

# О курсе и алгоритмах

Алгоритм: вход(значения)  $\rightarrow$  вычислительная процедура  $\rightarrow$  выход(значения)

Используются для решения задач:

Google Maps — построение маршрута = графы

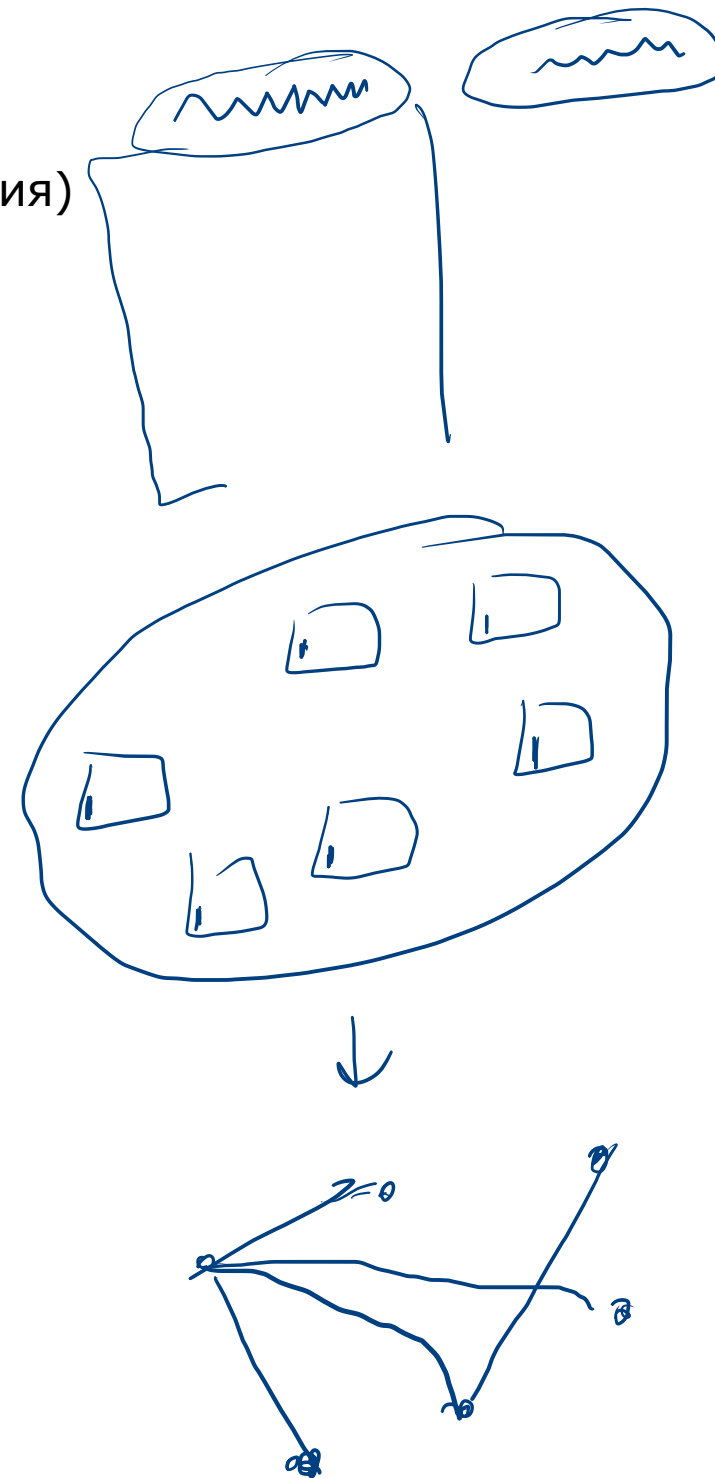
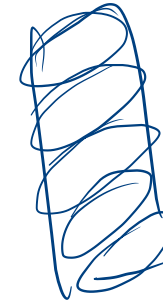
Биоинформатика — обработка геномных последовательностей = строки

Интернет — поиск страниц = хеш-таблицы  
и много других...

