Сложность алгоритмов

Это несложно

Что такое алгоритм

- это последовательность элементарных операций, обрабатывающая входную строку x для получения выходной строки y

Скорость алгоритма

- Дан массив [3, 14, 15, 9, 2, 6, 5, 35, 8, 9]

Сколько секунд понадобится компьютеру, чтобы найти индекс элемента 9 в данном массиве?

- Дан массив [3, 2, 5, 6, 8, 9, 9, 14, 15, 35]

А сколько секунд понадобится компьютеру, в отсортированном массиве?

Что такое сложность

Временная

это функция, показывающая точную верхнюю границу количества элементарных операций, необходимых для завершения работы алгоритма, в зависимости от количества символов во входе

Емкостная

аналогичная оценка для **дополнительной** памяти, необходимой для анализа входа. Память, использующаяся для хранения входа, не учитывается

Оценка сложности

- В лучшем случае (оценка снизу)
- В худшем случае (оценка сверху)
- В среднем

[3, 14, 15, 9, 2, 6, 5, 35, 8, 9, 42]

Ищем число 3

Ищем число 42

Ищем число

Оценка сложности

- «o-малое» f(n) = o(g(n))

f растет ничтожно медленно, по сравнению с g

lim $f(n) / g(n) = 0$ при $n \to \infty$

- «O-большое» f(n) = O(g(n))

f ограничена сверху функцией g

lim $f(n) / g(n) < \infty$ при $n \to \infty$

Сложность алгоритма

- Дан массив [3, 14, 15, 9, 2, 6, 5, 35, 8, 9]

Какова сложность алгоритма поиска в данном массиве?

- Дан массив [3, 2, 5, 6, 8, 9, 9, 14, 15, 35]

Какова сложность алгоритма поиска в отсортированном массиве?

Пример

```
int i = 1;
while (i < n)
{
    for(int j=0; j<n-2; j++)
        count++;
    i++;
}
```

Пример

```
int i = 1;
while (i < n)
{
    for(int j=0; j<n-2; j++)
        count++;
    i++;
}
```

Точное количество операций (n-1)*(n-2)

Пример

```
int i = 1;
while (i < n)
{
    for(int j=0; j<n-2; j++)
        count++;
    i++;
}
```

Оценка сверху $(n-1)*(n-2) = \Theta(n^2)$

Свойства O-большое

$O(f) + O(g) = O(\max\{f, g\})$

$C \cdot O(f) = O(f)$

$O(f) \cdot O(g) = O(f \cdot g)$

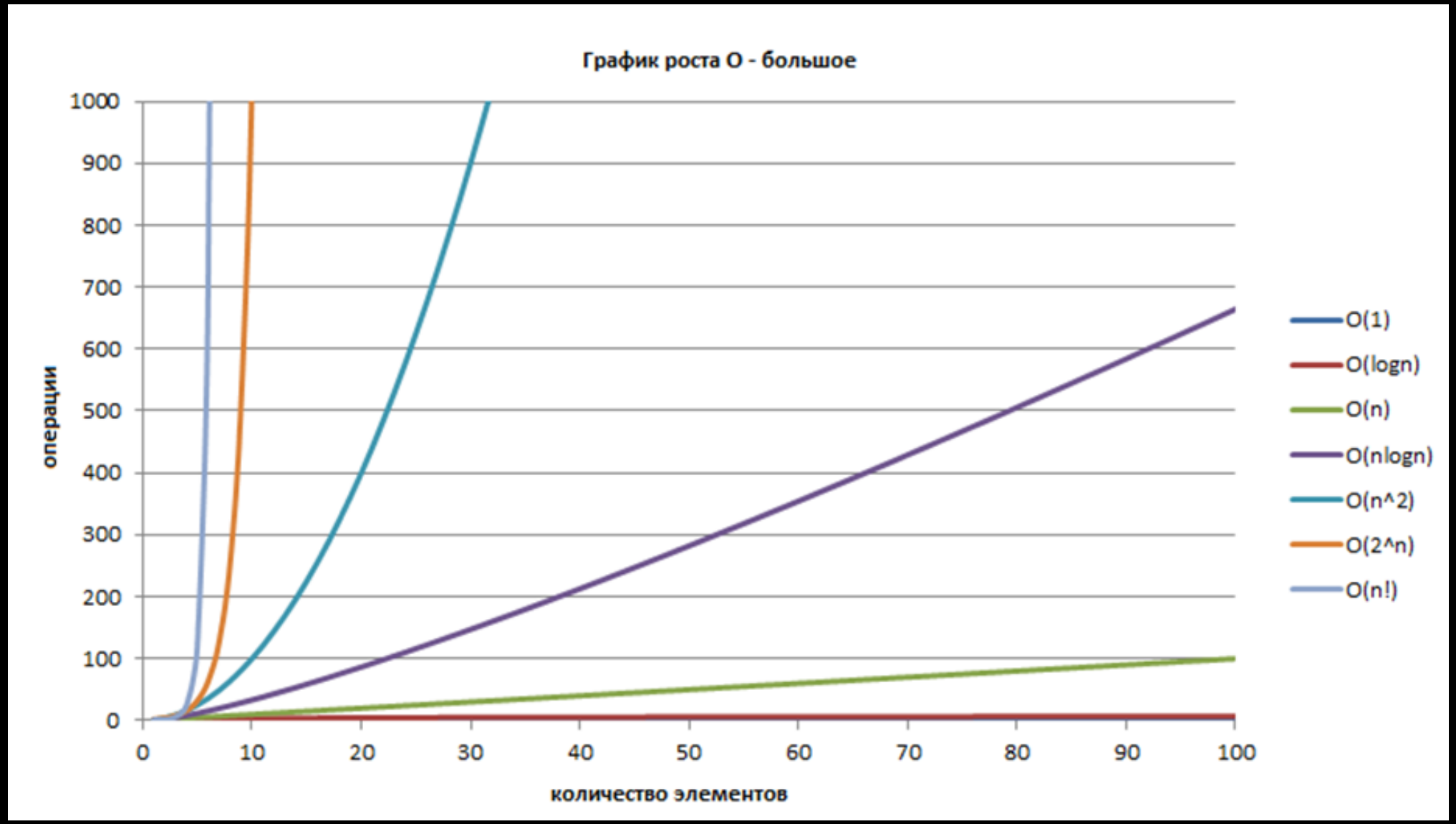
$O(n^2) + O(n) = O(n^2)$

$n \cdot O(n) = O(n^2)$

Классификация

- Если $f = \Theta\left(\log^k n\right)$: логарифмическим при $k = 1$, полилогарифмическим при $k > 1$.
- Если $f = \Theta(n)$: линейным.
- Если $f = \Theta\left(n \log^k n\right)$: linearithmic при $k = 1$, квазилинейным при $k > 1$.
- Если $f = \Theta\left(n^k\right)$: полиномиальным, при $k = 2$ квадратичным.
- Если $f = \Theta\left(2^{n^k}\right)$: экспоненциальным.

Оценка сложности



Оценка сложности

размер / сложность	10	20	30	40	50	60
n	0,00001 сек.	0,00002 сек.	0,00003 сек.	0,00004 сек.	0,00005 сек.	0,00005 сек.
n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	0,0016 сек.	0,0025 сек.	0,0036 сек.
n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,064 сек.	0,125 сек.	0,216 сек.
n^5	0,1 сек.	3,2 сек.	24,3 сек.	1,7 минут	5,2 минут	13 минут
2^n	0,0001 сек.	1 сек.	17,9 минут	12,7 дней	35,7 веков	366 веков
3^n	0,059 сек.	58 минут	6,5 лет	3855 веков	2×10^8 веков	$1{,}3 \times 10^{13}$ веков

Откуда берется квадрат?

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
    {
        …
    }
```

Откуда берется логарифм?

```
for (int i = 0; i < n; i*=2)
    {
        …
    }
```

Тренировка

```
var count = 0;
for (int i = 0; i < 1000000; i += n)
    count++;
```

Онлайн курс

https://ulearn.me/Course/complexity