Алгоритмы и структуры данных-1
Лекция 3

Дата: 18.09.2023

Программная инженерия, 2 курс
2023-2024 учебный год

Нестеров Р.А., PhD, ст. преподаватель
департамент программной инженерии ФКН

SET 1. Домашняя работа

Тематический модуль Инвариант цикла. Асимптотический анализ алгоритмов. Линейный контейнер

SET 1. Домашняя работа

Тематический модуль Инвариант цикла. Асимптотический анализ алгоритмов. Линейный контейнер

Блок A – Разработка, анализ корректности и сложности

Блок P – Реализация и обработка линейных контейнеров

SET 1. Домашняя работа

Тематический модуль Инвариант цикла. Асимптотический анализ алгоритмов. Линейный контейнер

Блок А – Разработка, анализ корректности и сложности

Блок Р – Реализация и обработка линейных контейнеров

18.09.2023 15:00 – 02.10.2023 22:00

План

Асимптотический анализ функции $T(n)$ временной сложности алгоритмов. Символы Ландау

Сортировка выбором: рекурсивный алгоритм

Сортировка слиянием: разделяй-и-властвуй (DaC)

Рекуррентное соотношение

Асимптотический анализ временной сложности

Порядок роста и Θ

Порядок роста и Θ

Время работы INSERTION SORT в худшем случае составляет $\Theta(n^2)$

Порядок роста и Θ

Время работы INSERTION SORT в худшем случае составляет $\Theta(n^2)$

Символ Θ обозначает асимптотически точную границу для функции временной сложности алгоритма

Порядок роста и Θ

Время работы INSERTION SORT в худшем случае составляет $\Theta(n^2)$

Символ Θ обозначает асимптотически точную границу для функции временной сложности алгоритма

$$\Theta\big(g(n)\big) = \{f(n) | \exists c_1, c_2, n_0 > 0 \forall n \geq n_0 : c_1 g(n) \leq g(n) \leq c_2 g(n)\}$$

Порядок роста и Θ

Время работы INSERTION SORT в худшем случае составляет $\Theta(n^2)$

Символ Θ обозначает асимптотически точную границу для функции временной сложности алгоритма

$$\Theta\big(g(n)\big) = \{f(n) | \exists c_1, c_2, n_0 > 0 \forall n \geq n_0 : c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Θ определяет множество функций

$$T(n) \in \Theta\big(g(n)\big) \Leftrightarrow T(n) = \Theta(g(n)$$

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$.

По определению, $\mathrm{c}_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$.

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$.

По определению, $c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$.

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$.

По определению, $c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2$.

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

$$\boldsymbol{c_1 \leq \frac{1}{2} - \frac{3}{n}, c_1 > 0}$$

$$\boldsymbol{c_2 \geq \frac{1}{2} - \frac{3}{n}, c_2 > 0}$$

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$.

По определению, $c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$.

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

$$\boldsymbol{c_1 \le \frac{1}{2} - \frac{3}{n}, c_1 > 0}$$

$$n \ge 7, c_1 \le \frac{1}{14}$$

$$\boldsymbol{c_2 \ge \frac{1}{2} - \frac{3}{n}, c_2 > 0}$$

$$n \ge 1, c_2 \ge \frac{1}{2}$$

Порядок роста и Θ

Показать, что $T(n) = \frac{1}{2}n^2 - 3n = \Theta(n^2)$.

По определению, $c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$.

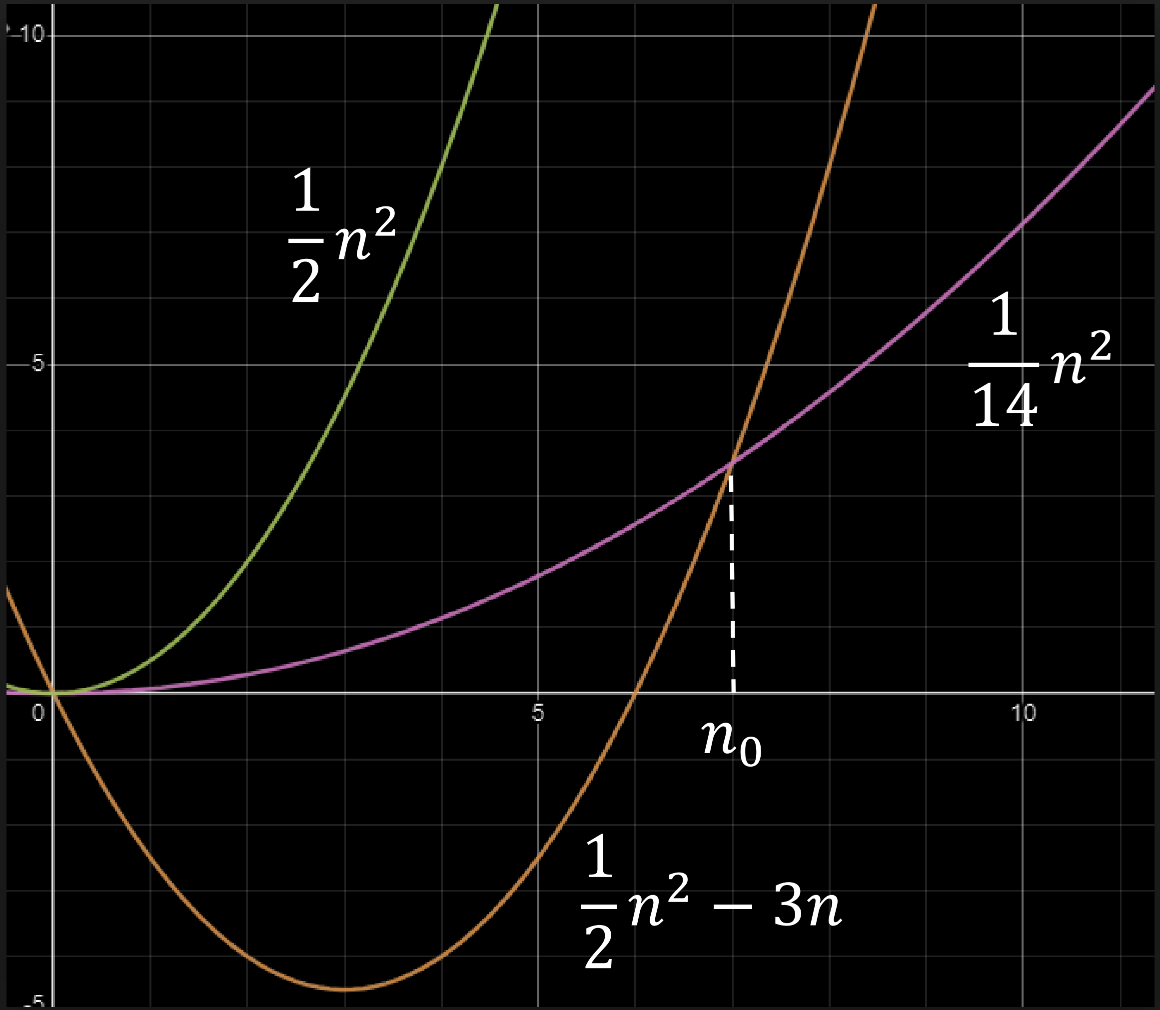
$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

$$\boldsymbol{c_1 \le \frac{1}{2} - \frac{3}{n}, c_1 > 0} \qquad \boldsymbol{c_2 \ge \frac{1}{2} - \frac{3}{n}, c_2 > 0}$$

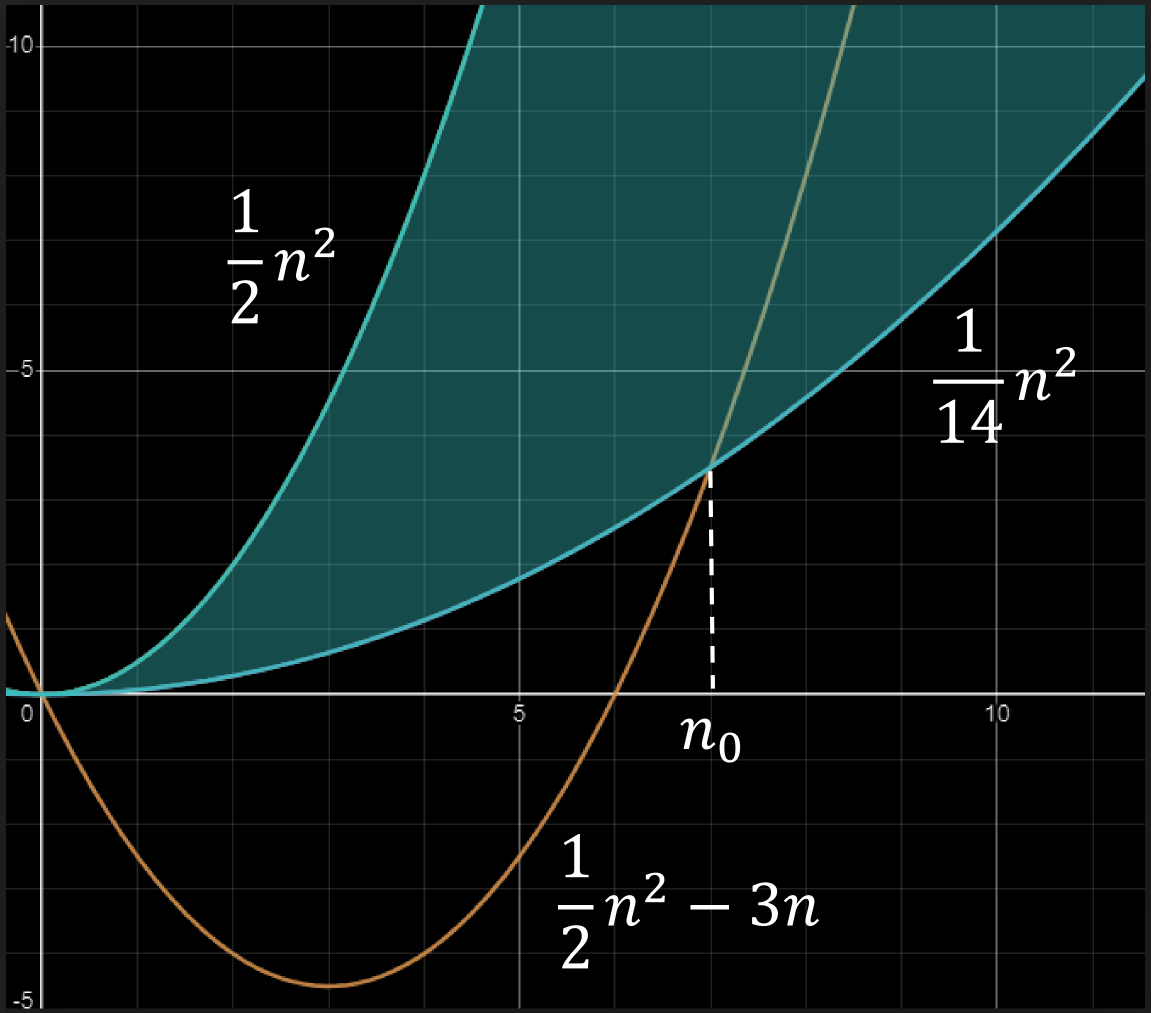
$$n \ge 7, c_1 \le \frac{1}{14} \qquad n \ge 1, c_2 \ge \frac{1}{2}$$

$$n_0 = 7, c_1 = \frac{1}{14}, c_2 = \frac{1}{2}$$

Порядок роста и Θ



Порядок роста и Θ



Порядок роста и Θ

Повысить или понизить порядок роста при анализе асимптотически точной границы нельзя!

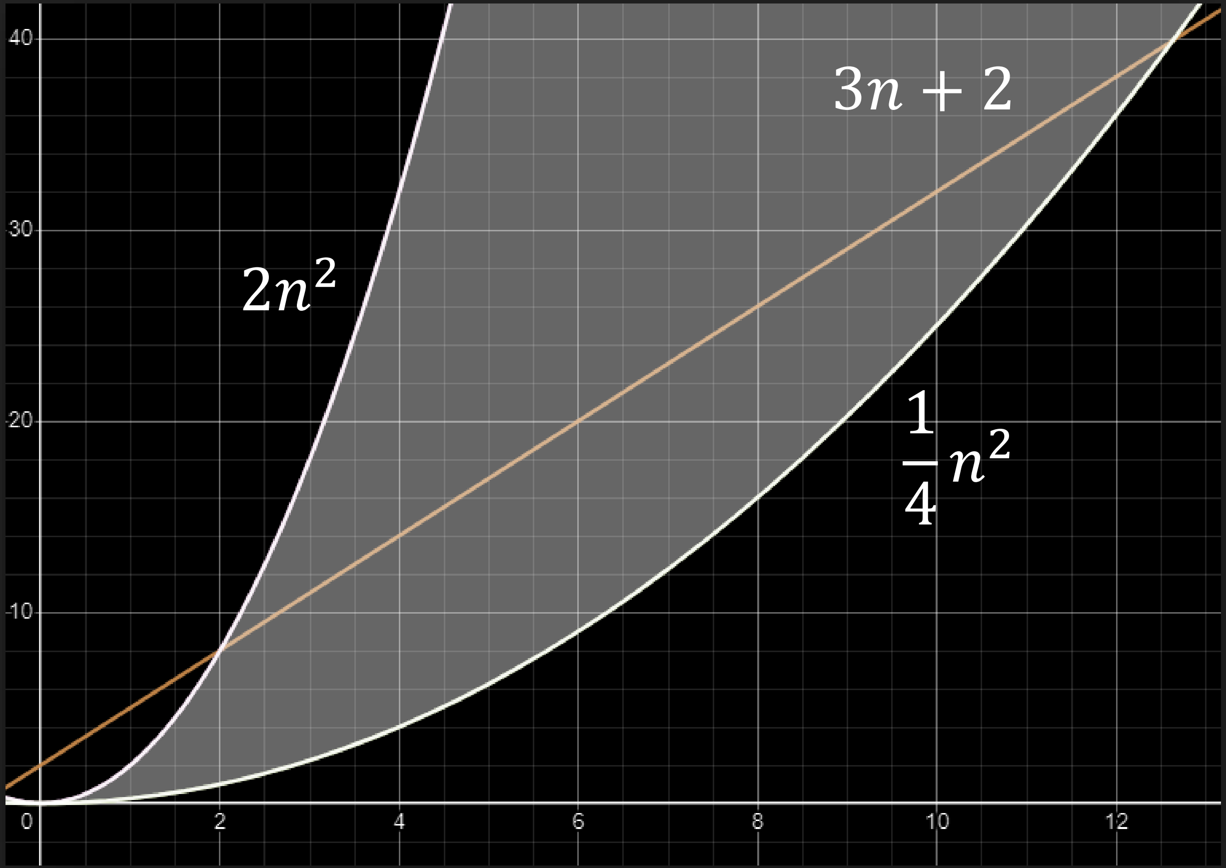
Порядок роста и Θ

Повысить или понизить порядок роста при анализе асимптотически точной границы нельзя!

