АиСД | SET-2 | A1

Потякин Арсений

TODO:

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.

2. Вычислите асимптотическую точную границу $\Theta(f(n))$ 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 итерации.

```
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
```

Task 1: (Рекуррентное соотношение)

Алгоритм 1: Базовый случай: $n \leq 20$, алгоритм возвращает элемент массива, т.е. $O(1)$. Далее происходит рекурсивный вызов, представимый в виде $T(n-5)$. После этого идут два вложенных цикла, работающие от 1 до $\frac{n}{2}$, что в сумме дает сложность $\frac{n^2}{4}$, или же $O(n^2)$. В конце происходит еще один рекурсивный вызов, который можно представить как $T(n-8)$ и возврат значения, равный константному времени. Таким образом, составим рекуррентное соотношение:

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

Алгоритм 2: Базовый случай: $n \leq 50$, алгоритм возвращает элемент массива, т.е. $O(1)$. Далее происходит рекурсивный вызов, представимый в виде $T(\frac{n}{4})$. После этого идет цикл, работающий от 1 до $\frac{n}{3}$, что дает сложность $O(n)$. В конце происходит еще один рекурсивный вызов, который можно представить как $T(\frac{n}{4})$ и возврат значения, равный константному времени. Таким образом, составим рекуррентное соотношение:

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

Task 2: (Точная граница)
Алгоритм 1: Решим при помощи дерева рекурсии. В корне дерева находится два рекуррентных вызова $n-5$ и $n-8$, а также выполнение двух циклов в сумме $O(n^2)$ операций. Глубину дерева можно приблизительно оценить как $O(\frac{n}{5})$, потому что 5 и 8 близки по величине, так как задачи уменьшаются по порядку. Каждый узел порождает два новых узла, таким образом всего на k-ом уровне будет 2^k узлов. На каждом уровне дерева выполняется $O(n^2)$ операций; так как глубина дерева равна $O(\frac{n}{5})$, то общая работа по дереву будет равна $O(n^2 \times \frac{n}{5}) \Longrightarrow O(n^3) \Longrightarrow \Theta(n^3)$.

Закрепим результат методом итерации. После нескольких итераций можно заметить, что каждый шаг размер задачи уменьшается на фиксированное число (8 в худшем случае), а на каждом шаге выполняется $O(n^2)$ операций. Количество уровней в таком случае $= n$, а значит временная сложность составляет $O(n^3) \Longrightarrow \Theta(n^3)$.

Алгоритм 2: Решим при помощи дерева рекурсии. Корень выполняет $O(n)$ операций и порождает два новых узла с размером задачи $\frac{n}{4}$, т.е. следующий уровень выполняет $O(\frac{n}{4})$ операций и порождает два новых угла с размером задачи $O(\frac{n}{4})$. Таким образом, каждый новый уровень выполняет в два раза больше задач, нежели предыдущий, но размер задачи уменьшается в 4 раза. Это означает, что глубина равна $\log_4 n$. На каждом этапе выполняется работа, пропорциональная $O(n)$. Из этого можно сделать вывод, что общая работа по дереву составит $T(n) = O(n \log_4 n)$, по формуле $\log_4 n = \frac{\log_2 n}{2}$, тогда $T(n) = O(n \frac{\log_2 n}{2}) \Longrightarrow O(n \log n) \Longrightarrow \Theta(n \log n)$