«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
Факультет информационных технологий и анализа больших данных
Кафедра информационных технологий

СЛОЖНОСТЬ АЛГОРИТМОВ И ЕЁ ЗНАЧЕНИЕ В ПОДГОТОВКЕ ИТ СПЕЦИАЛИСТОВ

Горохова Римма Ивановна, к.пед.н.,
доцент кафедры информационных технологий,
Финансовый университет при Правительстве Российской Федерации
Россия, г. Москва
E-mail: rigorokhova@fa.ru

Актуальность

-Основы программирования - возросшая значимость IT-компетенций в образовательной среде.

- Абстрактные структуры данных играют ключевую роль в программировании, позволяя систематизировать работу с информацией и обеспечивать эффективную организацию вычислений.

- Рассматривают не конкретные технические детали их хранения, а изучают такие структуры относительно логики доступа, добавления и удаления элементов.

- Возможность применять одни и те же принципы к различным реализациям, адаптируя их к разнообразным задачам и контекстам.

Основные вопросы

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Базовые концепции программирования	• принципы построения программ на языках программирования • понимания базовых структур данных
Основные структуры данных	• списки • кортежи • множества • словари
Сложные структуры данных	• стеки • очереди • деревья • односвязные списки
Алгоритмы работы со структурами данных	• важность алгоритмов для работы со структурами данных. • примеры популярных алгоритмов: сортировки, поиска и обхода. • подходы к улучшению производительности программ.
Визуализация структур данных	• построение изображений, графиков и диаграмм • построение визуализаций анализа данных

Сложность алгоритмов

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Сложность алгоритма - количественная оценка времени выполнения операций и объёма используемой памяти

Категории сложности:

Временная сложность –

характеризует, как время выполнения алгоритма растёт с увеличением объема входных данных.

Пространственная сложность –

описывает, сколько памяти требуется алгоритму в зависимости от размера входных данных

Оценка производительности алгоритма

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Трудности:

- Особенности реализации алгоритма
- Используемый язык программирования
- Характеристики аппаратного обеспечения
- Специфика входных данных

Сложности экспериментального анализа эффективности алгоритма

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Экспериментальное время работы двух алгоритмов сложно напрямую сравнить до тех пор, пока эксперименты не будут проведены на одинаковом аппаратном обеспечении и в одинаковом программном окружении.

Эксперименты могут быть выполнены только для ограниченного набора тестовых входных данных. Таким образом, они не учитывают время работы на входных данных, которые не были включены в набор тестовых входных данных, а эти результаты могут быть важны.

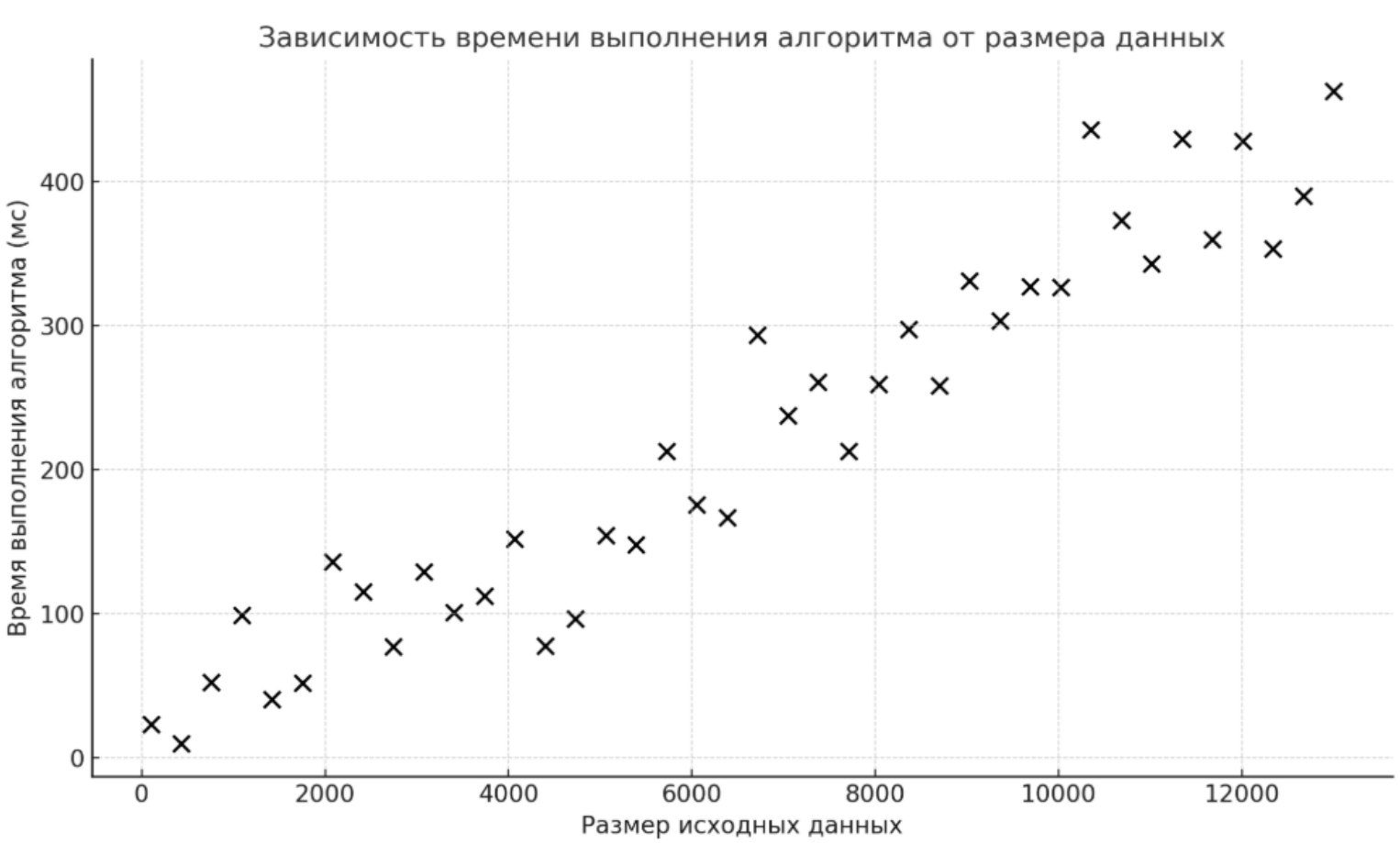
Для экспериментальной оценки производительности алгоритма он должен быть полностью реализован на одном из языков программирования.

Знание анализа сложности позволяет программисту оценивать, насколько эффективно тот или иной алгоритм решает задачу. Это важно для выбора наилучшего подхода к реализации функциональности, поскольку алгоритмы могут значительно различаться по времени и ресурсам, потребляемым в процессе выполнения

Зависимость времени от размера данных

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

При увеличении объёма данных время обработки возрастает, причём оно зависит не только от их количества, но и от внутренней структуры



При одинаковом объёме входных данных возможны вариации результатов:
- различия во внутренней организации данных ;
- состояние вычислительной системы (например, одновременным выполнением других процессов).

Экспериментальный подход не всегда является оптимальным.

Альтернативная методика

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Подсчет базовых операций на базе высокоуровневого описания алгоритма в виде псевдокода или фрагмента кода на языке программирования.

Набор рассматриваемых примитивных операций:
- Присвоение значения идентификатору
- Выполнение арифметической операции (например, сложения двух чисел)
- Сравнение двух чисел
- Получение одного элемента из контейнера (например, списка) по индексу
- Выполнение вызова функции (операции, выполняемые внутри функции, должны учитываться дополнительно)
- Возврат из функции

Асимптотический анализ сложности алгоритма

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Пусть $f(n)$ и $g(n)$ — две функции, определённые на множестве натуральных чисел и принимающие неотрицательные значения. Говорят, что $f(n)$ принадлежит $O(g(n))$, если существуют положительные константы c и n_0, такие что для всех $n \geq n_0$ выполняется неравенство $f(n) \leq c \cdot g(n)$.

