# Алгоритмы. ДЗ на неделю 1.

Прохоров Юрий, 771

---

## Задача 5 (с семинара)

### Алгоритм

```
int j = 1;
for (int i = 1; i < n + 1 && j < k + 1; i++) {
    if (x[i] == y[j]) j++;
}
if (j == k + 1) return 1;
return 0;
```

### Корректность

Рассмотрим три случая.

**1. Введены корректные последовательности.** То есть на выводе должен быть ответ 1. Пусть последовательность $\{t_1, t_2, \ldots, t_k\}$ такая, что $x[t_i] = y[i]$ для $i = 1, 2, \ldots, k$. Кроме того, пусть значения $t_i$ – наименьшие возможные, то есть

$$\forall i \in [1, t_1 - 1] \longmapsto x\,[i] \neq y\,[1]$$

$$\forall i \in [t_1 + 1, t_2 - 1] \longmapsto x\,[i] \neq y\,[2]$$

$$\ldots$$

$$\forall i \in [t_{k-1} + 1, t_k - 1] \longmapsto x\,[i] \neq y\,[k]$$

Таким образом, в цикле алгоритма при $i = t_1, t_2, \ldots, t_k$ значение переменной $j$ будем увеличиваться на 1. После последнего инкремента цикл завершится при $j = k + 1$, что соответствует выводу 1.

**2. Последовательность $\{x\}$ не содержит $y\,[1]$.** Тогда условие в теле цикла ни разу не выполнится, и переменная $j$ по окончании цикла останется равна 1, откуда следует вывод 0.

**3. Последовательность $\{x\}$ содержит только некоторую начальную подпоследовательность от $\{y\}$.** Пусть последовательность $\{x\}$ содержит элементы $\{y\,[1], y\,[2], \ldots, y\,[p]\}$ в том же порядке, $p < k$. Определим последовательность $\{t_1, t_2, \ldots, t_p\}$ по тому же правилу, что и в случае 1.

Пусть цикл алгоритма выполнился до значения $i = t_p$ включительно. После этого момента переменная $j = p$. Начиная с этого момента, нам остается найти внутри последовательности $\{x\,[t_p + 1], \ldots, x\,[n]\}$ последовательность $\{y\,[p + 1], y\,[p + 2], \ldots, y\,[k]\}$. Но эта задача в точности совпадает со случаем 2, поэтому после выполнения остатка цикла переменная $j$ останется равной $p < k$.

### Оценка по времени

Если $T(n, k)$ – время выполнения алгоритма, то

$$T(n, k) = c_1(n + k) + c_2 n + c_3 n + c_4 k + c_5 \Longrightarrow T(n, k) = \Theta(n + k),$$

где $c_1$ – константа ввода, $c_2$ – константы сравнений и присваиваний в шапке цикла, $c_3$ – константа сравнения в теле цикла, $c_4$ – константа присваивания в теле цикла, $c_5$ – инициализация переменных, вывод и другое. Стоит заметить, что в некоторых случаях итераций цикла будет меньше (не меньше, чем $max(n, k)$), но из-за ввода сложность остается $\Theta(n + k)$.

# Задача 1 (ДЗ)

**а) Верно.**

Покажем, что $n = O(n \log_2 n)$, то есть, что

$$\exists C > 0, n_0 \in \mathbb{N} : \forall n \geq n_0 \longmapsto n < Cn \log_2 n$$

Пусть $C = 1, n_0 = 3$, тогда

$$Cn \log_2 n = n \log_2 n > n \log_2 3 > n \cdot 1 = n, \qquad n \geq n_0 = 3.$$

**б) Неверно.**

Допустим, утверждение верно, тогда

$$\exists \varepsilon > 0, C > 0, n_0 \in \mathbb{N} : \forall n \geq n_0 \longmapsto n^{1+\varepsilon} < Cn \ln n \Longrightarrow$$

$$\Longrightarrow n^{\varepsilon} < C \ln n \Longrightarrow \frac{n^{\varepsilon}}{C \ln n} < 1$$

По свойствам предела последовательности и из приведенного выше следует, что если предел последовательности существует, то он удовлетворяет неравенству

$$\lim_{n \to \infty} \frac{n^{\varepsilon}}{C \ln n} \leq 1.$$

Рассмотрим функцию $f$:

$$f(x) = \frac{x^{\varepsilon}}{C \ln x}.$$

По правилу Лопиталя:

$$\lim_{x \to +\infty} \frac{x^{\varepsilon}}{C \ln x} = \lim_{x \to +\infty} \frac{\varepsilon x^{\varepsilon - 1}}{C \frac{1}{x}} = \lim_{x \to +\infty} \frac{\varepsilon}{C} x^{\varepsilon} = +\infty.$$

По определению предела функции по Гейне, для любой бесконечно большой последовательности, в частности, для последовательности натуральных чисел верно, что

$$\lim_{n \to \infty} \frac{n^{\varepsilon}}{C \ln n} = +\infty,$$

что является противоречием.

# Задача 2 (ДЗ)

**1. а) Возможно.**

$$\left.\begin{array}{l} f(n) = n \log n \\ g(n) = 1 \end{array}\right\} \qquad \Longrightarrow \qquad h(n) = n \log n = \Theta(n \log n).$$

**1. б) Невозможно.**

Допустим, это возможно, тогда

$$\exists C_3 > 0, n_3 \in \mathbb{N} : \forall n \geq n_3 \longmapsto n^3 < C_3 h(n) = C_3 \frac{f(n)}{g(n)}$$

Пусть для условий $f(n) = O(n^2)$, $g(n) = \Omega(1)$ константы равны $C_1, n_1$ и $C_2, n_2$ соответственно. Пусть $n_0 = max(n_1, n_2, n_3)$. Тогда при $n \geq n_0$:

$$n^3 < C_3 \frac{f(n)}{g(n)}, \qquad f(n) < C_1 n^2, \qquad 1 < C_2 g(n), \qquad \frac{1}{g(n)} < C_2 \Longrightarrow$$

$$n^3 < C_3 \frac{f(n)}{g(n)} < C_3 C_1 n^2 C_2 \Longrightarrow n < C = C_1 C_2 C_3.$$

Однако при $n = max(n_0 + 1, C + 1)$ последнее неравенство не выполняется – противоречие.

**2. Верхняя и нижняя оценки.**
Из пункта **1(б)** получим, что при $n > n_0 = max(n_1, n_2)$:

$$h(n) = \frac{f(n)}{g(n)} < C_1 n^2 C_2 = Cn^2 \Longrightarrow h(n) = O(n^2)$$

Пример таких функций:

$$f(n) = n^2, \qquad g(n) = 1, \qquad h(n) = n^2.$$

Более хорошей (более низкой) оценки сверху придумать нельзя, так как данных пример имеет точную оценку $\Theta(n^2)$, а верхняя оценка не может быть меньше точной.

Нижней оценки не существует, потому что можно взять функции

$$f(n) = \frac{1}{n^{100}}, \qquad g(n) = 1, \qquad h(n) = \frac{1}{n^{100}},$$

где показатель степени в знаменателе функции $f$ можно брать бесконечно большим, из-за чего лучшая нижняя оценка функции $h$ будет бесконечно малой.

# Задача 3 (ДЗ)

**(а)** Индуктивное расширение:

$$g(x_1, \ldots, x_n) = (\sum_{i=1}^{n} x_i;\ x.length)$$

Для этой функции существует функция $t$. Для краткости обозначим $x = \{x_1, \ldots, x_n\}$. $t(x)_1$ и $t(x)_2$ – первое и второе значения функции соответственно.

$$t(g(x), x_{n+1}) = (t(x)_1 + x_{n+1};\ \ t(x)_2 + \{x_{n+1}\}.length) = (\sum_{i=1}^{n+1} x_i;\ \{x, x_{n+1}\}.length) = t(x_1, \ldots, x_n, x_{n+1}).$$

**(б)** Индуктивное расширение:

$$g(x) = g(x_1, \ldots, x_n) = (\max(x_1, \ldots, x_n);\ \sum_{x_i = g(x)_1} 1) = (\text{max; число элементов, равных max})$$

Найдем соответствующую функцию $t$:

$$t(g(x), x_{n+1}) = (\max(g(x), x_{n+1});\ \sum_{x_i = t_1} 1) = (\max(x, x_{n+1});\ \sum_{x_i = t_1} 1) = g(x_1, \ldots, x_{n+1}),$$

где $t_1 = t(g(x), x_{n+1})_1$.

**(в)** Индуктивная функция:

$$f(x) = f(x_1, \ldots, x_n) = (x_n; Seq; Ans),$$

где $Seq$ – Sequence, число повторяющихся элементов с конца; $Ans$ – Answer, текущий ответ. Соответствующая функция $F$ из определения индуктивной функции:

$$F(f(x), x_{n+1}) = \left( x_{n+1};\ \begin{cases} Seq{+}{+}, & x_{n+1} = x_n \\ 1, & x_{n+1} \neq x_n \end{cases};\ \max(Ans, F_2) \right) = f(x_1, \ldots, x_{n+1}),$$

где $F_2 = F(f(x), x_{n+1})_2$.

Построим линейный по времени онлайн-алгоритм:

```
int last = x[1], Seq = 1, Ans = 1;
for (int i = 2; i < n + 1; i++) {
    if (x[i] == last) {Seq++} else {Seq = 1}
    last = x[i];
    Ans = max(Ans, Seq);
}
return Ans;
```

# Задача 4 (ДЗ)

## Алгоритм

Определим INF и NEG_INF как наибольшее и наименьшее возможные числа в используемом типе данных. На вход поданы последовательности $x[1..m]$, $y[1..p]$, $z[1..k]$ длинами $m$, $p$ и $k$ соответственно. Определим $n = m + p + k$.

```
x[m+1] = INF; y[p+1] = INF; z[k+1] = INF;
last = NEG_INF;
Ans = 0;
a = 1; b = 1; c = 1;
for (int i = 0; i < n; i++) {
    if (x[a] ≤ y[b] && x[a] ≤ z[c]) {
        if (last < x[a]) Ans++;
        last = x[a];
        a++;
    } else if (y[b] ≤ z[c]){
        if (last < y[b]) Ans++;
        last = y[b];
        b++;
    } else {
        if (last < z[c]) Ans++;
        last = z[c];
        c++;
    }
}
return Ans;
```

