Домашнее задание 1

Христолюбов Максим, 771

Задание 5[Шень 1.3.2]

Фиксируем $y[j], j = 1$ и перебираем $x[i]$, начиная с $i = 1$, пока не будет выполнятся $i = k_1, x[k_1] = y[1]$.

Далее перебираем $x[i]$, начиная с $i = k_1 + 1$, пока не будет выполнятся $i = k_2, x[k_2] = y[j], j = 2$

Повторяем эти действия для всех $y[1]$, либо пока не будет выполнятся равенство $i = n$. Если по окончании действия цикла, если $j = k + 1$ (то есть для всех $y[j]$ до $i = k$ включительно нашлись равные им $x[i]$), тогда $y[j]$ – подпоследовательность $x[i]$, иначе – нет.

Каждый $x[i]$ сравнивается один раз, счетчики i,j пробегают от 1 до n и k соответственно, значит кол-во операций линейно зависит от n и k, и сложность алгоритма $O(n + k)$.

Задание 1

а) Да, верно, т. к. $n \leq n \log n \ \forall n \in \mathbb{N}$.

б) Да, верно. Рассмотрим предел $\lim\limits_{n\to\infty} \frac{n \ln n}{n^{1+\varepsilon}} = \lim\limits_{n\to\infty} \frac{\ln n}{n^{\varepsilon}} = \lim\limits_{n\to\infty} \frac{\frac{1}{n}}{\varepsilon n^{\varepsilon - 1}} = \lim\limits_{n\to\infty} \frac{1}{n^{\varepsilon}} = 0$ по правилу Лопиталя, значит, $n \ln(n) = O\left(n^{1+\varepsilon}\right)$

Задание 2

1.

а) Да, возможно, например, $g(n) = \frac{n}{\log n}, f(n) = n^2, h(n) = \frac{n^2 \log n}{n} = n \log n = \Theta(n \log n)$

б) Нет, не возможно.

$\lim\limits_{n\to\infty} \frac{f(n)}{g(n)n^3} = \lim\limits_{n\to\infty} \frac{f(n)}{n^2} \cdot \lim\limits_{n\to\infty} \frac{1}{g(n)} \cdot \lim\limits_{n\to\infty} \frac{1}{n} = 0 \cdot 0 \cdot 0 = 0$. Эти пределы равны 0 из того, что $f(n) = O\left(n^2\right), g(n) = \Omega(1)$

2.

$h(n) = O\left(n^2\right)$, при $g(n) = 1,\ f(n) = n^2$
$h(n) = \Omega\left(\frac{1}{n^C}\right)\ \forall C > 0$, при $g(n) = 1, f(n) = \frac{1}{n^C}$

Задание 3

а) При вводе нового элемента добавляем 1 в счетчик общего числа элементов ($O(n)$ операций), и прибавляем значение нового элемента к хранящемуся в памяти будущей сумме элементов, которая изначально равна 0 ($O(n)$ операций). После вводе всех чисел частное суммы и общего числа элементов будет ответом.

б) При вводе нового числа оно сравнивается с значением max (изначально равно первому элементу). Если оно равно ему, то добавляем 1 ($O(n)$ операций) в счетчик элементов равных максимальному (изначально счетчик 1), если оно больше max, то сбрасываем счетчик ($O(n)$ операций) и записываем значение элемента в max ($O(n)$ операций), если оно меньше - просто переходим к следующему элементу. Значение счетчика - ответ.

в) Запоминаем предыдущий элемент. Если новый элемент равен ему, то добавляем 1 ($O(n)$ операций) в счетчик последних равных элементов идущих подряд i (изначально счетчик $i = 1$). После этого сравниваем i с max длинной наибольшей серии ($O(n)$ операций), если $i > max$, то $max = i$. Если новый элемент не равен предыдущему, то счетчик i сбрасывается. max - ответ.

Задание 4

Будем проходить по 3 массивам с помощью 3 индексов - i,j,k, изначально равные 1, тогда $x[1], y[1], z[1]$ - первые элементы массивов.

Сначала сравним между собой $x[1], y[1], z[1]$, и запишем наименьший из них в переменную p, увеличим счетчик различных чисел на 1 (изначально 0). Будем увеличивать на 1 каждый из индексов i,j,k, пока не будут выполнятся неравенства $x[1] > p, y[1] > p, z[1] > p$.

Тогда еще раз сравним $x[1], y[1], z[1]$, и запишем наименьший из них в переменную p, увеличим счетчик различных чисел на 1. Будем продолжать эти действия пока i,j,k не станут равняться кол-ву элементов в соответствующих массивах.

Счетчик различных чисел в итоге даст ответ.

В алгоритме индексы i,j,k пробегают все элементы своих массивов по одному разу, сравнивая их на каждом шаге с p и иногда сравнивая $x[1], y[1], z[1]$, записывая наименьший из них в переменную p и увеличивая счетчик различных чисел на 1. Кол-во всех этих действий линейно зависит от размеров массивов, значит сложность алгоритма $O(n)$.

Т.к счетчик срабатывает всегда когда изменяется p, а p принимает всевозможные значения из массивов без повторений (в силу упорядоченности массивов), то алгоритм работает корректно.

Задание 5

а)

Будем составлять массив $p[i]$ следующим образом: на i-том месте в массиве будет стоять длина наибольшей возрастающей подпоследовательности чисел исходного массива, заканчивающейся i-ым элементом. Элементы $p[i]$ будут находится так: если перед i-тым элементом нет чисел меньше его, то $p[i] = 1$, т. к. единственная возрастающая подпоследовательность состоит из одного i-ого числа исходного массива. Если существуют числа меньшие его, то $p[i] = max(p[k_j]) + 1$, где k_j- номера элементов, которые меньше i-ого элемента.

Действительно, подпоследовательности, оканчивающиеся на i-ый элемент - это подпоследовательности, в которых перед i-ым числом стоит число, меньшее его, длины p[k] с дописанным в конце i-ым числом.

Для того чтобы заполнить $p[i]$ необходим сравнить i-ый элемент со всеми предыдущими, что займет $O(n)$, длина массива $p[i]$ - n, значит сложность алгоритма - $O\left(n^2\right)$).