = \begin{cases} \Theta(1) & , n \leq 50 \\ 2T_2\left(\dfrac{n}{4}\right) + \Theta(n) & , n > 50 \end{cases} \tag{2}$$

Пункт 2. Разберем второй алгоритм. Докажем, что $T_2(n) = \Theta(n)$. Для этого применим метод подстановки два раза:

1. Гипотеза $T_2(n) = O(n) \Leftrightarrow T_2(n) \leq cn$. Тогда $T_2\left(\frac{n}{4}\right) \leq \frac{cn}{4}$. Делаем подстановку:

$$\begin{aligned} T_2(n) &\leq 2 \cdot \frac{cn}{4} + dn \\ &= \frac{cn}{2} + dn \\ &= \left(\frac{c}{2} + d\right) n \leq cn \end{aligned}$$

d фиксированная положительная константа, найдем для нее подходящее c. Например, при $c = 4d$: $\left(\frac{4d}{2} + d\right) n \leq 4dn \Leftrightarrow 3dn \leq 4dn$, что верно $\forall n \geq 1$ ■

2. Гипотеза $T_2(n) = \Omega(n) \Leftrightarrow T_2(n) \geq cn$. Тогда $T_2\left(\frac{n}{4}\right) \geq \frac{cn}{4}$. Делаем подстановку:

$$\begin{aligned} T_2(n) &\geq 2 \cdot \frac{cn}{4} + dn \\ &= \frac{cn}{2} + dn \\ &= \left(\frac{c}{2} + d\right) n \geq cn \end{aligned}$$

d фиксированная положительная константа, найдем для нее подходящее c. Например, при $c = d$: $\left(\frac{d}{2} + d\right) n \geq dn \Leftrightarrow 1.5dn \geq dn$, что верно $\forall n \geq 1$ ■

$$T_2(n) = O(n) \cap T_2(n) = \Omega(n) \Longrightarrow T_2(n) = \Theta(n)$$