В курсе изучим:

- Сортировки
- Элементарные структуры данных (стеки, очереди, списки)
- Кучи и двоичные деревья поиска
- Графы и алгоритмы на них
- Хеш-таблицы
- Алгоритмы для работы со строками

AT... —



## Зачем оценивать время работы алгоритма?

Задача: отсортировать массив чисел

A

$10^9$  ком/сек

$n^2$  опер

Массив

$10^4$

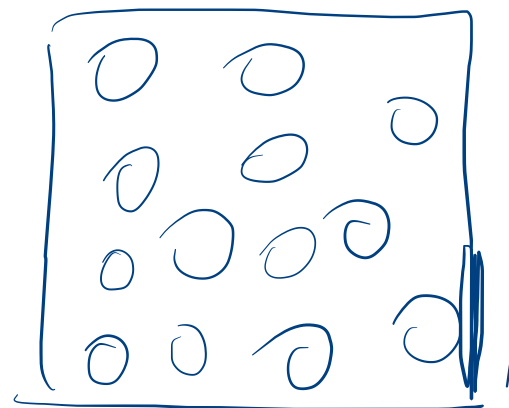
элементов

$n = 10^7$

B

$10^7$  ком/сек

$n \log n$  опер



7	5	1	13	19
---	---	---	----	----

1 5 7 13 19

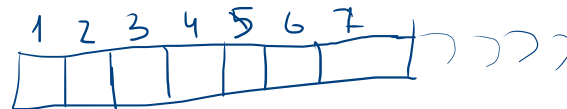
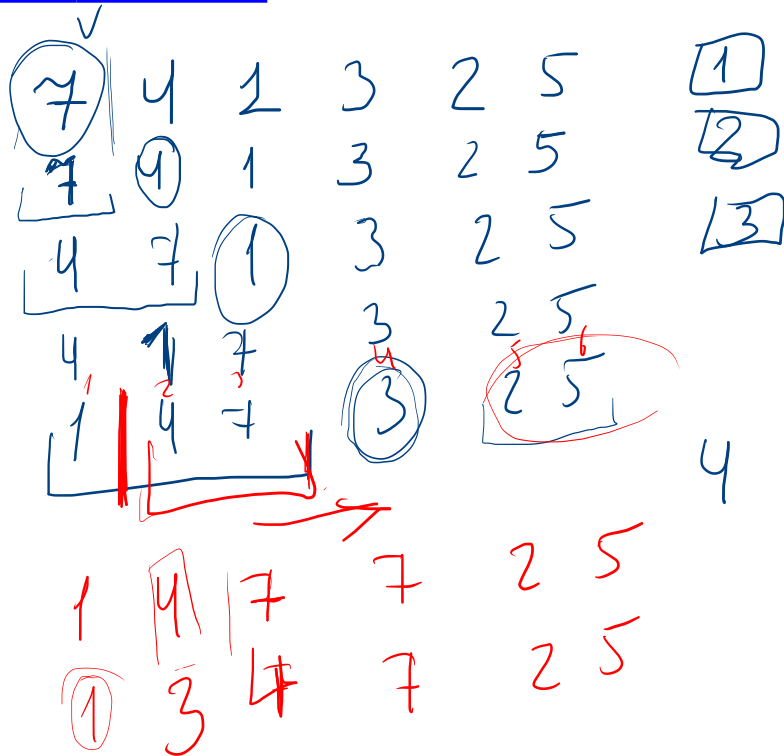
$$\frac{(10^7)^2}{10^9} = 10^5 \text{ сек} > \text{сутки}$$

$$\frac{10^7 \cdot \log 10^7}{10^7} \approx 23,25 \text{ сек}$$

Вывод: скорость компьютера менее важный показатель, чем эффективность алгоритма

## Сортировка вставкой

Пример:

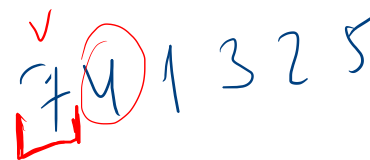
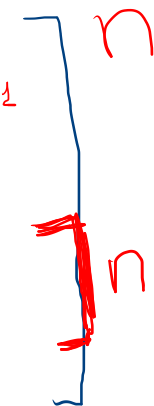


$$\begin{aligned} \text{key} &= 3 \\ i &= j - 1 = 3 \end{aligned}$$

Оценка времени работы:

INSERTION-SORT(A)

```
1 for j = 2 to A.length
2   key = A[j]
3   // Вставка A[j] в отсортированную
   // последовательность A[1..j-1]
4   i = j - 1
5   while i > 0 и A[i] > key
6     A[i+1] = A[i]
7     i = i - 1
8   A[i+1] = key
```



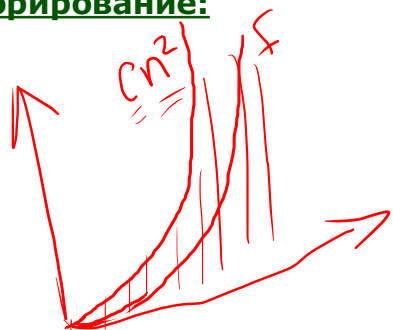
key = 4



**Худший случай:** массив упорядочен в порядке обратном искомому

**Лучший случай:** массив упорядочен

Мажорирование:



