ОПРЕДЕЛЕНИЕ СЛОЖНОСТИ АЛГОРИТМА

Подготовила студентка гр. 23536/1

Антипова Екатерина

ВРЕМЯ РАБОТЫ АЛГОРИТМА

- Пусть $T(n)$ – время работы алгоритма в наихудшем случае или в среднем.
- **Определение.** Если существуют такие $c > 0$ и $n_0 > 0$, что для всех $n \geq n_0$ $T(n) \leq cn^k$, то говорят, что $T(n)$ имеет порядок (степень роста) $O(n^k)$.
- **Определение.** Если существуют такие $c > 0$ и $n_0 > 0$, что для всех $n \geq n_0$ $T(n) \leq cf(n)$, то говорят, что $T(n)$ имеет порядок (степень, функцию роста) $O(f(n))$.

ВЫЧИСЛЕНИЕ ВРЕМЕНИ РАБОТЫ АЛГОРИТМА. ПРАВИЛА ОЦЕНКИ СТЕПЕНЕЙ РОСТА

- **Правило сумм.**

Пусть $T_1(n)$, $T_2(n)$ - вычислительные сложности фрагментов алгоритмов P_1, P_2, имеющие степени роста $O\big(f(n)\big)$ и $O(g(n))$ соответственно. Тогда вычислительная сложность $T_1(n) + T_2(n)$ имеет степень роста $O(\max\{f(n); g(n)\})$.

- **Правило произведений.**

Пусть $T_1(n)$, $T_2(n)$ - вычислительные сложности фрагментов алгоритмов P_1, P_2, имеющие степени роста $O\big(f(n)\big)$ и $O(g(n))$ соответственно. Тогда вычислительная сложность $T_1(n)T_2(n)$ имеет степень роста $O(f(n)g(n))$.

ПРАВИЛА АНАЛИЗА ВРЕМЕНИ ВЫПОЛНЕНИЯ АЛГОРИТМА

1. Время выполнения операций **:=, input, output** пропорционально константе (степень роста O(1)).
2. Время выполнения последовательности инструкций оценивается по правилу сумм.
3. Время выполнения условных операторов складывается из времени вычисления логического выражения (степень роста обычно O(1)) и времени выполнения условных инструкций, т.е. оно имеет степень роста $\max\{T_{then};\ T_{else}\}$.
4. Время выполнения циклов оценивается как количество выполненных итераций, умноженное на наибольшее время выполнения тела цикла или на сумму времени исполнения тела цикла и времени вычисления условия выхода из цикла (входа в цикл).

ПРИМЕР РЕШЕНИЯ

Задание: исследовать зависимость от $n > 0$ общего числа операций, требуемых фрагментом алгоритма.

```
l:=0
for i:=1 to [√(n³ + 1)] do
begin
        k:=i
        while k > 0 do
        begin
            l:=l+k
            k:=[k/3]
        end
end
```

l:=0 — $O(1)$

<u>for</u> i:=1 <u>to</u> $\left[\sqrt{n^3+1}\right]$ <u>do</u>

<u>begin</u>

k:=i — $O(1)$

<u>while</u> k > 0 <u>do</u>

<u>begin</u>

l:=l+k — $O(1)$

k:=[k/3] — $O(1)$

<u>end</u>

<u>end</u>

- Цикл **for** будет повторяться $\left[\sqrt{n^3+1}\right]$ раз, поэтому число выполнений имеет функцию роста

$$O\left(\left[\sqrt{n^3+1}\right]\right) \sim O\left(\sqrt{n^3+1}\right) \sim O\left(n^{\frac{3}{2}}\right).$$

- Цикл **while** будет повторяться пока $k > 0$, а k с каждым разом уменьшается в 3 раза.

Пусть $k=9$: $k=[9/3]=3$;

$k=[3/3]=1$;

$k=[1/3]=0$.

При $k=9$ цикл повторится 2 раза, делаем вывод, что при произвольном k число повторений $\log_3 k$. Поскольку $k := i$, то $k_{max} = i_{max}$, следовательно, максимальное число выполнений имеет функцию роста

$$O(\log_3 i_{max}) \sim O\left(\log_3\left[\sqrt{n^3+1}\right]\right) \sim$$

$$\sim O\left(\log_3 \sqrt{n^3+1}\right) \sim O(\log(n^3+1)) \sim O(\log n).$$

- Цикл **while** вложен в цикл **for**, поэтому сложность внешнего цикла **for** есть $T(n) = T_1(n)T_2(n)$, где $T_1(n)$ – число повторений цикла **for**, а $T_2(n)$ – вычислительная сложность цикла **while.**

- Функции роста: $T_1(n) \sim O\left(n^{\frac{3}{2}}\right)$, $T_2(n) \sim O(\log n)$.

