АиСД 2021. Первый семестр

Задания на практики M3131-M3133

⟨Версия от 17 сентября 2021 г.⟩

Темы

1 Асимптотики **2**

1.1 Практика 2

1.2 Письменное ДЗ 3

2 Сортировки **4**

2.1 Практика 4

Неделя 1. Асимптотики

> **Def.** Напомним используемые далее определения:
>
> ▷ $f(n) = \mathcal{O}(g(n)) \;\equiv\; \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \leqslant c \cdot g(n)$
>
> ▷ $f(n) = \Omega(g(n)) \;\equiv\; \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \geqslant c \cdot g(n)$
>
> ▷ $f(n) = \Theta(g(n)) \;\equiv\; \exists n_0, c_1, c_2 > 0 : \forall n \geqslant n_0 :$
> $c_1 \cdot g(n) \leqslant f(n) \leqslant c_2 \cdot g(n)$

Практика

1.1. Для каждой из рассматриваемых дальше функций $f(n)$ найдите наиболее компактно записываемую $g(n)$, что $f(n) = \Theta(g(n))$, и докажите это соотношение.

(a) $f(n) = 7n^2 - 7(n-3)^2$
(b) $f(n) = 5n + 2\sqrt[3]{n}$
(c) $f(n) = 10(n+1)^2 + 3(n-2)$
(d) $f(n) = \log(\sqrt{n}) + \sqrt{\log n}$
(e) $f(n) = n \cdot 3^{n+1} + n^{10}$
(f) $f(n) = \frac{10n^2+2}{7n-1}$
(g) $f(n) = \log(2n \log n)$

1.2. Для следующих пар функций $f(n)$ и $g(n)$ покажите, верно ли, что $f(n) = \mathcal{O}(g(n))$ и докажите свой ответ.

(a) $f(n) = \log n$
$g(n) = \sqrt{n}$
(b) $f(n) = n\sqrt{n}\log(n)$
$g(n) = n\log^3 n$
(c) $f(n) = n$
$g(n) = (\log n)^{\log n}$

1.3. Время работы некоторого алгоритма задано следующим рекуррентным соотношением. Найдите Θ-асимптотику времени работы этого алгоритма, *построив дерево рекурсивных вызовов.*

(a) $T(n) = T(n-1) + 2n^2$
(b) $T(n) = T\left(\frac{n}{2}\right) + n$
(c) $T(n) = 3T\left(\frac{n}{2}\right) + 2$
(d) $T(n) = T\left(\frac{n}{3}\right) + \log n$

1.4. Докажите следующие утверждения *по индукции.* В качестве базы во всех пунктах можно считать, что $T(1) = 1$.

(a) Если $T(n) = 2T(\sqrt{n}) + 1$, то $T(n) = \mathcal{O}(\log n)$
(b) Если $T(n) = 2T\left(\frac{n}{2}\right) + n$, то $T(n) = \mathcal{O}(n \log n)$
(c) Если $T(n) = 2T\left(\frac{n}{2}\right) + n$, то $T(n) = \Omega(n \log n)$
(d) Если $T(n) = 3T\left(\frac{n}{2}\right) + 1$, то $T(n) = \Omega(n)$

1.5. Докажите, что $\sum\limits_{t=1}^{n} \frac{1}{t} = \Omega(\log n)$.

1.6. Докажите или опровергните следующие утверждения. В качестве опровержения достаточно привести контрпример и показать, почему для него утверждение не выполняется.

(a) Если $f(n) = \mathcal{O}(g(n))$, то $f^2(n) = \mathcal{O}\big(g^2(n)\big)$

(b) Если $f(n) = \mathcal{O}(g(n))$, то $2^{f(n)} = \mathcal{O}\big(2^{g(n)}\big)$

(c) Если $f(n) = \mathcal{O}(g(n))$, то $\log f(n) = \mathcal{O}(\log g(n))$

Письменное ДЗ

★ Правила сдачи

- ▷ Письменное ДЗ оформляется в LaTeX, написанные от руки решения приниматься не будут
- ▷ Решения надо прислать **до 24 сентября 2021, 23:59** на почту `itmo.algo.teaching+y2021@gmail.com`
- ▷ Тема письма должна быть указана в виде «`Группа HW01 Фамилия Имя`» (без кавычек), например: «`M3130 HW01 Иванов Иван`». Обратите внимание, что буква «M» в названии группы – латинская
- ▷ К письму должны быть приложены как сгенерированный `.pdf`-файл, так и исходный `.tex`-файл

1. Для каждой из приведенных ниже программ найдите и аргументируйте точную $\mathcal{O}$-асимптотику времени ее работы.