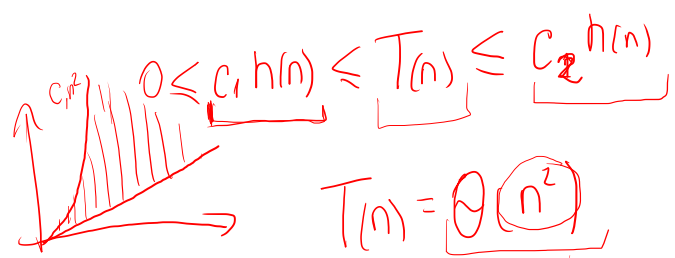
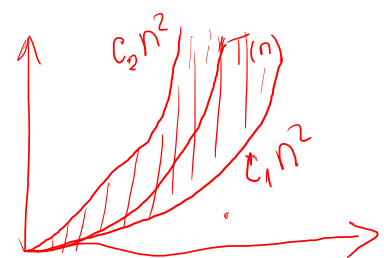
$$\begin{aligned} \sum_{j=2}^n j-1 &= \sum_{j=1}^{n-1} j = \frac{(n-1)n}{2} = \frac{n^2-n}{2} \\ &= O(n^2) \\ g &= O(n \log n) \end{aligned}$$

Асимптотические обозначения

1. Точная оценка.

```
INSERTION-SORT(A)
1 for j = 2 to A.length
2   key = A[j]
3   // Вставка A[j] в отсортированную
   последовательность A[1..j-1].
4   i = j - 1
5   while i > 0 и A[i] > key
6     A[i+1] = A[i]
7     i = i - 1
8   A[i+1] = key
```

$T(n)$  — время работы алгоритма *размера вход. данных*  
 $T(n) = \Theta(h(n))$   
 $n, n^2, n \log n, n^3$   
 $\Theta(h(n)) = \begin{cases} T(n) : \exists c_1 > 0, c_2 > 0, n_0 > 0 \\ \forall n > n_0 : \end{cases}$



2. Верхняя оценка.

$$O(h(n)) = \begin{cases} T(n), \exists c > 0, n_0 > 0 \\ \forall n > n_0, 0 \leq T(n) \leq c \cdot h(n) \end{cases}$$



$h(n)$  — верхняя асимптотическая оценка



3. Нижняя оценка.

$$\Omega(h(n)) = \begin{cases} T(n) : \exists c > 0, n_0 > 0 \\ \forall n > n_0, 0 \leq c \cdot h(n) \leq T(n) \end{cases}$$

4. Теорема о связи.  $\Theta, \Omega, O$

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



## Как проверять предположения об оценках

Пример доказательства, что функция является точной асимптотической оценкой

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

или  $C_1 > 0, C_2 > 0, n_0$

$$0 \leq C_1 \cdot n^2 \leq \frac{1}{2}n^2 - 3n \leq C_2 \cdot n^2$$

$$C_1 \leq \left[ \frac{1}{2} - \frac{3}{n} \right] \leq C_2$$

$n \uparrow \Rightarrow \frac{1}{2} - \frac{3}{n} \uparrow$

$$\frac{1}{2} - \frac{3}{n} > 0 \quad \text{верно при } n \geq 7$$

$$\frac{1}{2} - \frac{3}{n} \geq \frac{1}{14} \quad \text{при } n \geq 7$$

$$C_1 = \frac{1}{14}$$

$$\text{при } n \geq 1, \quad C_2 = \frac{1}{2}$$

$$C_1 = \frac{1}{14}$$

$$C_2 = \frac{1}{2}$$

$$n_0 = \max(7, 1) = 7$$

## Свойства O

$$\boxed{T_1(n) = O(g_1(n))} \quad T_2(n) = \boxed{O(g_2(n))}$$

1) Правило сумм  $T_1(n) + T_2(n) = \boxed{O(\max(g_1(n), g_2(n)))}$

2) Правило произведений  $T_1(n) \cdot T_2(n) = O(g_1(n) \cdot g_2(n))$

3) Умножение на константу

4) Прибавление константы

## Корректность сортировки вставками

Один из вариантов — с помощью инварианта цикла

**Опр** Инвариант цикла — истинное утверждение, которое сохраняется перед и после каждой итерации цикла алгоритма

Инициализация — верно до начала работы

Сохранение — верно после каждой итерации

Завершение — верно после выполнения алгоритма

Инвариант: Массив  $A[1 \dots j-1]$  содержит те же элементы, что были изначально, но в отсортированном виде

Инициализация:

Сохранение:

Завершение: