Домашняя работа

Котов Артем, МОиАД2020

4 февраля 2021 г.

Содержание

Task 1

(a) Введем $\widetilde{C} = \max\limits_{n} \left\{ \frac{f(n)}{g(n)} \right\}$. При ограничениях, наложенных на ф-ции f и g, $\widetilde{C} > 0$, тогда мы можем $\forall n$ с $\widetilde{C} > 0$ рассмотреть следующую цепочку: $\forall n \ f(n) \leq f(n) \leq \widetilde{C} g(n) \Longrightarrow f(n) \leq \widetilde{C} g(n)$. Таким образом, в определении O-большого можно пренебречь условием $\exists N : \ldots$ (это верно для обоих случаев: $\mathbb{N} \to \mathbb{N}$ и $\mathbb{N} \to \mathbb{R}_+$)

(b) Рассмотрим функции

$$f(n) = \begin{cases} 3, n = 1 \\ n, n > 1 \end{cases}$$

$$g(n) = n^2$$

Очевидно, что в данном случае нельзя пренебречь условием $\exists N : \ldots$, так как вначале не для $\forall C f(n) < C g(n)$. Если этим условием не пренебрегать, то $f(n) = o(g(n))$. Опять же, это верно для обоих случаев $\mathbb{N} \to \mathbb{N}$ и $\mathbb{N} \to \mathbb{R}_+$.

P.S. Хотя формально, у нас не было ограничений на поведение функций f и g, для случая $\mathbb{N} \to \mathbb{R}_+$, важно, чтобы они были достаточно хорошие (то есть, по видимому, не имели сингулярностей в конечных точках). Сделав поправку на “природу” исследуемых функций (оценка сложности алгоритма), мы можем вполне разумно считать, что с ростом n функции также возрастают и не имеют каких-то странных разрывов по типу $\frac{1}{x - x_0}$

Task 2

$$f(n) = o(g(n)) \Longleftrightarrow \lim \frac{f(n)}{g(n)} = 0$$

$$\frac{2^{f(n)}}{2^{g(n)}} \to 0? \quad 2^{f(n) - g(n)} = 2^{g(n)\left(\frac{f(n)}{g(n)} - 1\right)} \to 2^{-g(n)}$$

Почему это работает? Так как $g(n) > 1$, то мы всегда можем так вынести $g(n)$ в показателе степени. Так как $f(n) > 1$, то $\frac{f(n)}{g(n)} \to 0 \Longrightarrow g(n) \to \infty \Longrightarrow 2^{-g(n)} \to 0$.

Task 3

A	B	O	o	Θ	ω	Ω
$\log^k n$	n^ε	$+$	$+$	$-$	$-$	$-$
n^k	C^n	$+$	$+$	$-$	$-$	$-$
$\sqrt{n}$	$n^{\sin n}$	$-$	$-$	$-$	$-$	$-$
2^n	$2^{n/2}$	$-$	$-$	$-$	$+$	$+$
$n^{\log m}$	$m^{\log n}$	$+$	$-$	$+$	$-$	$+$
$\log n!$	$\log(n^n)$	$+$	$-$	$+$	$-$	$+$

1) Приведем $\log^k n$ к натуральному логарифму: $\log^k n = \left(\frac{\ln n}{\ln 2}\right)^k$, а функцию B приведем к экспоненциальной форме: $e^{\varepsilon \ln n}$. Произведем замену переменных: $\ln n = x$, тогда наша задача сведена к сравнению следующих функций (опушена несущественная константа $\ln 2$): $A = x^k$ и $B = e^{\varepsilon x}$

   Замечание. *Вообще, этот случай сведен ко второму случаю с точностью до обозначения констант, разбор второго случая ниже.*

2) • O, o) Рассмотрим сначала o-малое: $n^k = o(C^n)$

   Доказательство.
   $$\frac{n^k}{C^n} = \frac{e^{k \ln n}}{e^{n \ln C}} = e^{k \ln n - n \ln C} \tag{1}$$
   Рассмотрим разные C:
   Если $C < 1$, то $\ln C < 0 \Longrightarrow (1) = e^{k \ln n + n|\ln C|} \xrightarrow[n\to\infty]{} \infty \quad \forall k.$
   Если $C = 1$, то $\ln C = 0 \Longrightarrow (1) = e^{k \ln n} \xrightarrow[n\to\infty]{} \infty \quad \forall k > 0$. В случае же, если $k < 0$, то $(1) \xrightarrow[n\to\infty]{} 0$
   Если $C > 1$, то $\ln C > 0 \Longrightarrow (1) = e^{k \ln n - n|\ln C|} \xrightarrow[n\to\infty]{} 0 \quad \forall k.$
   Для разумных значений параметров ($C > 1, k > 0$) $n^k = o(C^n)$ ■

   Теперь займемся O: $n^k = O(C^n)$

   Доказательство. Так как в определение o-малого $\forall C\, \exists N : \forall n > N\ f(n) < Cg(n)$, то, очевидно, что какое-то конкретное C существует, следовательно, если $f(n) = o(g(n))$, то $f(n) = O(g(n))$.
   В нашем случае, $n^k = o(C^n) \Longrightarrow n^k = O(C^n)$ ■

   • $n^k \neq \omega(C^n)$

   Доказательство. $n^k = \omega(C^n) \Leftrightarrow C^n = o(n^k)$, покажем, что это равенство неверно (для разумных значений параметров):
   $$\frac{C^n}{n^k} = e^{n \ln C - k \ln n} \tag{2}$$

Если $C > 1$, то (2) $\xrightarrow[n\to\infty]{} \infty$.
Таким образом, $C^n \neq o(n^k) \Rightarrow n^k \neq \omega(C^n)$ ■

Замечание. *$C^n = o(n^k)$ возможно в случае, когда $C \leq 1$, тогда $n^k = \omega(C^n)$*

- $n^k \neq \Omega(C^n)$

  Доказательство. $n^k = \Omega(C^n) \Leftrightarrow C^n = O(n^k)$. Аналогично предыдущему док-ву можно показать, что последнее неверно, следовательно, $n^k \neq \Omega(C^n)$. ■

- $n^k \neq \Theta(C^n)$

  Доказательство. $n^k = \Theta(C^n) \Leftrightarrow n^k = O(C^n)$ и $C^n = O(n^k)$. Нарушение последнего уже показано в предыдущем случае, следовательно $n^k \neq \Theta(C^n)$ ■

3) В этом случае вообще все плохо с "хвостом" у $n^{\sin n}$: он болтыхается между 0 и n, следовательно у нас никогда не будет универсальных констант C, так, чтобы эти условия выполнялись для всего хвоста ($\sqrt{n}$ — монотонно возрастающая функция).

Доказательство. Рассмотрим поведение функции $n^{\sin n}$, она ограничена сверху n, а снизу 0, то есть $0 \leq n^{\sin n} \leq n \, \forall n$.

Замечание. *вообще, $n^{\sin n}$ ни при каких натуральных n не может быть равной n, так как для этого необходимо, чтобы $\sin n = 1$, а это возможно только для иррационального $n = \pi/2$.*

Рассмотрим, например, $\sqrt{n} \neq o(n^{\sin n})$ по определению о-малого:

$$\forall C > 0 \; \exists N : \forall n > N \; \sqrt{n} < C n^{\sin n}$$

но $n^{\sin n}$ при больших n всегда имеет значения (так как $\sin$ — периодическая функция от -1 до 1), сколько угодно близких к 0, следовательно, таких C нет.

Замечание. *Так как $-1 \leq \sin n \leq 1$, то можно ограничить $n^{\sin n} \geq n^{-1} = \frac{1}{n} \xrightarrow[n\to\infty]{} 0$*

Аналогично показывается и $\sqrt{n} \neq O(n^{\sin n})$, а, следовательно, и $\sqrt{n} \neq \Theta(n^{\sin n})$.

Рассмотрим еще $\sqrt{n} \neq \omega(n^{\sin n})$: по определению

$$\forall C > 0 \; \exists N : \forall n > N \; C n^{\sin n} < \sqrt{n}$$

Как ранее уже говорилось, $n^{\sin n}$ ограничена сверху n, а, точнее, всегда имеются значения, сколько угодно близкие к n, при этом $\frac{n}{\sqrt{n}} \xrightarrow[n\to\infty]{} \infty$, следовательно, нет таких C, чтобы выполнялось $C n^{\sin n} < \sqrt{n} \quad \forall n > N$, таким образом, $\sqrt{n} \neq \omega(n^{\sin n})$. Аналогично, $\sqrt{n} \neq \Omega(n^{\sin n})$ ■

4) · $2^n \neq o(2^{n/2})$ и $2^n \neq O(2^{n/2})$

Доказательство. $2^{n-n/2} = 2^{n/2} \xrightarrow[n\to\infty]{} \infty \Longrightarrow 2^n \neq o(2^{n/2})$ и $2^n \neq O(2^{n/2})$ ■