(a)
```
for i = 0..n:
    for j = 0..i:
        for k = 0..j:
            print(i, j, k)
```

(b)
```
for i = 0..n:
    j = 0, k = 2 * i
    while j < k:
        j++, k--
```

(c)
```
i = 1
while i < n:
    for j = 0..i:
        print(i, j)
    i = i * 2
```

(d)
```
for i = 0..n:
    j = i
    while j > 0:
        j = j / 2
```

2. Докажите следующие соотношения по определению (выберите константы c и n_0 и докажите соответствующее неравенство).

(a) $\log n = \Omega(20)$

(b) $2^n = \mathcal{O}(3^n)$

(c) $n(n-8) = \Omega\big(n^2\big)$

(d) $3n + 2\sqrt{n} = \mathcal{O}(n \log n)$

(e) $n! = \Omega(5^n)$

3. Докажите, что если $f(n) = \mathcal{O}(h(n))$ и $g(n) = \mathcal{O}(h(n))$, то $f(n) + g(n) = \mathcal{O}(h(n))$.

4. Докажите по индукции, что если $T(n) = 2T\big(\frac{n}{4}\big) + \log_2 n$, то $T(n) = \mathcal{O}\big(\log^2 n\big)$.

Неделя 2. Сортировки

> Во всех следующих задачах, если не указано обратное, n по умолчанию будет означать длину рассматриваемого в задаче массива. Если массивов несколько, за n обозначается сумма их длин.
>
> Эталонные реализации квадратичных сортировок, на которые стоит опираться при решении задач этой недели, можно найти по ссылке.

Практика

2.1. Для следующих массивов выпишите их состояния после четвертой (нумерация с одного, `i = 0` – это первая итерация) итерации внешнего цикла сортировки выбором и сортировки вставками.

(a) $[\mathtt{5}, \mathtt{2}, \mathtt{7}, \mathtt{1}, \mathtt{6}, \mathtt{4}, \mathtt{3}]$

(b) $[\mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{7}, \mathtt{6}, \mathtt{5}, \mathtt{1}]$

(c) $[\mathtt{7}, \mathtt{6}, \mathtt{5}, \mathtt{4}, \mathtt{3}, \mathtt{2}, \mathtt{1}]$

2.2. В отсортированный массив в произвольное место вставили случайное число. Придумайте алгоритм, который за $\mathcal{O}(n)$ сортирует полученный массив.

> **Def.** Сортировка называется *стабильной*, если порядок одинаковых (равных) элементов массива не меняется после сортировки.

2.3. Для каждой из следующих сортировок покажите, является ли она стабильной. Если это зависит от реализации, покажите как именно реализовать стабильный вариант выбранной сортировки.

(a) сортировка выбором

(b) сортировка вставками

(c) сортировка слиянием

> **Def.** *Перестановкой* размера n называется массив размера n, в котором каждое целое число от 1 до n встречается ровно по одному разу.

2.4. Постройте для произвольного n перестановку, на которой сортировка вставками делает наибольшее число операций `swap(a[i], a[j])` (обмен элементов массива местами). Сколько в точности обменов совершается? Единственна ли такая перестановка?

2.5. Постройте для произвольного n перестановку, на которой сортировка выбором делает наибольшее число операций `a[i] < a[j]` (сравнение элементов массива). Сколько в точности сравнений совершается? Единственна ли такая перестановка?

2.6. Отсортируйте за $\mathcal{O}(n)$

(a) массив, в котором хранятся только нули и единицы

(b) перестановку размера n

(c) массив, все элементы которого – целые числа от 1 до n

2.7. Даны два отсортированных по неубыванию массива длины n. Придумайте алгоритм, который за $\mathcal{O}(n)$ определяет, есть ли в них одинаковый элемент.

2.8. Постройте для произвольного n два отсортированных массива `a` и `b`, на которых функция слияния `merge` выполняет максимальное количество сравнений `a[0] <= b[0]`. Сколько в точности сравнений совершается на таких массивах?

2.9. Модифицируйте сортировку подсчетом так, чтобы ее можно было применять к массивам, содержащим отрицательные числа.