
Erhaft

Анализ алгоритмов.

RAM - модель вычислений

асимптотический анализ вычислительной сложности.

„Big Oh" - асимптотическая нотация

RAM (Random Access Machine) - машина с произвольным доступом к памяти.

Работа RAM:

1) для исполнения простой операции (+; -; *, =, if, call) требуется 1 временной шаг

2) циклы и подпрогр-и - состоят из простых программ.
Время исполнения цикла / подпрограммы завис от к-ва операций.

3) Каждое обращение к памяти - 1 временной шаг!
Кэш и диск в RAM не исп-ся! => PC имеет ∞ V памяти.

We ~~count~~ measure time under the RAM model by counting the number of steps an algorith takes.

RAM - мат. модель исп-ся для оценки ресурсов оптимальной работы алгоритма

вычислитель (CPU) (операции с-ми)
• на сколько быстро вычисл. алг
T - время
Для измерения исп-ся ф-я T(n)

память (memory)
• M - объем памяти для д-х, кот. мы располагаем.
• срав-ся для измерения ф-ция M(n)

Best-Case, Worst-Case, Average-Case Complexity. (p. 31)

To understand how good/bad the algorithm is in general we must know how it works <u>over</u> all possible instances.
задачи

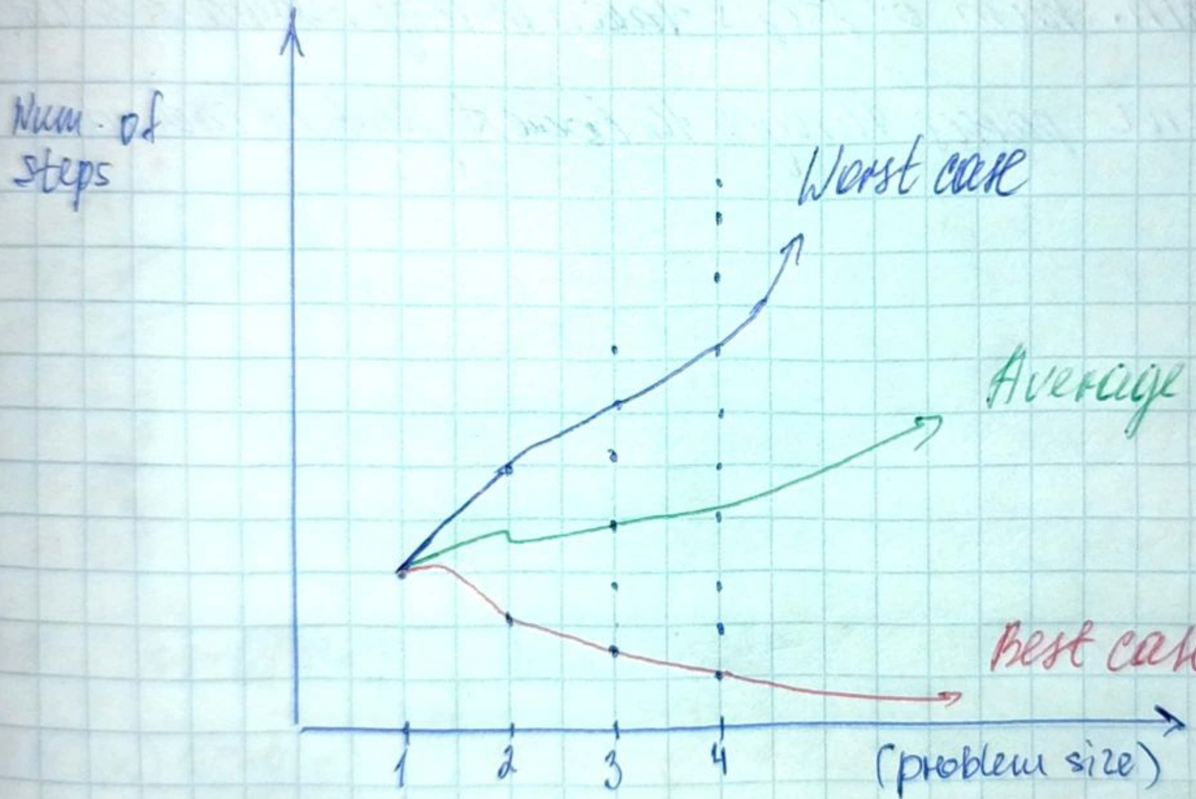
Устр-во машины:

RAM
вычислитель
память.

disk
сетевое взаимод-е
CPU
Кэш (cache)

У алгоритма как и ф-ции есть ОДЗ и обл. определ.

