Алгоритмы и O-нотация

AIT
Hi-tech team

НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Иметь представление об оценке сложности алгоритмов

ПЛАН ЗАНЯТИЯ

- Оценка алгоритмов
- Вспоминаем математику
- Оценка сортировки выбором
- Оценка бинарного поиска
- Асимптотический анализ

ОЦЕНКА АЛГОРИТМОВ

Оценка **фактического времени** работы программы

Оценка **фактического количества примитивных операций**, которых требует алгоритм на определенном наборе данных (сравнение, присваивание, обращение к элементу массива).

Оценка скорости роста количества примитивных операций в зависимости от входных данных (**асимптотический анализ**).

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Минусы оценки времени

Плюсы и минусы оценки количества операций

ВСПОМИНАЕМ МАТЕМАТИКУ

Степень числа - это математическая операция, которая включает два числа: **основание** и **показатель** степени. Операция степени записывается как a^n, где a - основание, n - показатель. Значение a^n говорит о том, сколько раз число a умножается само на себя. Например $3^4 = 3 * 3 * 3 * 3 = 81$

Логарифм - это математическая операция, обратная возведению в степень. Логарифм числа b по основанию a определяется как степень, в которую нужно возвести a, чтобы получить b. Записывается это как $log_a b = x$, что читается как "логарифм b по основанию a равен x".
Например $log_3 81 = 4$

Функция - зависимость одной величины от другой, обозначается как $y = f(x)$. Где y - значение функции f в точке x. Например, парабола - $y = x^2$

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Степень

Логарифм

Функция

ОЦЕНКА СОРТИРОВКИ ВЫБОРОМ

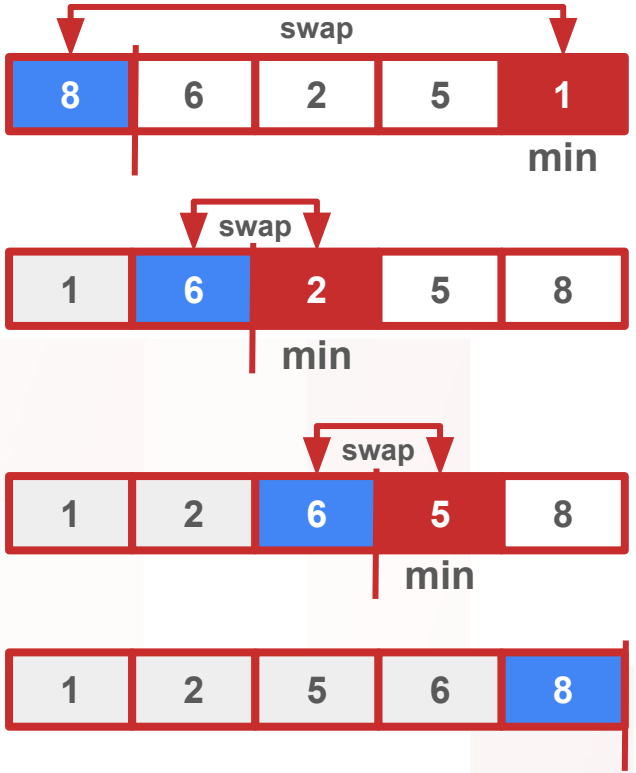
Посчитаем количество сравнений, выполняемых алгоритмом. В первом проходе алгоритм проверяет $n-1$ элементов, чтобы найти минимальный и переместить его на первую позицию.

Во втором проходе алгоритм уже не проверяет первые два элемента, поэтому он проверяет $n-2$ элементов, чтобы найти следующий минимальный и переместить его на вторую позицию.

Этот процесс продолжается, и на каждом следующем шаге количество проверяемых элементов уменьшается на один.