- По правилу произведений:

$$T(n) \sim O(n^{\frac{3}{2}} \log n).$$

- Ответ: $T(n)$ имеет функцию роста $O(n^{\frac{3}{2}} \log n)$.

РЕКУРРЕНТНЫЕ СООТНОШЕНИЯ

МЕТОД ПОДСТАНОВКИ

- Заключается в последовательной подстановке аргументов до достижения начальных условий.
- *В принципе* дает точное решение, однако может потребовать вычисления некоторых сумм.

Примеры решения

Задание: найти методом подстановок решение T и верхнюю оценку T (только верхнюю оценку в случае невозможности вычисления суммы), удовлетворяющего следующему рекуррентному соотношению

$$T(n) = \begin{cases} 1, n = 1, \\ 2T(n-1) + n, n \geq 2. \end{cases}$$

Решение:

$$T(n) = 2T(n-1) + n = 2\Big(2T\big((n-1)-1\big) + (n-1)\Big) + n =$$

$$= 2^2\, T(n-2) + 2(n-1) + n = 2^2\Big(2T\big((n-2)-1\big) + (n-2)\Big) + 2(n-1) + n =$$

$$= 2^3 T(n-3) + 2^2(n-2) + 2(n-1) + n = \cdots =$$

$$= 2^k\, T(n-k) + \sum_{j=0}^{k-1} 2^j (n-j) =$$

[для достижения начального условия необходимо, чтобы $n - k = 1 \Rightarrow k = n - 1$, подставляем $k = n - 1$ и начальное условие $T(1) = 1$]

$$= 2^{n-1} T(1) + \sum_{j=0}^{n-2} 2^j (n-j) = 2^{n-1} + n \sum_{j=0}^{n-2} 2^j - \sum_{j=1}^{n-2} 2^j j\,.$$

Для вычисления последней суммы выведем общую формулу:

$$\sum_{j=1}^{n} a^j j = a\sum_{j=1}^{n} a^{j-1} j = a\sum_{j=1}^{n}(a^j)' = a\left(\sum_{j=0}^{n} a^j\right)' = a\left(\frac{1-a^{n+1}}{1-a}\right)' =$$

$$= a\left(\frac{-(n+1)a^n(1-a)+(1-a^{n+1})}{(1-a)^2}\right).$$

Теперь подставляем эту формулу с $a = 2$ и заканчиваем вычисление:

$$T(n) = 2^{n-1} + n\sum_{j=0}^{n-2} 2^j - 2\sum_{j=0}^{n-2} 2^{j-1} j = 2^{n-1} + n\sum_{j=0}^{n-2} 2^j - 2\sum_{j=0}^{n-2}(2^j)' =$$

$$= 2^{n-1} + n\frac{1-2^{n-1}}{1-2} - 2\left(\frac{-(n-1)2^{n-2}(1-2)+(1-2^{n-1})}{(1-2)^2}\right) =$$

$$= 2^{n-1} + n(2^{n-1}-1) - (n-1)2^{n-1} - 2 + 2^n = 2^{n+1} - n - 2.$$

Ответ: $T(n) = 2^{n+1} - n - 2;\, T(n) \sim O(2^n)$.

Задание: найти методом подстановок решение T и верхнюю оценку T (только верхнюю оценку в случае невозможности вычисления суммы), удовлетворяющего следующему рекуррентному соотношению

$$T(n) = \begin{cases} 1, n = 0, \\ 2, n = 1, \\ 3T(n-2) + \sqrt{n}, n \geq 2. \end{cases}$$

Решение:

$$T(n) = 3T(n-2) + \sqrt{n} = 3\left(3T(n-4) + \sqrt{n-2}\right) + \sqrt{n} =$$
$$= 3^2\, T(n-4) + 3\sqrt{n-2} + \sqrt{n} = 3^2\left(3T(n-6) + \sqrt{n-4}\right) + 3\sqrt{n-2} + \sqrt{n} =$$
$$= 3^3T(n-6) + 3^2\sqrt{n-4} + 3\sqrt{n-2} + \sqrt{n} = \cdots = 3^kT(n-2k) + \sum_{j=0}^{k-1} 3^j\sqrt{n-2j}\ .$$

1. Для достижения начального условия $T(1) = 1$ необходимо

$$n - 2k = 1 \ \Rightarrow k = \frac{n-1}{2}$$

(n нечётно). Подставляем k и начальное условие:

$$T(n)=3^{\frac{n-1}{2}}T(1)+\sum_{j=0}^{\frac{n-3}{2}}3^j\sqrt{n-2j}=2\cdot3^{\frac{n-1}{2}}+\sum_{j=0}^{\frac{n-3}{2}}3^j\sqrt{n-2j}\le 2\cdot3^{\frac{n-1}{2}}+\sqrt{n}\sum_{j=0}^{\frac{n-3}{2}}3^j=2\cdot3^{\frac{n-1}{2}}+\sqrt{n}\frac{3^{\frac{n-1}{2}}-1}{2}.$$

Тогда $T(n)\sim O\left(\sqrt{n}\cdot 3^{\frac{n}{2}}\right)$.

2. Для достижения начального условия $T(0)=1$ необходимо

$$n-2k=2\ \Rightarrow k=\frac{n}{2}$$

(n чётно). Подставляем k и начальное условие:

$$T(n)=3^kT(n-2k)+\sum_{j=0}^{k-1}3^j\sqrt{n-2j}\ =3^{\frac{n}{2}}\cdot T(0)+\sum_{j=0}^{\frac{n-2}{2}}3^j\sqrt{n-2j}=$$

$$=3^{\frac{n}{2}}+\sum_{j=0}^{\frac{n-2}{2}}3^j\sqrt{n-2j}\le 3^{\frac{n}{2}}+\sqrt{n}\sum_{j=0}^{\frac{n-2}{2}}3^j=3^{\frac{n}{2}}+\sqrt{n}\frac{3^{\frac{n}{2}}-1}{2}.$$

Тогда $T(n)\sim O\left(\sqrt{n}\cdot 3^{\frac{n}{2}}\right)$.

Ответ: $T(n)\sim O\left(\sqrt{n}\cdot 3^{\frac{n}{2}}\right)$.

ОБЩЕЕ РЕШЕНИЕ КЛАССА РЕКУРРЕНТНЫХ СООТНОШЕНИЙ

- Пусть исходную задачу размера n можно разделить на a подзадач размера $\frac{n}{b}$, $d(n)$ – число операций, требуемых на сборку подзадач в задачу размера n, – управляющая функция. Считаем, что задача размера 1 требует 1 операцию.
- Такой рекурсивный алгоритм приводит к рекуррентному соотношению:

$$T(n) = \begin{cases} 1, n = 1, \\ aT\left(\frac{n}{b}\right) + d(n), n > 1. \end{cases}$$

- Структура решения:

$$T(n) = n^{\log_b a} + \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right), n = b^k.$$

Однородное решение

Частное решение

- Практическое правило:

1. Если однородное решение больше управляющей функции, то частное решение имеет тот же порядок роста, что и однородное решение.
2. Если управляющая функция растет на порядок n^s $(s > 0)$ быстрее однородного решения, то частное решение имеет тот же порядок роста, что и управляющая функция.
3. Если управляющая функция имеет тот же порядок роста, что и однородное решение или растет не медленнее, чем $(\log n)^k$ $(k > 0)$, то частное решение растет, как управляющая функция, умноженная на $\log n$.

- Функция f называется мультипликативной, если $f(xy) = f(x)f(y)$.
- Если управляющая функция мультипликативна: то частное решение равно

$$\sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right) = \frac{n^{\log_b a} - n^{\log_b d(b)}}{\frac{a}{d(b)} - 1}.$$

Пример решения

Задание: решить рекуррентное соотношение и найти степень роста решения

$$T(n) = \begin{cases} 1, n = 1, \\ 2T\left(\frac{n}{5}\right) + \sqrt{n}, n \geq 2. \end{cases}$$

Решение:

$$a = 2, b = 5, d(n) = \sqrt{n};\ k = \log_5 n;$$

$$T(n) = n^{\log_5 2} + \sum_{j=0}^{k-1} 2^j \sqrt{5^{k-j}} = n^{\log_5 2} + \sum_{j=0}^{\log_5 n - 1} 2^j \sqrt{\frac{5^{\log_5 n}}{5^j}} =$$

$$= n^{\log_5 2} + \sqrt{n} \sum_{j=0}^{\log_5 n - 1} \left(\frac{2}{\sqrt{5}}\right)^j = n^{\log_5 2} + \frac{n^{\log_5 2} - n^{\log_5 \sqrt{5}}}{\frac{2}{\sqrt{5}} - 1} = n^{\log_5 2} + \sqrt{5}\frac{\sqrt{n} - n^{\log_5 2}}{\sqrt{5} - 2}.$$

Ответ:

$$T(n) = n^{\log_5 2} + \sqrt{5}\frac{\sqrt{n} - n^{\log_5 2}}{\sqrt{5} - 2};$$

$T(n) \sim O(\sqrt{n})$.