~~When~~ Each input instance can be presented as a point on a graph where the x-axis (the size of input problem) & the y-axis denotes the number of steps taken by alg. in this instance (входной экземпляр).



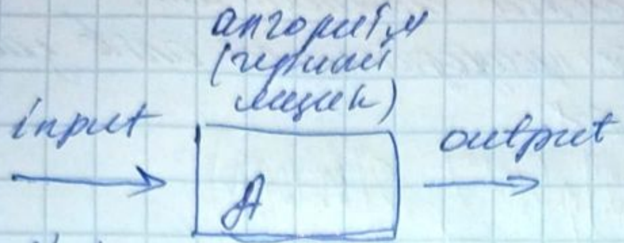
Worst case complexity — function defined by the max. number of steps taken in any instance of size n. This represents the curve passing through the highest point in each column.

The average case complexity (= expected time) of the alg.

- the func. defined by the aver. num. of steps over all instances of size n. экземпляров -размер

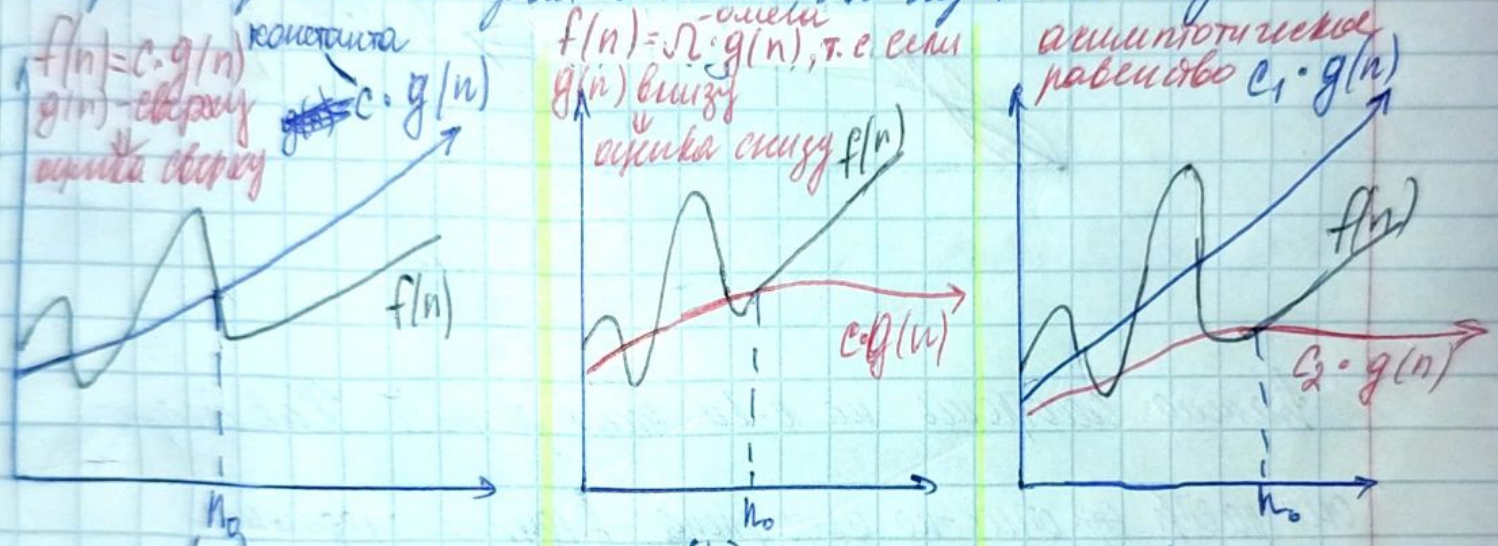
The best case complexity of alg. - the func. defined by min. num. of steps taken in any instance of size n. The curve passes through the lowest point in each column.