Например, $T(n) = 3n + 2 \neq \Theta(n^2)$

Порядок роста и Θ

Например, $T(n) = 3n + 2 \neq \Theta(n^2)$



Верхняя граница и O

Символ O обозначает асимптотическую верхнюю границу функции временной сложности алгоритма

Верхняя граница и O

Символ O обозначает асимптотическую верхнюю границу функции временной сложности алгоритма

Фактически, соответствует худшему случаю работы алгоритма

Верхняя граница и O

Символ O обозначает асимптотическую верхнюю границу функции временной сложности алгоритма

Фактически, соответствует худшему случаю работы алгоритма

$$O\big(g(n)\big) = \{f(n) | \exists c, n_0 > 0 \forall n \geq n_0 : f(n) \leq cg(n)\}$$

Верхняя граница и *O*

```
int sum = 0;
for ( size_t i = 0; i < n; ++i )
    for ( size_t j = 0; j < i; ++j )
        for ( size_t k = 0; k < j; ++k )
            sum += i + j + k;
```

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	
`    for ( size_t j = 0; j < i; ++j )`	
`        for ( size_t k = 0; k < j; ++k )`	
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	
`    for ( size_t j = 0; j < i; ++j )`	
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j \cdot c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2 + i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2 + i}{2} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2 + i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2+i}{2} c_1 = \frac{n^3+n^2+n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2+i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2+i}{2} c_1 = \frac{n^3+n^2+n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2+i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2+i}{2} c_1 = \frac{n^3+n^2+n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2+i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	c_1

$$T(n) = \frac{1}{4} c_1 (n^3 + n^2 + n) + 1$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2+i}{2} c_1 = \frac{n^3+n^2+n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2+i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$j \cdot c_1$
`            sum += i + j + k;`	$\Theta(1)$

$$T(n) = \frac{1}{4} c_1 (n^3 + n^2 + n) + 1$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2+i}{2} c_1 = \frac{n^3+n^2+n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\sum_{j=0}^{i} j\, c_1 = \frac{i^2+i}{2} c_1$
`        for ( size_t k = 0; k < j; ++k )`	$\Theta(j)$
`            sum += i + j + k;`	$\Theta(1)$

$$T(n) = \frac{1}{4} c_1 (n^3 + n^2 + n) + 1$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\sum_{i=0}^{n} \frac{i^2 + i}{2} c_1 = \frac{n^3 + n^2 + n}{4} c_1$
`    for ( size_t j = 0; j < i; ++j )`	$\Theta(i^2)$
`        for ( size_t k = 0; k < j; ++k )`	$\Theta(j)$
`            sum += i + j + k;`	$\Theta(1)$

$$T(n) = \frac{1}{4} c_1 (n^3 + n^2 + n) + 1$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\Theta(n^3)$
`    for ( size_t j = 0; j < i; ++j )`	$\Theta(i^2)$
`        for ( size_t k = 0; k < j; ++k )`	$\Theta(j)$
`            sum += i + j + k;`	$\Theta(1)$

$$T(n) = \frac{1}{4}c_1(n^3 + n^2 + n) + 1$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$\Theta(n^3)$
`    for ( size_t j = 0; j < i; ++j )`	$\Theta(i^2)$
`        for ( size_t k = 0; k < j; ++k )`	$\Theta(j)$
`            sum += i + j + k;`	$\Theta(1)$

$$T(n) = \frac{1}{4}c_1(n^3 + n^2 + n) + 1 = \Theta(n^3)$$

Верхняя граница и O

Вложенный цикл	Время
`int sum = 0;`	1
`for ( size_t i = 0; i < n; ++i )`	$O(n^3)$
`    for ( size_t j = 0; j < i; ++j )`	$O(i^2)$
`        for ( size_t k = 0; k < j; ++k )`	$O(j)$
`            sum += i + j + k;`	$O(1)$

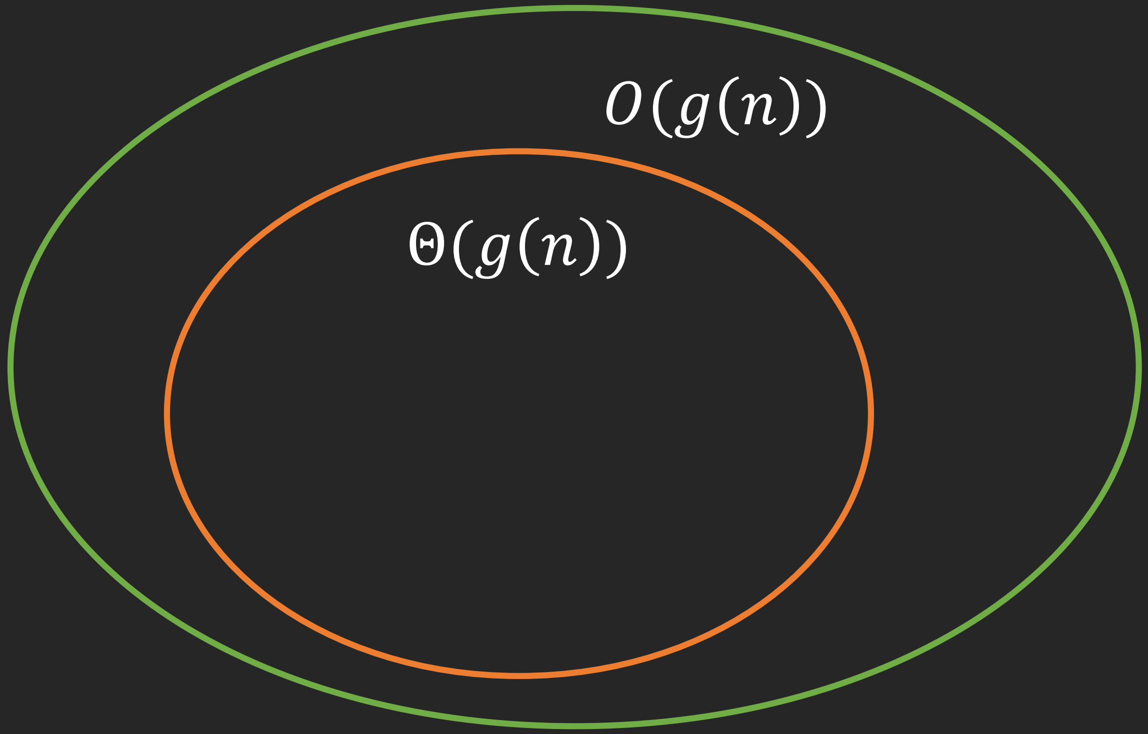
$$T(n) = \frac{1}{4}c_1(n^3 + n^2 + n) + 1 = O(n^3)$$

Верхняя граница и O

Если $T(n) = \Theta(g(n))$, то $T(n) = O(g(n))$

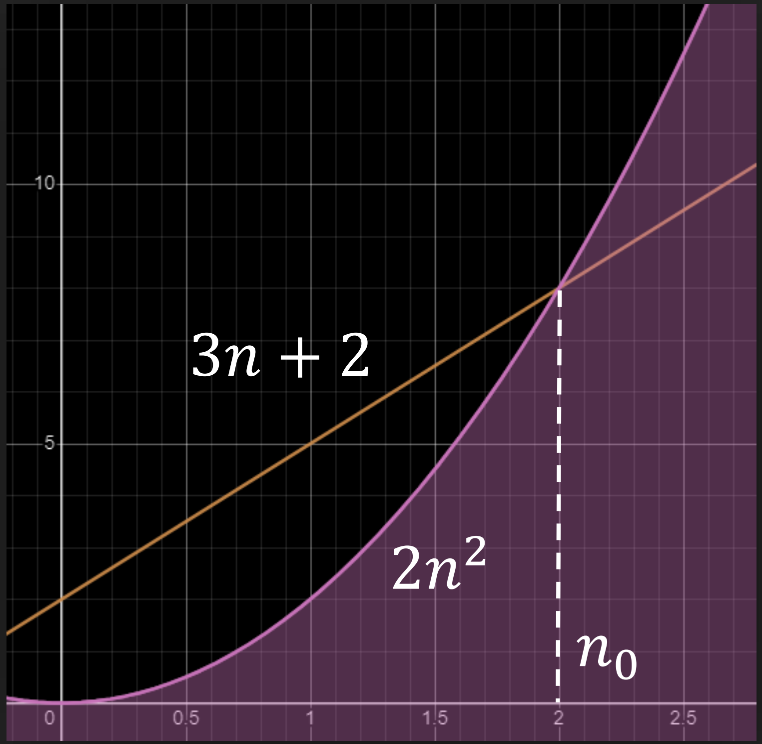
Верхняя граница и O

Если $T(n) = \Theta(g(n))$, то $T(n) = O(g(n))$



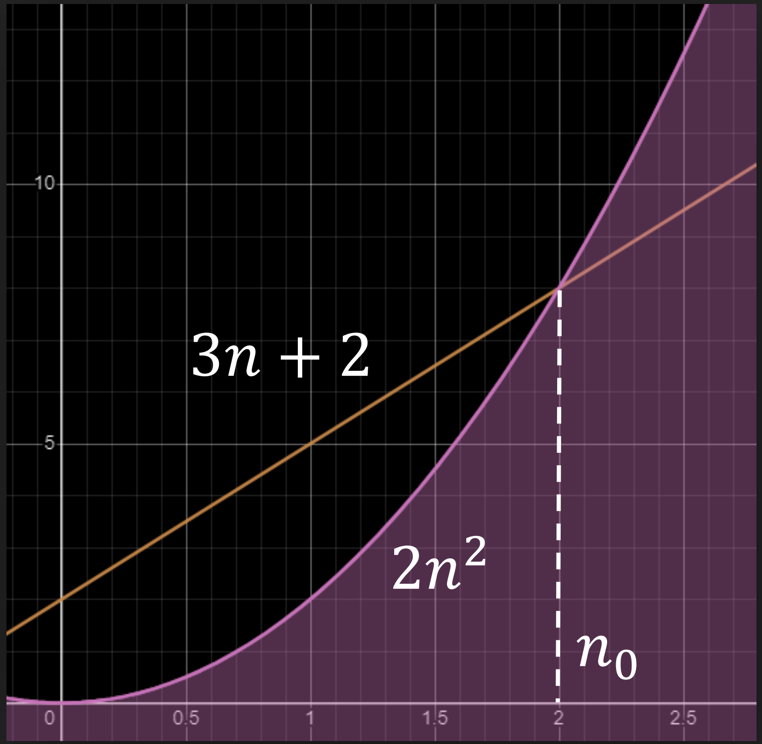
Верхняя граница и O

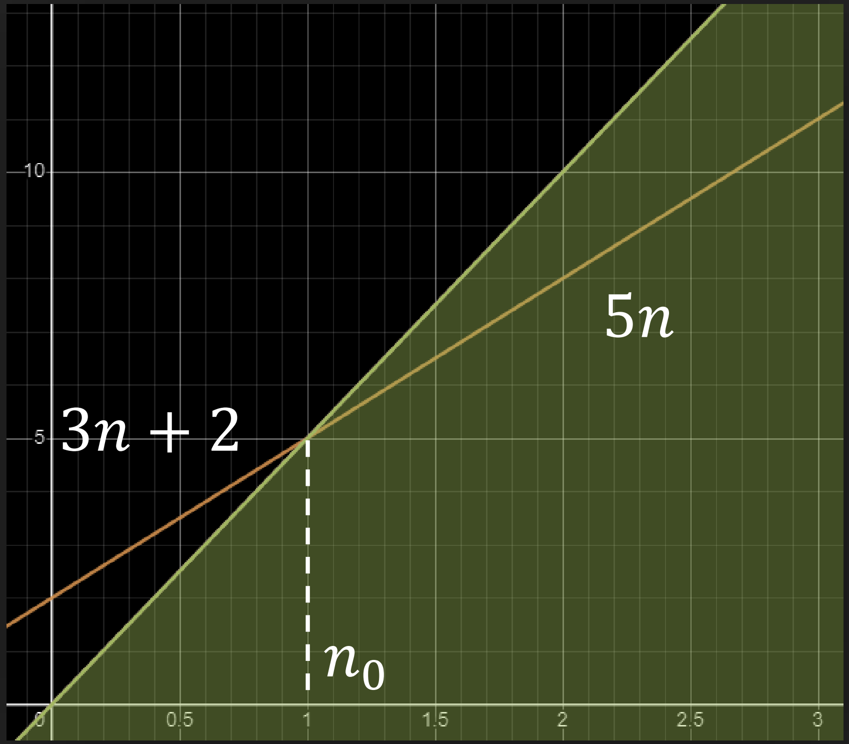
Для верхней границы временной сложности можно повысить порядок, т.е. $T(n) = 3n + 2 = O(n^2)$



Верхняя граница и O

Для верхней границы временной сложности можно повысить порядок, т.е. $T(n) = 3n + 2 = O(n^2)$





Нижняя граница и Ω

Символ Ω обозначает асимптотическую нижнюю границу функции временной сложности алгоритма

Нижняя граница и Ω

Символ Ω обозначает асимптотическую нижнюю границу функции временной сложности алгоритма

Фактически соответствует лучшему случаю работы алгоритма

Нижняя граница и Ω

Символ Ω обозначает асимптотическую нижнюю границу функции временной сложности алгоритма

Фактически соответствует лучшему случаю работы алгоритма

$T(n)$ для INSERTION SORT в лучшем случае является линейной функцией, $T(n) = \Omega(n)$

Связь символов O, Θ, Ω

$$T(n) = \Theta\big(g(n)\big) \Leftrightarrow \begin{cases} T(n) = O\big(g(n)\big) \\ T(n) = \Omega\big(g(n)\big) \end{cases}$$

SELECTION SORT и временная сложность рекурсии

Схема работы SELECTION SORT

Найти максимальный элемент в заданной последовательности

Схема работы SELECTION SORT

Найти максимальный элемент в заданной последовательности

Обменять значения последнего элемента и максимального

Схема работы SELECTION SORT

Найти максимальный элемент в заданной последовательности длины n

Обменять значения последнего элемента и максимального

Повторить для последовательности длины $n - 1$

Схема работы SELECTION SORT

```
selectionSort(A, n)
    if n <= 1 return
    pos = 0
    max = A[pos]
    for i = 0 to n
        if (A[i] > max)
            pos = i;
            max = A[pos]
    swap(A[n - 1], A[pos])
    selectionSort(A, n – 1)
```

	Время
`selectionSort(A, n)`	$T(n) = \ldots$
`    if n <= 1 return`	
`    pos = 0`	
`    max = A[pos]`	
`    for i = 0 to n`	
`        if (A[i] > max)`	
`            pos = i;`	
`            max = A[pos]`	
`    swap(A[n - 1], A[pos])`	
`    selectionSort(A, n – 1)`	

	Время
`selectionSort(A, n)`	$T(n) = \ldots$
`    if n <= 1 return`	$T(0) = T(1) = \Theta(1)$
`    pos = 0`	
`    max = A[pos]`	
`    for i = 0 to n`	
`        if (A[i] > max)`	
`            pos = i;`	
`            max = A[pos]`	
`    swap(A[n - 1], A[pos])`	
`    selectionSort(A, n – 1)`	

	Время
selectionSort(A, n)	$T(n) = \ldots$
if n <= 1 return	$T(0) = T(1) = \Theta(1)$
pos = 0	$\Theta(1)$
max = A[pos]	
for i = 0 to n	
if (A[i] > max)	
pos = i;	
max = A[pos]	
swap(A[n - 1], A[pos])	
selectionSort(A, n − 1)	

	Время
selectionSort(A, n)	$T(n) = \ldots$
if n <= 1 return	$T(0) = T(1) = \Theta(1)$
pos = 0	$\Theta(1)$
max = A[pos]	
for i = 0 to n	
if (A[i] > max)	$\Theta(1)$
pos = i;	
max = A[pos]	
swap(A[n - 1], A[pos])	
selectionSort(A, n – 1)	

	Время	
`selectionSort(A, n)`	$T(n) = \dots$	
`if n <= 1 return`	$T(0) = T(1) = \Theta(1)$	
`pos = 0`	$\Theta(1)$	
`max = A[pos]`		
`for i = 0 to n`		$\Theta(n)$
`if (A[i] > max)`	$\Theta(1)$	
`pos = i;`		
`max = A[pos]`		
`swap(A[n - 1], A[pos])`		
`selectionSort(A, n – 1)`		

<table>
<tr><th></th><th colspan="2">Время</th></tr>
<tr><td><code>selectionSort(A, n)</code></td><td colspan="2">$T(n) = \ldots$</td></tr>
<tr><td><code>if n <= 1 return</code></td><td colspan="2">$T(0) = T(1) = \Theta(1)$</td></tr>
<tr><td><code>pos = 0</code></td><td colspan="2" rowspan="2">$\Theta(1)$</td></tr>
<tr><td><code>max = A[pos]</code></td></tr>
<tr><td><code>for i = 0 to n</code></td><td></td><td rowspan="4">$\Theta(n)$</td></tr>
<tr><td><code>if (A[i] > max)</code></td><td rowspan="3">$\Theta(1)$</td></tr>
<tr><td><code>pos = i;</code></td></tr>
<tr><td><code>max = A[pos]</code></td></tr>
<tr><td><code>swap(A[n - 1], A[pos])</code></td><td colspan="2">$\Theta(1)$</td></tr>
<tr><td><code>selectionSort(A, n - 1)</code></td><td colspan="2">$T(n-1)$</td></tr>
</table>

$T(n)$ для SELECTION SORT

$$T(n) = \Theta(1) + \Theta(n) + \Theta(1) + T(n-1)$$

$T(n)$ для SELECTION SORT

$$T(n) = \Theta(1) + \Theta(n) + \Theta(1) + T(n-1)$$

$$T(n) = T(n-1) + \Theta(n)$$

$T(n)$ для SELECTION SORT

$$T(n) = \Theta(1) + \Theta(n) + \Theta(1) + T(n-1)$$

$$T(n) = T(n-1) + \Theta(n)$$

Функция временной сложности представляет собой рекуррентное соотношение

$T(n)$ для SELECTION SORT

$$T(n) = \Theta(1) + \Theta(n) + \Theta(1) + T(n-1)$$

$$T(n) = T(n-1) + \Theta(n)$$

Функция временной сложности представляет собой рекуррентное соотношение

Требуется определить верхнюю границу временной сложности

$T(n)$ для SELECTION SORT

$$T(n) = T(n-1) + \Theta(n) = T(n-1) + O(n)$$

$T(n)$ для SELECTION SORT

$$T(n) = T(n-1) + O(n) \leq T(n-1) + cn$$

$T(n)$ для SELECTION SORT

$$T(n) = T(n-1) + O(n) \leq T(n-1) + cn$$

$$\begin{aligned} T(n) &\leq T(n-1) + cn \leq T(n-2) + c(n-1) + cn \leq \\ &\leq T(n-3) + c(n-2) + c(n-1) + cn \leq \cdots \leq \\ &\leq T(1) + c(2 + 3 + \cdots + n) \end{aligned}$$

$T(n)$ для SELECTION SORT

$$T(n) = T(n-1) + O(n) \le T(n-1) + cn$$

$$T(n) \le T(n-1) + cn \le T(n-2) + c(n-1) + cn \le$$
$$\le T(n-3) + c(n-2) + c(n-1) + cn \le \cdots \le$$
$$\le T(1) + c(2 + 3 + \cdots + n) \overset{\overset{T(1)=\Theta(1)}{\downarrow}}{=} \Theta(1) + c\sum_{i=2}^{n} i =$$

$T(n)$ для SELECTION SORT

$$T(n) = T(n-1) + O(n) \leq T(n-1) + cn$$

$$T(n) \leq T(n-1) + cn \leq T(n-2) + c(n-1) + cn \leq$$
$$\leq T(n-3) + c(n-2) + c(n-1) + cn \leq \cdots \leq$$
$$\leq T(1) + c(2 + 3 + \cdots + n) = \Theta(1) + c\sum_{i=2}^{n} i =$$
$$= O(1) + O(n^2) = O(n^2)$$

$T(n)$ для SELECTION SORT

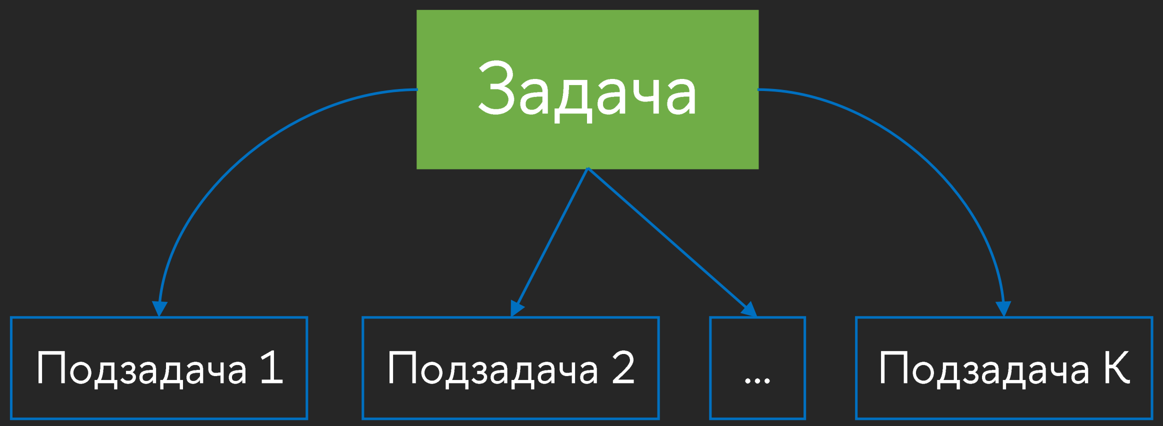
Таким образом, имеем асимптотическую верхнюю границу рекуррентного соотношения

$$T(n) = T(n-1) + \Theta(n) = O(n^2)$$

MERGE SORT: разделяй-и-властвуй

Разделяй-и-властвуй (DaC)

DIVIDE

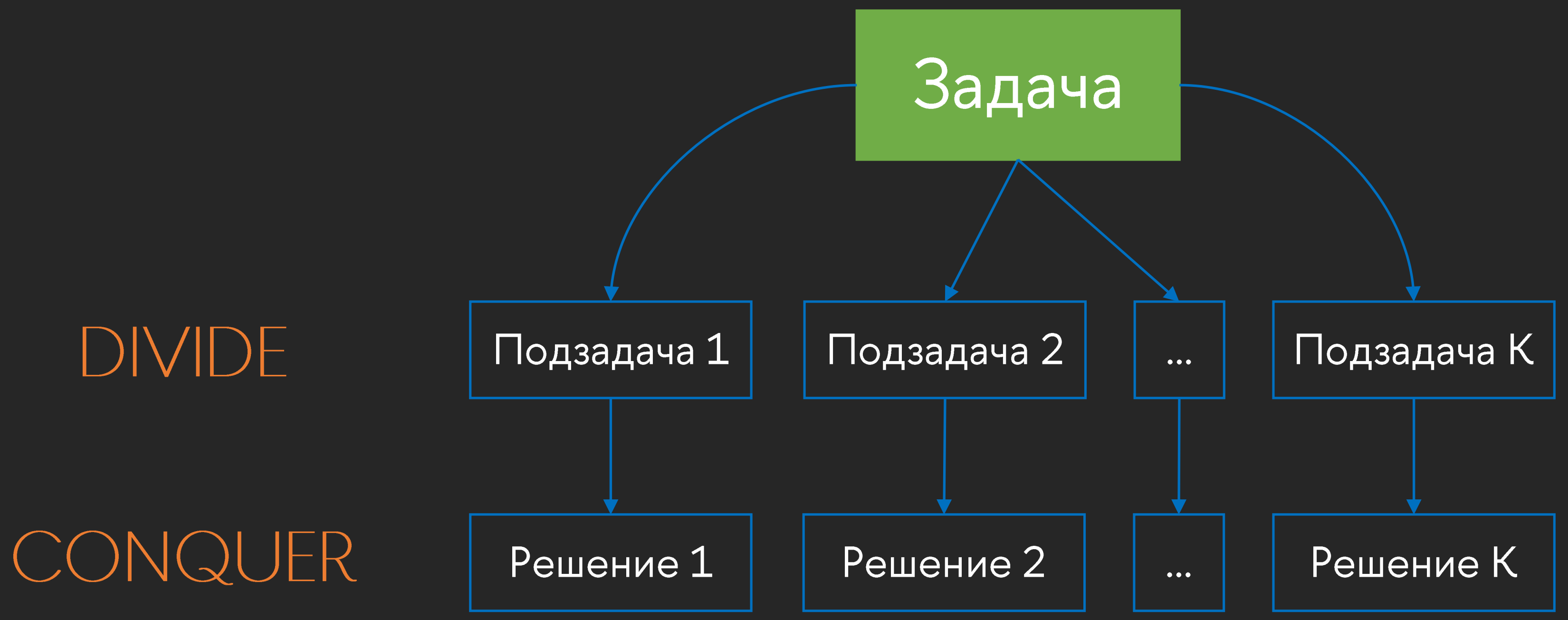


Разделяй-и-властвуй (DaC)

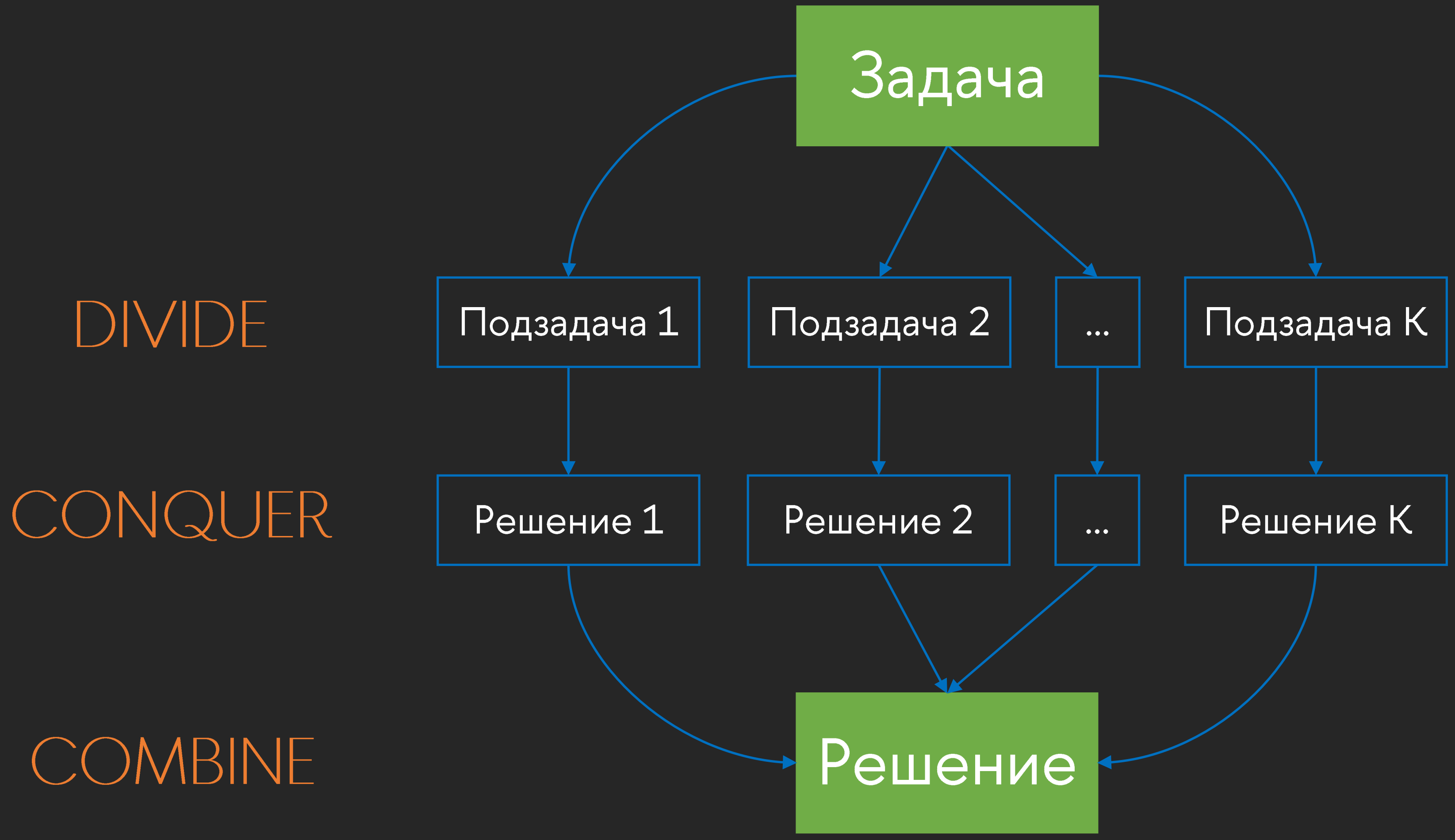
DIVIDE

Подзадача I ∩ Подзадача J = ∅

Разделяй-и-властвуй (DaC)



Разделяй-и-властвуй (DaC)



MERGE SORT

DIVIDE

Разделить последовательность из n элементов на две по $n/2$ элементов в каждой

MERGE SORT

DIVIDE	Разделить последовательность из n элементов на две по $n/2$ элементов в каждой
CONQUER	Рекурсивно отсортировать две полученные подпоследовательности

MERGE SORT

DIVIDE	Разделить последовательность из n элементов на две по $n/2$ элементов в каждой
CONQUER	Рекурсивно отсортировать две полученные подпоследовательности
COMBINE	Объединить две отсортированные последовательности