## Корректность

На первой итерации цикла выбирается самое минимальное число и записывается в переменную $last$. В следующей итерации это число заменяется на следующее в соответствующей последовательности. По условию, последовательности отсортированы по возрастанию, поэтому вычисленный на новой итерации минимум не меньше значения переменной $last$. Из этого следует, что если бы сохраняли эти числа (вычисленные минимумы) в отдельном массиве $Q$, то он был бы отсортирован по возрастанию.

На каждом шаге этот минимум ($min$) сравнивается с переменной $last$, что эквивалентно сравнению соседних элементов массива $Q$. Если $min = last$, то значение ответа $Ans$ не меняется, иначе – увеличивается на 1.

Допустим, цикл впервые дошел до последнего элемента массива. Пусть для определенности этим массивом оказался массив $x$; только что переменной $a$ установилось значение $a = m + 1$. Это означает, что пока не закончился хотя бы один из других массивов, массив $x$ не может содержать минимум, так как $x[m + 1] = INF$. Аналогично со вторым массивом $y$.

На каждом шаге цикла ровно одна из переменных $a$, $b$ или $c$ увеличивается на 1. Поэтому когда закончится последний массив $z$, переменные будут иметь значения:

$$a = m + 1, \qquad b = p + 1, \qquad c = k + 1.$$

Общий инкремент всех переменных равен $m+p+k = n$, что означает, что цикл закончит свою работу, поэтому сравнения трех величин $INF$ никогда не произойдет.

## Оценки по времени

Если $T(n)$ - время работы алгоритма, то

$$T(n) = c_1 n + c_2 n + c_3 = \Theta(n),$$

где $c_1$ - константа при вводе, $c_2$ - константа, за которую выполняется тело цикла, $c_3$ - константа при инициализации переменных вне цикла, при выводе и другое.

При любом вводе алгоритм работает за одно и то же время (с точки зрения порядка роста).

# Задача 5 (ДЗ)

## Алгоритм

Пусть $A$ – массив длины $n$, содержащий элементы последовательности. Объявим массивы $B$ и $C$ длины $n$ типа данных *integer*. $Q$ – массив, являющийся выводом.

```
B[0] = 0; C[0] = -1; Len = 0;
for (int i = 0; i < n; i++) {
    int e = 0; int pos = -1;
    for (int j = 0; j < i; j++) {
        if (A[i] > A[j] && B[j] > e) {e=B[j]; pos = j;}
    }
    if (pos > -1) {B[i] = e + 1;} else {B[i] = 0;}
    C[i] = pos;
    Len = max(B[i], Len);
}
int i = 0;
while (B[i] < Len) i++;
while (B[i] > -1) {
    Q[B[i]] = A[i];
    i = C[i];
}
```

## Корректность

На каждом шаге внешнего цикла вычисляется длина наибольшей строго возрастающей подпоследовательности с учетом того, что на предыдущем шаге она тоже была вычислена, и записывается в массив $B$. Для этого на $i$-м шаге выбираются все элементы среди $\{A[0], \ldots, A[i-1]\}$, меньшие, чем $A[i]$, среди их значений $B[i]$ выбирается максимум – $B[j], j \in [1, \ldots, i-1]$, то есть $j$ – конец самой длинной искомой подпоследовательности. Тогда элемент $A[i] > A[j]$ – хвост новой наибольшей строго возрастающей подпоследовательности длиной $B[j]+1$. Сама позиция $j$ сохраняется в массив $C$ на место $C[i]$.

Тот факт, что на каждом шаге алгоритм вычисляет самую длинную строго возрастающую подпоследовательность, гарантирует, что на выходе мы получим верный результат.

Для извлечения искомого массива мы сначала найдем любой элемент, имеющий максимальное значение массива $B$ – элемент $A[i]$. Это последний, наибольший элемент подпоследовательности. Перейдя по адресу $C[i]$ мы попадем в предыдущий элемент и так далее, пока $B[i] \geq 0$.

## Оценки по времени

Внешний цикл алгоритма выполняется $n$ раз, а внутренний - $i$ раз. Поэтому общее число итераций внутреннего цикла, тело которого выполняется за константное время:

$$\sum_{i=0}^{n-1} = \frac{n(n-1)}{2} \text{ повторений}$$

Время работы алгоритма

$$T(n) = c_1 n + c_2 \frac{n(n-1)}{2} + c_3 = \Theta(n^2),$$

где $c_1$ – константа ввода, $c_2$ – константа внутреннего цикла, $c_3$ – другие константы.

При любом вводе алгоритм выполняет $\Theta(n^2)$ повторений цикла, поэтому оценка является точной при любой входной последовательности.