Рассмотрим алгоритм функционально.



$N = |input|$ (размер входа)

Скорость раб. алг. завис. от того что подали на вход.



(a) (b) (c)

a: $f(n) = O(g(n))$

b: $f(n) = \Omega(g(n))$

c: $f(n) = \Theta(g(n))$ — тета

O: $0 \le f(n) \le c \cdot g(n)$

т.е. $f(n)$ всегда будет $< g(n)$, умнож-ная константу, начиная с n_0.
Существует n_0 начиная с кот-го ф-я $f(n)$ нах-ся в рамках неравенства $0 \le f(n) \le c \cdot g(n)$

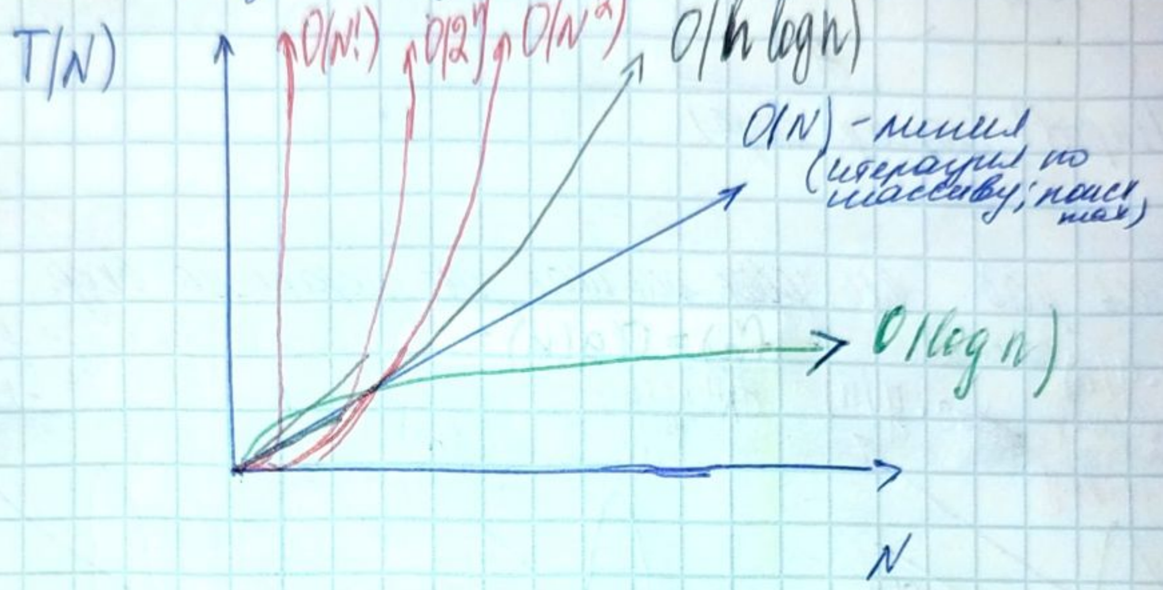
Ω: $0 \le c \cdot g(n) \le f(n)$

Θ: $c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$

У одной ф-ции $(g(n))$ есть 2 разн. константы $(c_1; c_2)$ ~~…~~ $\Rightarrow c_1, c_2$ определяют пояс внутри кот. нах-ся ф-я $f(n)$

$g(n)$ дает ограничения для $f(n)$! Определения верны после некой константы n_0. Значения слева от n_0 нас не интересуют (тк малы). Интересуют и большие

The „Big Oh" notation. – относительное представление сложности алг.
Показ-ет как будет меняться производительность алг-в в завис-ти от роста входящих д-х.



Важно смотреть на х-ки этих ф-ций. Как быстро она растет ~~др~~ относ-но друг друга. Будут ли предельные переходы. ~~Если~~ Что будет если на вход подать больш. ч-ло. Если линейная, т.о Ок. Лог-ки – Ок, если квадрат. и или !, то время вычисл-я очень увел-ся.

Оценка порядка роста ф-ций.

$f(n) = O(g(n))$

При увел-и к-ва вход. д-х могут расти 2 пок-ля:
- время выполнения алг.
- к-во памяти занимаемое алг. для обработки д-х.

Big Oh - показ-т скорость роста времени и памяти при увел-и к-ва входн. пар-ов.

сложность
O(n) - линейная ф-ция. С ростом к-ва входн. д-х растет и время (и т-1 прямая завис-ть)

константная сложность.
O(1) - константа. Т.е. рост. к-ва входн. д-х не влияет на время вып-я алг.

O(log n) - логарифмическая сложность (н.р. алг. бинарного поиска)

O(n log n) - сложность алгоритма merge sort.

O(n²) - (н.р. бежим по 2-м спискам один вложен во 2-й)

O(n³) - кубическая. (бежим по 3-м вложенным циклам)

O(2^n) - экспоненциальная сложность.

$O(n!)$ – факториальная сложность

В Big O мы отбрасываем все константы!

$f(n) = 2n$ и $g(n) = n$ – идентично

$O(2n) = O(n)$

Пример 1:

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

(пренебрегаем $O(n)$), т.к. $O(n^2)$ растет в разы быстрее $O(n)$.

Ex. 2:

$O(n + \log n) = O(n) + O(\log n) = O(n)$, т.к. $O(\log n) < O(n)$

Ex 3:

$O(60 \cdot 2^n + 10n^{100}) = O(2^n + n^{100})$ (отбрасыв. константы)

$= O(2^n) + O(n^{100}) = O(2^n)$, т.к. $O(2^n) > O(n^{100})$, при $n \to \infty$

Ex 4:

$O(n^2 + m) = O(n^2 + m)$ – $\overset{\text{параметр}}{m}$ – нельзя убрать т.к. ничего о нем не известно

Ex 5:

Есть 3 влож-х цикла. 1й и 2й идут от 0 до n, а 3й от 1 до 10.

Какая результирующая сложность?

$O(100\,000 * n^2) = O(n^2)$

Ex 4:

Есть 2 влож-х цикла. 1-й от 0 до n, 2-й от i до n.

Индекс 2-го влож-го эл-нта нач-ся с эл-та до которого работал 1-й цикл.

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

Ex. 6:

$f(n) = 3n^2 - 100n + 6 = O(3n^2) + O(-100n) + O(6) = O(n^2)$

$f(n) =$

Ex 7: Верно ли $2^{n+1} = \Theta(2^n)$ -?

$f(n) = O(g(n))$

$O(2^{n+1}) = O(2^n \cdot 2)$

$2 \cdot 2^n \leq c \cdot 2^n$, для $c \geq 2$

Ex 8: Верно ли $2^{n+1} = \Omega(2^n)$ -?

$f(n) = \Omega(g(n))$, если $c > 0$, при кот-й для всех больших зн-й $f(n) \geq c \cdot g(n)$. Это условие верно для $0 < c \leq 2$. Границы O-большое и Ω-большое подразумевают $2^{n+1} = \Theta(2^n)$

Ex 9: Верно ли $(x+y)^2 = O(x^2+y^2)$ выр-е верно, если найдем с такое что $(x+y)^2 \le c(x^2+y^2)$

$(x+y)^2 = x^2+2xy+y^2$

Пусть $x \le y$, тогда

$x \le y$

$2x \le 2y$

$2xy \le 2y^2$

$2xy \le 2y^2 \le 2(x^2+y^2)$

Пусть $x \ge 0$

$2yx \le 2x^2 \le 2(x^2+y^2)$

Выражение $2xy$ ограничивается двойным значением ф-ции x^2+y^2. $\Rightarrow (x+y)^2 \le 3(x^2+y^2)$. $\Rightarrow$ Решение верно.

Growth Rates & Dominance Relations

(Скорость роста и отношение доминиров-я)

With the Big Oh' notation we cavalierly (- убирать) discard the multiplicative constants. Thus,

$f(n) = 0.001n^2$ & $g(n) = 1000n^2$ are treated equally as n^2 for all values of n

Running time of common functions is measured in nanoseconds