Это создает следующую серию операций:
$(n-1)+(n-2)+(n-3)+\ldots+1$.
Математически это арифметическая прогрессия, сумма элементов которой равна:

$$S_n = \frac{n(n-1)}{2}$$



ОЦЕНКА СОРТИРОВКИ ВЫБОРОМ

$$S_n = \frac{n(n-1)}{2}$$

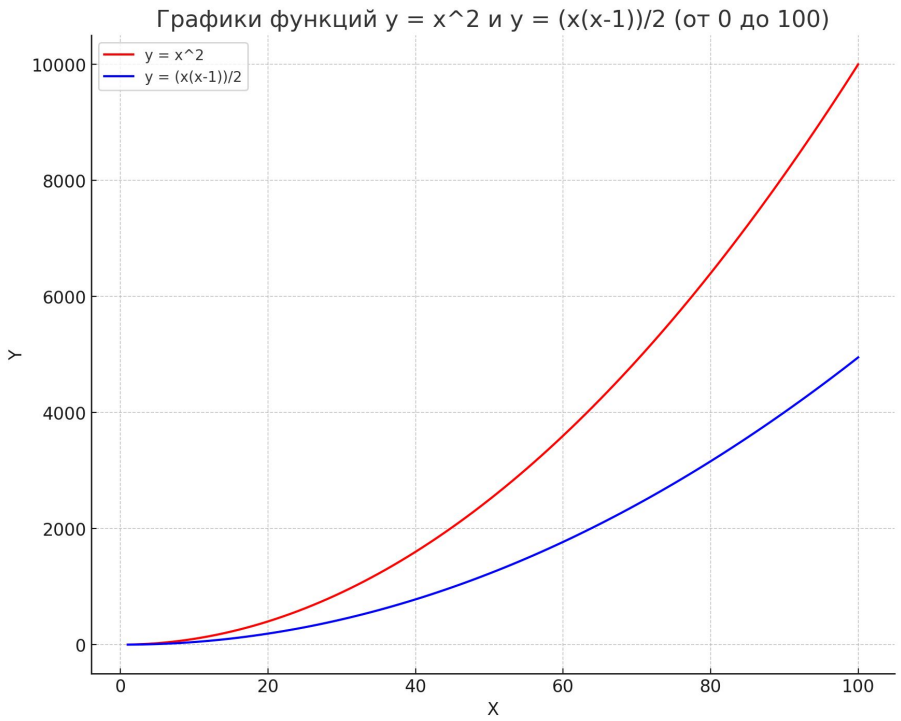
Где S_n - сумма элементов (количество операций сравнения, которые задействует алгоритм).

Таким образом, для массива длины **5** количество операций сравнения будет равно **10**.

Почему говорят, что сложность алгоритма сортировки выбором составляет $O(n^2)$? Заменим **n** на положительное целое **x** и посмотрим на графики функций.

Можно заметить, что значение синей функции **асимптотически** (приближенно) стремится к значению красной.

Следовательно, **асимптотическая сложность** алгоритма сортировки выбором равна $O(n^2)$.



СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Способ оценки сортировки выбором

Арифметическая прогрессия

Асимптотическое приближение

ОЦЕНКА БИНАРНОГО ПОИСКА

В лучшем случае алгоритм найдет искомый элемент за одно сравнение, следовательно, его асимптотическая сложность для лучшего случая O(1).

Для анализа худшего случая рассмотрим массив длины 128, на каждом шаге отсекая половину:

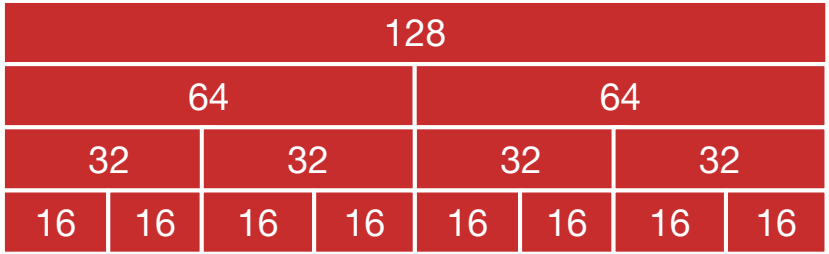
1-ое сравнение с центром, осталось 64 элемента
2-ое сравнение с центром, осталось 32 элемента
3-е сравнение с центром, осталось 16 элементов
4-ое сравнение с центром, осталось 8 элементов
5-ое сравнение с центром, осталось 4 элемента
6-ое сравнение с центром, осталось 2 элемента
7-ое сравнение с центром, остался 1 элемент

Таким образом, мы сделали 7 сравнений для нахождения числа. Это показатель степени, в которую нужно возвести число 2, чтобы получить 128 (исходное количество элементов).