Наименование	Функция
Постоянная	1
Логарифмическая	logn
Линейная	n
Линейно-логарифмическая	nlogn
Квадратичная	n^2
Кубическая	n^3
Экспоненциальная	a^n

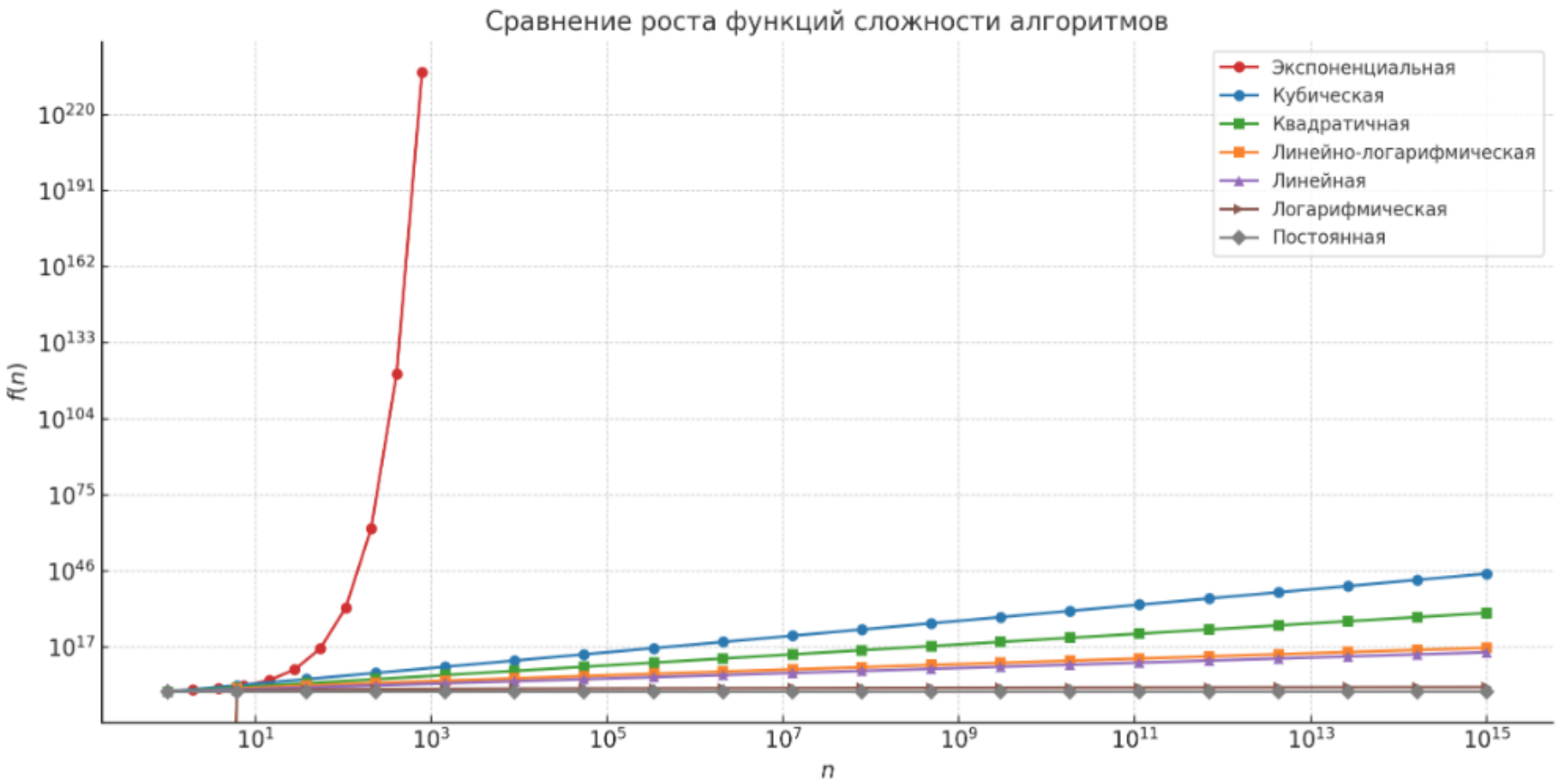
Подобный подход позволяет сравнивать эффективность различных алгоритмов, не привязываясь к конкретным аппаратным или программным реализациям

Классификация алгоритмов по асимптотической сложности

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Постоянная (O(1))	• время выполнения алгоритма практически неизменно при увеличении объёма входных данных , независимость работы алгоритма от размера обрабатываемой информации
Логарифмическая (O(logn))	• увеличение времени выполнения происходит существенно медленнее, чем при линейном росте, поскольку логарифмическая функция обладает значительно меньшей скоростью возрастания по сравнению со степенной зависимостью
Линейная (O(n))	• время работы алгоритма пропорционально количеству обрабатываемых элементов, отражая прямую зависимость между входными данными и затраченными ресурсам
Линейно-логарифмическая (O(nlogn))	• наблюдается прирост сложности, который превышает линейный, но остаётся существенно ниже квадратичного, что характерно для алгоритмов, где доминирует предварительная сортировка или подобные процедуры
Квадратичная ($O(n^2)$)	• время выполнения возрастает пропорционально квадрату числа элементов, что типично для алгоритмов, реализующих вложенные циклы или иные процедуры, приводящие к многошаговому перебору
Кубическая ($O(n^3)$)	• характеризуется ещё более стремительным увеличением времени работы, чем квадратичная сложность, и обычно наблюдается в алгоритмах с трёхуровневой структурой вложенных операций
Экспоненциальная ($O(a^n)$, $a>1$):	• время выполнения растёт экспоненциально с увеличением объёма входных данных, что делает такие алгоритмы практически неприменимыми для задач с большими данными

Графики функций асимптотических оценок сложности алгоритмов



В задачах, связанных с обработкой больших объёмов информации, преимущественное внимание уделяется оценкам, основанным на ***линейной или линейно-логарифмической*** зависимости.

Применение более сложных алгоритмических моделей зачастую исключается, поскольку реализация требуемых объёмов данных оказывается практически неосуществимой

Пространственная сложность

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Анализ пространственной сложности позволяет разработчикам выбирать алгоритмы, обеспечивающие необходимый баланс между производительностью и расходом памяти.

Постоянная (O(1))	• алгоритм использует фиксированное количество памяти
Линейная (O(n))	• потребление памяти пропорционально размеру входных данных
Квадратичная ($O(n^2)$)	• потребление памяти может расти по мере увеличения входных данных (например, при использовании двумерных массивов).

Сложность алгоритмов

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Поиск в отсортированном массиве с помощью бинарного поиска имеет временную сложность O(log n).

Алгоритм сортировки слиянием имеет временную сложность O(n log n) и пространственную сложность O(n).

Эти категории и нотации позволяют разработчикам и исследователям сравнивать эффективность различных алгоритмов и выбирать наиболее подходящие для конкретных задач.

Выводы

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

- Анализ сложности алгоритмов позволяет оценивать масштабируемость решений, сравнивать альтернативные подходы и определять узкие места в реализации, которые могут стать объектом дальнейшей оптимизации.
- Применяя данные методы, разработчики могут создавать гибкие и эффективные системы, способные работать с растущими объёмами данных без существенного ухудшения производительности.
- Современные исследования в области анализа алгоритмов направлены на разработку новых методов, учитывающих особенности распределённых вычислений, параллелизм и обработку потоковых данных.
- Применение адаптивных алгоритмов позволяет создавать масштабируемые решения, отвечающие современным требованиям к быстродействию и эффективности в условиях постоянно изменяющихся вычислительных ресурсов.
- Изучение сложности алгоритмов помогает программистам находить узкие места в приложениях. Понимание теории сложности позволяет разрабатывать более оптимизированные решения и минимизировать использование ресурсов, таких как процессорное время и память.

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Спасибо за внимание!

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
Факультет информационных технологий и анализа больших данных
Кафедра информационных технологий

СЛОЖНОСТЬ АЛГОРИТМОВ И ЕЁ ЗНАЧЕНИЕ В ПОДГОТОВКЕ ИТ СПЕЦИАЛИСТОВ

Горохова Римма Ивановна, к.пед.н.,
доцент кафедры информационных технологий,
Финансовый университет при Правительстве Российской Федерации
Россия, г. Москва
E-mail: rigorokhova@fa.ru