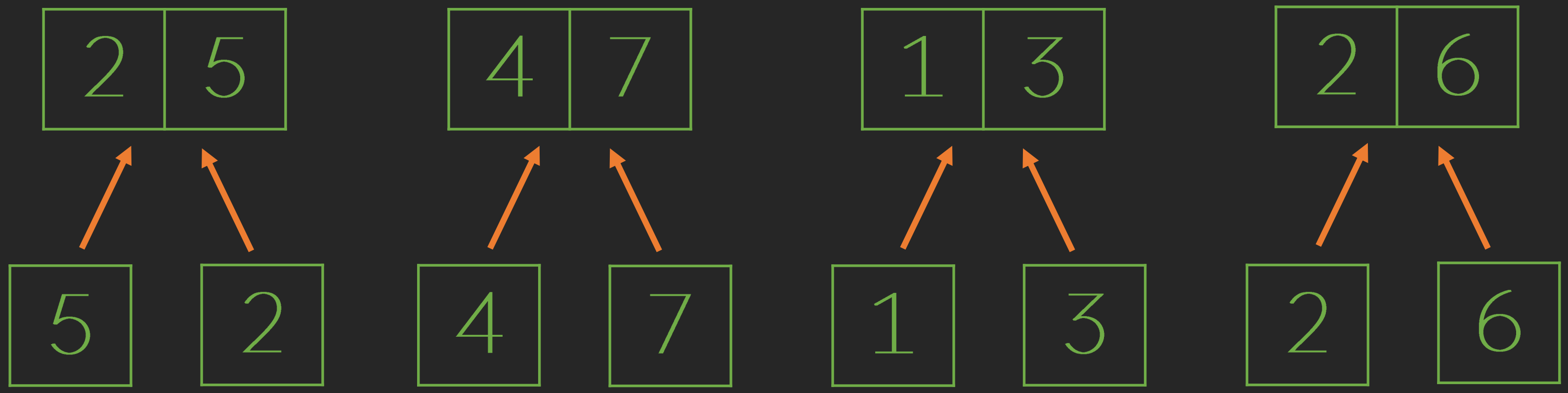
MERGE SORT

5	2	4	7	1	3	2	6

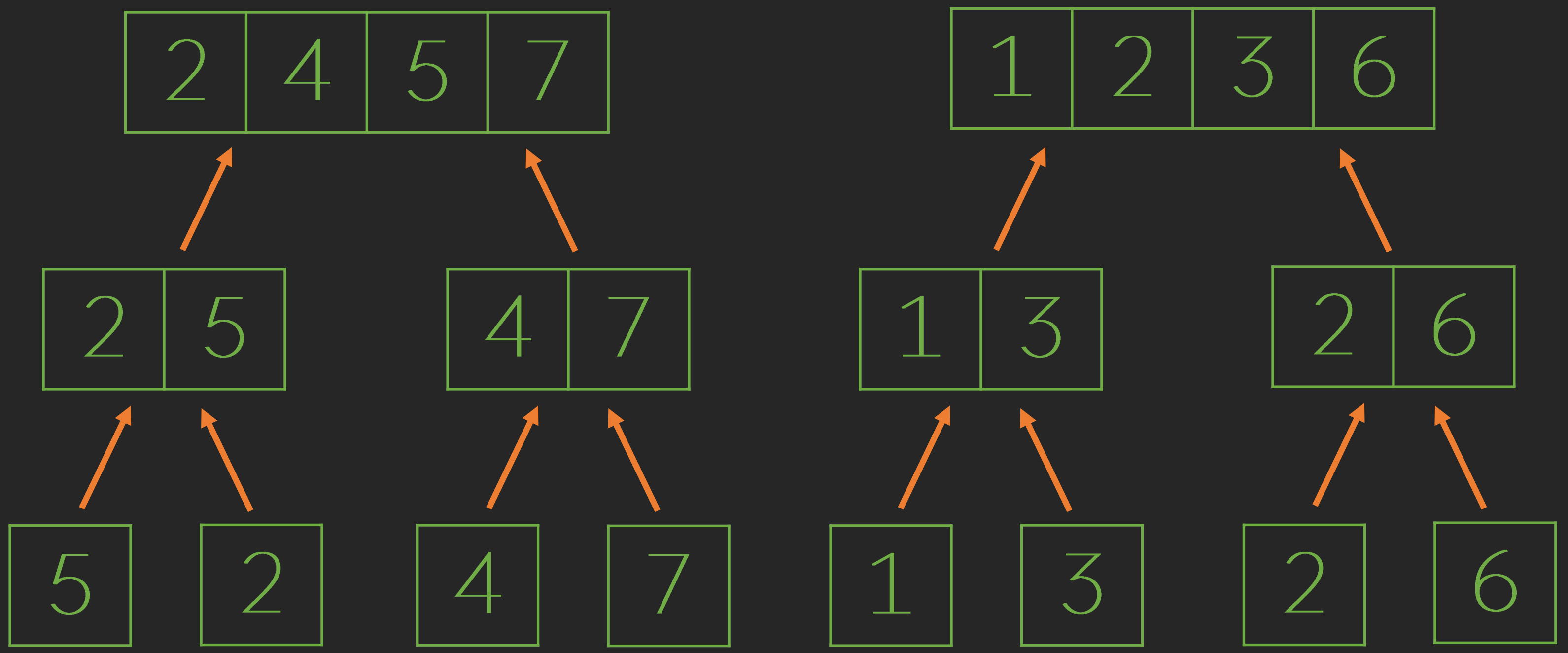
MERGE SORT

5	2	4	7	1	3	2	6

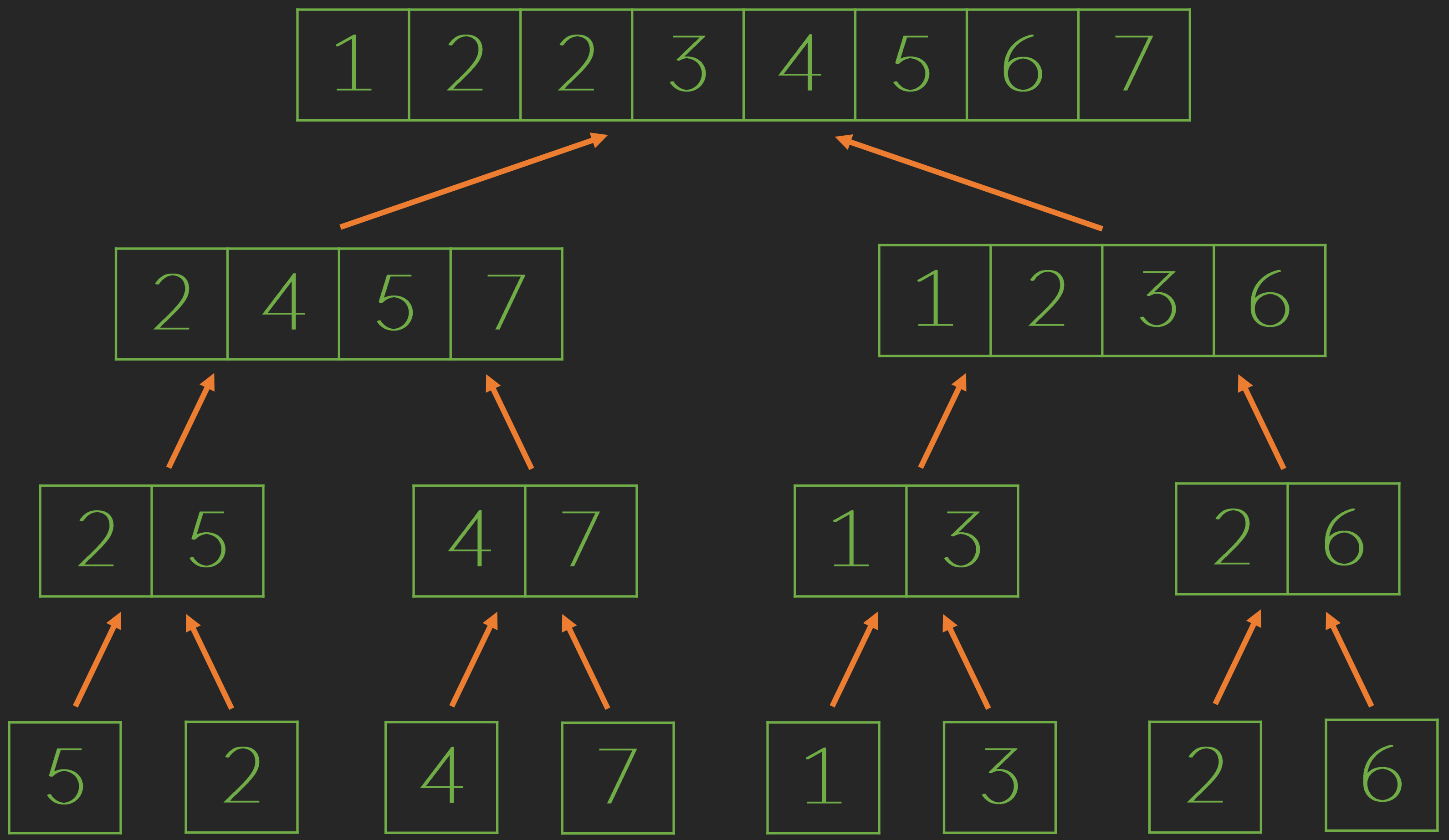
MERGE SORT



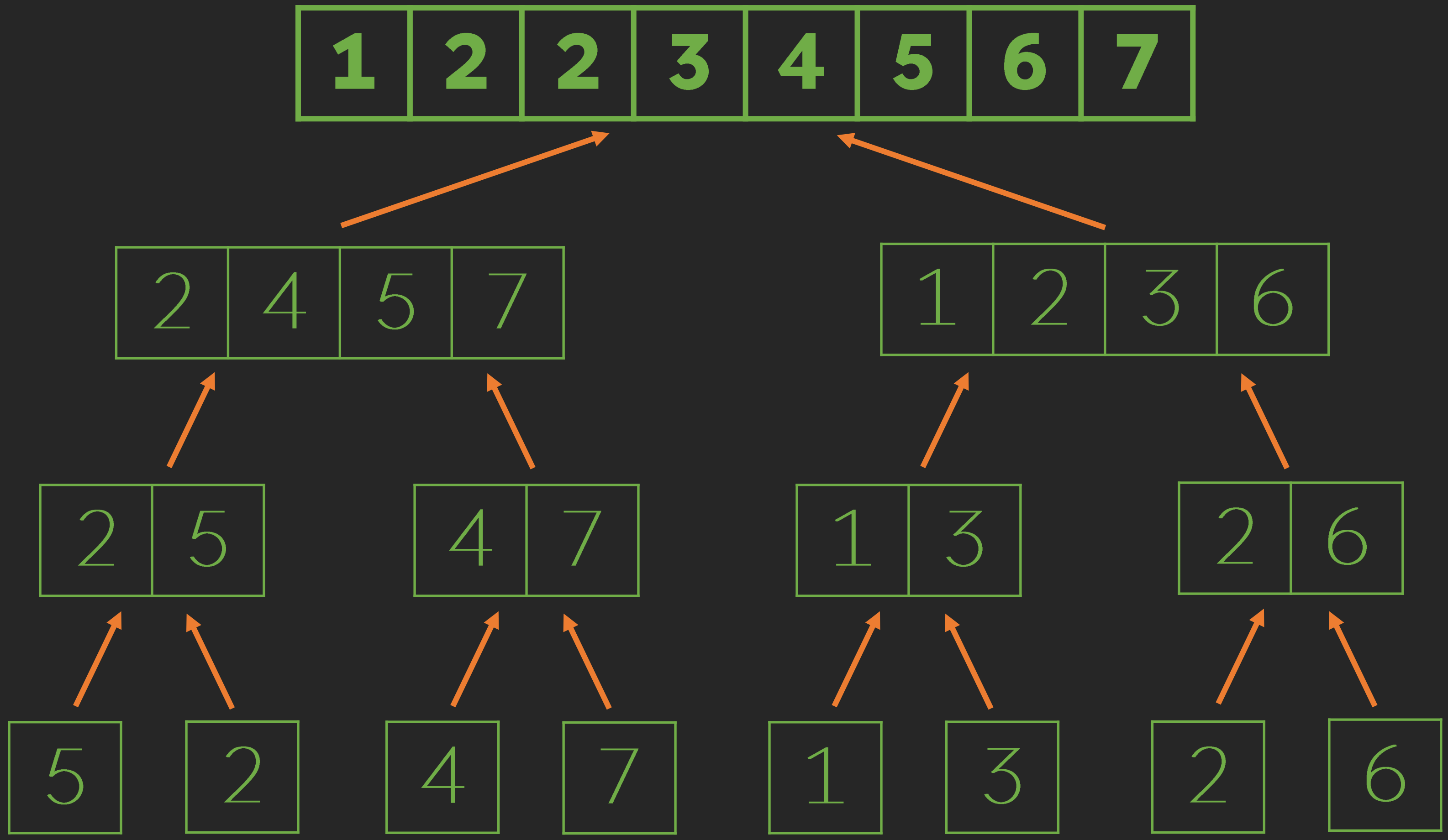
MERGE SORT



MERGE SORT



MERGE SORT



T(n) для MERGE SORT

```
mergeSort(A, l, r)
    if l < r
        m = (l + r) / 2
        mergeSort(A, l, m)
        mergeSort(A, m + 1, r)
        merge(A, l, m, r)
```

$T(n) = \ldots$

T(n) для MERGE SORT

```
mergeSort(A, l, r)
    if l < r
        m = (l + r) / 2
        mergeSort(A, l, m)
        mergeSort(A, m + 1, r)
        merge(A, l, m, r)
```

$T(n) = \ldots$

$\Theta(1)$

T(n) для MERGE SORT

mergeSort(A, l, r)	$T(n) = \ldots$
if $l < r$	$\Theta(1)$
$m = (l + r) / 2$	$\Theta(1)$
mergeSort(A, l, m)	$T(n/2)$
mergeSort$(A, m + 1, r)$	$T(n/2)$
merge(A, l, m, r)	

T(n) для MERGE SORT

mergeSort(A, l, r)	$T(n) = \ldots$
if $l < r$	$\Theta(1)$
$m = (l + r)\ /\ 2$	$\Theta(1)$
mergeSort(A, l, m)	$T(n/2)$
mergeSort(A, $m + 1$, r)	$T(n/2)$
merge(A, l, m, r)	$C(n)$

T(n) для MERGE SORT

Получено рекуррентное соотношение для оценки временной сложности

T(n) для MERGE SORT

Получено рекуррентное соотношение для оценки временной сложности

$$T(n) = \begin{cases} \Theta(1) & \text{при } n = 1 \\ 2T(n/2) + C(n) & \text{при } n > 1 \end{cases}$$

T(n) для MERGE SORT

Получено рекуррентное соотношение для оценки временной сложности

$$T(n) = \begin{cases} \Theta(1) & \text{при } n = 1 \\ 2T(n/2) + C(n) & \text{при } n > 1 \end{cases}$$

Нужны общие методы для решения подобных рекуррентных соотношений

Recap

Асимптотический анализ временной сложности алгоритмов – символы Θ, O, Ω

Рекуррентные соотношения для анализа сложности алгоритмов
$T(n) = T(n-1) + \ldots, T(n) = aT(n/b) + \ldots$

Teaser – Лекция 4

Метод подстановки и дерево рекурсии для решения рекуррентных соотношений

Алгоритм умножения квадратных матриц

Алгоритм Карацубы