По определению логарифма получаем:

$$\log_2 128 = 7$$

Следовательно, в худшем случае поиск элемента методом бинарного поиска в массиве длины n потребует $\log_2 n$ сравнений. Такая сложность алгоритма обозначается как O(log n)



СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Способ оценки бинарного поиска

Логарифмическая сложность бинарного поиска

АСИМПТОТИЧЕСКИЙ АНАЛИЗ

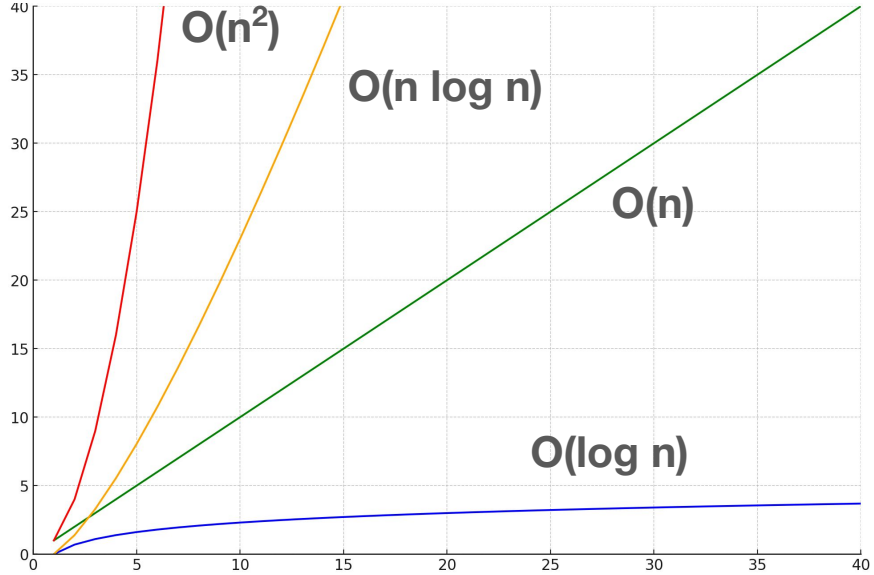
Асимптотика позволяет избежать точных значений, при этом описывая **порядок** роста, она показывает насколько быстро увеличивается время работы алгоритма при увеличении входных данных.

Асимптотический анализ дает абстрактное понимание эффективности алгоритма.

Так, обозначение $O(n^2)$ для сортировки выбором означает, что время работы (количество операций сравнения) растет **квадратично** от количества входных данных.

Мы также игнорируем константы и низшие порядки:

$5n^2 + 3n + 8$ будет асимптотически оцениваться как $O(n^2)$, поскольку именно n^2 в данном выражении отвечает за быстрый рост функции.



O(1) - получение элемента по индексу в массиве
O(log n) - бинарный поиск
O(n) - линейный поиск
O(n log n) - сортировка слиянием
O(n²) - сортировка выбором, пузырьковая сортировка

СТАВИМ ПЛЮС, ЕСЛИ ПОНЯТНО

Плюсы асимптотического подхода

Известные асимптотики

ПОИГРАЕМ ;)

- Оценка алгоритма
- Асимптотический анализ
- Идея оценки сортировки выбором
- Идея оценки бинарного поиска

ДОМАШНЕЕ ЗАДАНИЕ

1. Написать пару предложений с обоснованием сложности алгоритмов: линейного поиска, бинарного поиска, сортировки выбором
2. По желанию (или нет?): изучить алгоритм сортировки слиянием и написать обоснованную оценку его сложности
3. По желанию: реализовать алгоритм сортировки слиянием

AIT
Hi-tech team

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team