Основные алгоритмы 1

Ковалев Алексей

10 февраля 2022 г.

1. (a) Алгоритм выведет все простые числа, меньшие или равные n, в порядке возрастания.

   (b) Алгоритм идет по массиву, пока не встретит простое число, а после этого продолжает идти по массиву до конца, проверяя делимость и меняя нули на единицы (эти операции делаются за $O(1)$). То есть алгоритм делает $\Theta(n)$ итераций на каждом простом числе. Пользуясь теоремой о распределении простых чисел, имеем
$$\pi(x) \sim \frac{x}{\ln x},$$
   где $\pi(x)$ – функция распределения простых чисел, получаем, что временная сложность алгоритма есть $\Theta\left(\frac{n^2}{\log n}\right)$.

   (c) Алгоритм не является полиномиальным, так как длина входа $k = \lceil \log_{10} n \rceil + 1$, а $\frac{n^2}{\log n}$ не является полномом от k.

2.
$$g(n) = \sum_{k=0}^{n} c^k$$

   Пользуясь формулой суммы геометрической прогрессии получаем:
$$g(n) = \frac{1 - c^{n+1}}{1 - c},\ c \neq 1$$

   (a) $c < 1$:
$$1 \leq 1 - c^{n+1} \leq g(n) \Rightarrow g(n) = \Omega(1)$$
$$g(n) = \frac{1 - c^{n+1}}{1 - c} \leq \frac{1}{1 - c} \Rightarrow g(n) = O(1)$$
$$g(n) = \Omega(1) \wedge g(n) = O(1) \Rightarrow g(n) = \Theta(1) \quad \square$$

   (b) $c = 1$:
$$g(n) = \sum_{k=0}^{n} c^k = \sum_{k=0}^{n} 1 = n + 1 \Rightarrow f(n) = \Theta(n) \quad \square$$

   (c) $c > 1$:
$$g(n) = \frac{c^{n+1} - 1}{c - 1} \geq \frac{c^{n+1}}{c - 1} = c^n \frac{c}{c - 1} \Rightarrow g(n) = \Omega(c^n)$$
$$g(n) = \frac{c^{n+1} - 1}{c - 1} \leq c^{n+1} = c \cdot c^n \Rightarrow g(n) = O(c^n)$$
$$g(n) = \Omega(c^n) \wedge g(n) = O(c^n) \Rightarrow g(n) = \Theta(c^n) \quad \square$$

3. (a) $n = O(n \log n)$ – верно.
   Пусть x – основание логарифма. Тогда $\forall n \geq x \ \log_x n \geq 1 \Rightarrow n \log_x n \geq n$.
   То есть $\exists C = 1 \ \exists N = x\colon \ \forall n \geq N \ n \leq Cn \log n \Rightarrow n = O(n \log n) \quad \square$

   (b) $\exists \varepsilon > 0\colon \ n \log n = \Omega(n^{1+\varepsilon})$ – неверно.
   Предположим, что утвеждение верно. Тогда $\exists C \ \exists N\colon \ \forall n \geq N \ n \log n \geq Cn^{1+\varepsilon} \Rightarrow \log n \geq Cn^{\varepsilon}$.
   Продифференцируем это неравенство:
$$\frac{1}{n} \geq C\varepsilon \frac{n^{\varepsilon}}{n}$$
$$1 \geq C\varepsilon n^{\varepsilon}$$
   Но $\exists N$: $\forall n \geq N \ C\varepsilon n^{\varepsilon} \geq 1$ – противоречие. $\Rightarrow n \log n \neq \Omega(n^{1+\varepsilon}) \quad \square$

4. (a) $h(n) = \Theta(n \log n)$ – возможно, например при $f(n) = n^2$; $g(n) = \frac{n}{\log n}$

   (b) $h(n) = \Theta(n^3)$ – невозможно, так как $h(n) = \frac{f(n)}{g(n)}$ максимальна при $f(n) = O\left(n^2\right)$; $g(n) = \Omega(1)$ и равна $h(n) = O\left(n^2\right)$

   (c) Как было сказано выше, лучшей верхней оценкой является $O\left(n^2\right)$, лучшей нижний оценки привести нельзя, поскольку $f(n)$ может быть сколь угодно малой.

5.

$$g(n) = \sum_{k=1}^{\lceil \log_2 n \rceil} \sum_{m=0}^{2^k} \left(\left\lceil \frac{n}{2} \right\rceil + \lceil \log_2 n \rceil \right) = \sum_{k=1}^{\lceil \log_2 n \rceil} \sum_{m=0}^{2^k} (\Theta(n) + \Theta(\log n)) =$$
$$= (\Theta(\log n) + \Theta(n)) \cdot (\Theta(\log n) + \Theta(n)) = \Theta\left(n^2\right)$$

   Первая сигма соответсвует циклу по bound, вторая сигма соответсвует циклу по i.

7. Заведем три переменные, изначально равные 0, которые будут отвечать за индекс рассматриваемых в данный момент элементов в массивах (одна переменная отвечает за один массив). Также заведем переменуую res, в которой будем хранить количество различных элементов в объединении массивов (она также изначально равна 0). На каждом шаге будем сравнивать значения элементов массивов с индексами, которые мы храним, а именно находить минимум из трех элементов. Важно, что среди трех элементов может быть несколько равных минимуму. В тех массивах, в которых значение рассматриваемых элементов минимально, переходим к следующему элементу, то есть увеличиваем на 1 индекс, отвечающий за этот массив. При этом ровно 1 раз увеличиваем res. Если какой-то индекс стал больше или равен, чем количество элементов в соответсвующем ему массиве, просто перестаем далее учитывать этот массив. Если все индексы стали больше или равны, чем количество элементов в соответствующих массивах, то алгоритм заканчивает работу.
   Доказательство корректности: алгоритмы закончит работу, когда все массивы будут пройдены. Все массивы будут пройдены, так как на каждом шаге мы увеличиваем индексы в массивах, где текущий элемент меньше или равен остальных, а среди любого множества элементов всегда найдется хотя бы один минимальный. После работы алгоритма в res будет ответ, так как массивы изначально отсортированы, и мы увеличивали значение этой переменной всегда, когда встречали новый элемент.
   Оценка сложности: мы линейно проходим по каждому из массивов, выполняя за каждый шаг константное число сравнений и изменений переменных, то есть если длина самого длинного из них n, то время работы – $\Theta(n)$. Мы дополнительно храним всего 4 переменные, а занчит затраты памяти – $\Theta(1)$ (если считать, что массивы нам даны и их не надо считывать; если же считать, что нам требуется сначала считать массивы, то потребуется $\Theta(n)$ памяти).