· $2^n \neq \Theta(2^{n/2})$

Доказательство. так как $2^n \neq O(2^{n/2})$, то из $2^n \neq \Theta(2^{n/2})$ ■

· $2^n = \omega(2^{n/2})$

Доказательство. $2^n = \omega(2^{n/2}) \Leftrightarrow 2^{n/2} = o(2^n)$.
Рассмотрим $\frac{2^{n/2}}{2^n} = 2^{-n/2} \xrightarrow[n\to\infty]{} 0 \Longrightarrow 2^{n/2} = o(2^n) \Longrightarrow 2^n = \omega(2^{n/2})$ ■

· $2^n = \Omega(2^{n/2})$

Доказательство. Так как $2^n = \omega(2^{n/2})$, то $2^n = \Omega(2^{n/2})$ ■

5) Преобразуем к натуральному логарифму и приведем к единой экспоненциальной форме: $e^{\frac{\ln n \ln m}{\ln 2}}$ что для A, что для B (т. е. $A \equiv B$), следовательно, это одинаковые функции, для них нарушаются условия, в которых $\forall C > 0$.

6) · $\log n! \neq o(\log n^n)$, но $\log n! = O(\log n^n)$

Доказательство. Воспользуемся ф-лой Стирлинга (ох и не хотелось так доказывать, но в предыдщем варианте я что-то совсем бред написал): $n! \simeq \frac{n^{n+1/2}}{e^n}$ (выкинули незначащие константные коэффициенты)

$$\log n! \simeq \ln n! \simeq \ln \frac{n^{n+1/2}}{e^n} = (n + 1/2) \ln n - n$$

Для B функция $n \ln n$. Рассмотрим отношение $\frac{n \ln n + 1/2 \ln n - n}{n \ln n} = 1 + \frac{1}{2n} - \frac{1}{\ln n} \xrightarrow[n\to\infty]{} 1$, следовательно, $\log n! \neq o(\log n^n)$, но $\log n! = O(\log n^n)$ ■

· $\log n! = \Theta(\log n^n)$

Доказательство. Уже показали, что $\log n! = O(\log n^n)$, осталось показать, что $\log n^n = O(\log n!)$:

$$\frac{n \ln n}{n \ln n + 1/2 \ln n - n} = \frac{1}{1 + 1/2 \frac{1}{n} - \frac{1}{\ln n}} \xrightarrow[n\to\infty]{} 1$$

Замечание. *Это заодно показывает, что* $\log n^n \neq o(\log n!)$

Следовательно, $\log n^n = O(\log n!)$ и окончательно имеем $\log n! = \Theta(\log n^n)$ ■

· $\log n! \neq \omega(\log n^n)$

Доказательство. На предыдущем шаге было показано, что $\log n^n \neq o(\log n!)$, следовательно, $\log n! \neq \omega(\log n^n)$, так как они эквивалентны. ■

· $\log n! = \Omega(\log n^n)$

Доказательство. Так как $\log n! = \Omega(\log n^n) \Leftrightarrow \log n^n = O(\log n!)$, а последнее мы показали на пред-предыдущем шаге, то $\log n! = \Omega(\log n^n)$ ■

Task 4

$$[a_1, \ldots, a_n] \in \mathbb{N}, S \in \mathbb{N}$$

Введем $r_i = \max\limits_{r} \left\{ \sum\limits_{j=i}^{r} a_j \leq S \right\}$, то есть это максимальное количество элементов справа от i-ого, которое можно взять в сумму и не "перепрыгнуть" через S.

Начнем находить такие r_i:

1) Поставим два указателя (ал-я левый и правый) на первый элемент
2) Пока $\sum\limits_{i=1}^{r} a_i \leq S$ будем сдвигать правый указать ($r+= 1$) до тех пор, пока не дойдем до некоторого элемента, добавление в сумму которого превзойдет S
3) Откатим правый указатель на 1 влево: индекс элемента, на который указывает правый указатель и есть искомый r_1 (в этот момент мы могли уже получить искомую сумма, поэтому проверим, $\sum\limits_{i=1}^{r_1} a_i = S$?)
4) Если нет, то подвинем левый указатель (он все это время оставался на первом элементе) на единицу вправо ($l+= 1$).
5) Так как все элементы — натуральные числа, то $\sum\limits_{i=2}^{r_1} a_i < \sum\limits_{i=1}^{r_1} a_i$
6) Проверим, можем ли подвинуть правый указатель, если да, то продолжаем так двигать, пока сумма не окажется большей S, затем повторим сдвижку левого указателя. Если изначально нельзя было сдвинуть правый указатель, то $r_2 = r_1$ и мы подвинем левый указатель еще раз на 1 направо и т.д.
7) В итоге, рано или поздно, либо на каком-то элементе получим сумму, равную S, либо упремся правым указателем в конец массива (тут мы можем констатировать факт что, либо последняя найденная сумма равна S, либо такого искомого отрезка для данного S в массиве нет), либо два указателя встретятся на одном элементе (что значит, что этот элемент сам по себе больше, чем S). В последнем случае, сдвинем оба указателя на следующий элемент и будем продолжать подобную схему далее.

В описанной схеме, в худшем случае мы сделаем два пробега по всему массиву (случай, когда у нас только последний элемент равен S, а остальных не хватает, чтобы набрать необходимую сумма).

Task 5

(a) Используем стэк s:

- Для каждого элемента a_i исходного массива делаем следующие (проходим массив слева напарво, то есть $i = 1 \dots n$):
  1) Пока s непуст и $s_{\text{top}} \geq a_i$: S.pop
  2) Если s пуст, то у a_i нет элементов, которые левее и меньше его
  3) Иначе, самый правый элемент, который левее и меньше a_i есть s_{top}
  4) s.push(a_i)

(b) Аналогично предыдущему пункту, только теперь массив проходим справа налево, то есть $i = n \dots 1$

(c) С использованием двух предыдущих пунктов, мы можем найти для каждого элемента a_i соответствующие l_i и r_i, в интервале которых a_i будет минимальным. Составим $\pi_i = (r_i - l_i + 1) \cdot \min\limits_{j \in [l_i, r_i]} a_j$ и сохраним их вместе с парой (l_i, r_i), то есть что-то типа кортежа $\Pi_i = (\pi_i, l_i, r_i)$. Найдем $\max\limits_i \Pi_i[0]$. Соответствующие l_i и r_i и будут искомыми.

(d) Аналогично предыдущему, только если в прошлой задаче “весом” каждого элемента была длина промежутка, на котором он минимален, то теперь вес — сумма всех элементов на этом промежутке.

Task 6

Рассмотрим исходный массив $[a_1, \dots, a_n]$

1) Создадим новый массив $[b_1, \dots, b_n]$ такой, что $b_1 = a_1$, $b_i = a_i - a_{i-1}$ для $i = 2, \dots, n$, то есть это разница между соседними элементами исходного массива, сложность $O(n)$.

2) Для каждого запроса add(l,r,x) сделаем следующее:

   - $b_l \mathrel{+}= x$.
   - Если $r < n$, то $b_{r+1} \mathrel{-}= x$.

   Замечание. *Обработка каждого запроса делает за константное время $O(1)$, тогда для m запросов будет $O(m)$*

3) После обработки запросов будем выводить исходный массив по следующей схеме: $a_i = a_{i-1} + b_i$, а вот эта штука уже делается для каждого $i = 1, \dots, n \implies$ сложность $O(n)$ для вывода. Результирующая сложность $O(n + m)$

Замечание. *Почему a_i выведенная таким образом будет является корректным? Рассмотрим, для простоты, один запрос add(l,r,x).*

Для членов с $i < l a_i = a_{i-1} + b_i = a_{i-1} + a_i - a_{i-1} = a_i$, то есть начало массива не изменилось, как и должно быть.

Для $i = l$: $a_l = a_{l-1} + a_l - a_{l-1} + x = a_l + x$, то есть начало поданного отрезка действительно увеличилось на x, для последующих элементов $i = l, \ldots, r$ x будет уже содержаться в предыдущем элементе, а b_i не содержат добавочки в виде x, но будет компенсировать изначальное (не увеличенное на x) значение.

Для $i = r + 1 : b_{r+1} = a^{old}_{r+1} - a^{old}_r - x$, то есть для $a_{r+1} = a_r + b_{r+1} = a_r + a^{old}_{r+1} - a^{old}_r - x = a^{old}_{r+1} + \underbrace{(a_r - a^{old}_r)}_{x} - x = a^{old}_{r+1}$, таким образом, мы подавили вклад x для a_{r+1} элемента, то есть элемент не изменился, и последующие элементы не содержат x.