Алгоритмы и О-нотация

Оценка алгоритмов

Оценка работы программы может производиться различными способами, каждый из которых имеет свои преимущества и недостатки.

Оценка фактического времени работы программы

Этот метод заключается в замере реального времени, которое программа занимает на выполнение задачи. Предоставляет конкретные цифры времени выполнения, что полезно для реальных приложений. Но время выполнения сильно зависит от характеристик оборудования (железа) и окружения выполнения (операционной системы, других работающих программ), что делает результаты менее универсальными и трудными для сравнения.

Оценка фактического количества примитивных операций

Подсчет количества базовых операций (сравнений, присваиваний, обращений к элементам массива), которые выполняет алгоритм. Это более стабильная метрика, поскольку она не зависит от внешних условий исполнения программы.Тем не менее результаты зависят от конкретного набора входных данных и могут не отражать общую эффективность алгоритма при разных условиях.

Асимптотический анализ

Оценка скорости роста количества примитивных операций в зависимости от размера входных данных. Это включает понятие **"О большое" (Big O notation)**, которое описывает поведение алгоритма и предоставляет общее представление об эффективности, особенно на больших объемах данных.

Каждый из этих методов оценки работы программы имеет свою сферу применения и может быть более или менее подходящим в зависимости от конкретных целей анализа и условий, в которых выполняется программа. Важно уметь выбирать подходящий метод оценки в зависимости от поставленной задачи и имеющихся условий.

Математика

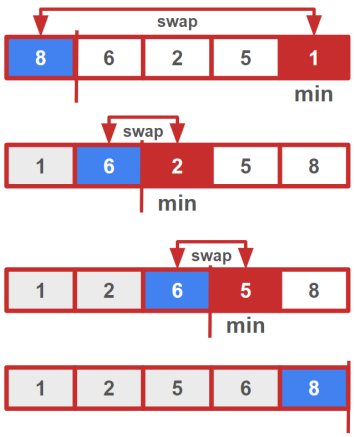
Степень числа - это математическая операция, которая включает два числа: основание и показатель степени. Операция степени записывается как a^n, где a - основание, n - показатель. Значение a^n говорит о том, сколько раз число a умножается само на себя.

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 - $y = x^2$

Оценка сортировки выбором



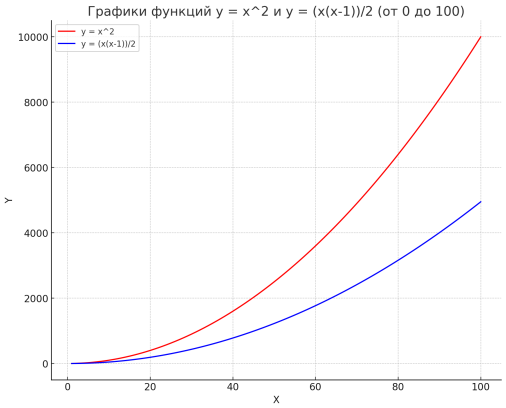
Общее количество сравнений в алгоритме: Алгоритм выполняет серию сравнений для определения минимального элемента в массиве и его последующего перемещения на соответствующую позицию.

Первый проход: В первом проходе алгоритм сравнивает **n-1** элементов, чтобы найти минимальный элемент и переместить его на первую позицию. Здесь **n** - общее количество элементов в массиве.
Второй проход: На втором проходе количество сравниваемых элементов уменьшается, поскольку первые два элемента уже отсортированы. Алгоритм сравнивает **n−2** элементов, чтобы найти следующий минимальный и переместить его на вторую позицию.
Последующие проходы: Процесс продолжается аналогичным образом. На каждом последующем проходе количество проверяемых элементов уменьшается на один. Это означает, что на третьем проходе сравниваются **n−3** элементов, на четвертом - **n−4** и так далее.

Таким образом, с каждым проходом алгоритма количество сравнений постепенно уменьшается, ускоряя процесс поиска минимального элемента по мере продвижения сортировки. Общее количество сравнений - **(n−1)+(n−2)+(n−3)+…+1**.
Математически это **арифметическая прогрессия**, сумма элементов которой равна:

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

Где $\mathbf{S_n}$ - 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 $\mathbf{O(n^2)}$? 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 $\mathbf{O(n^2)}$.

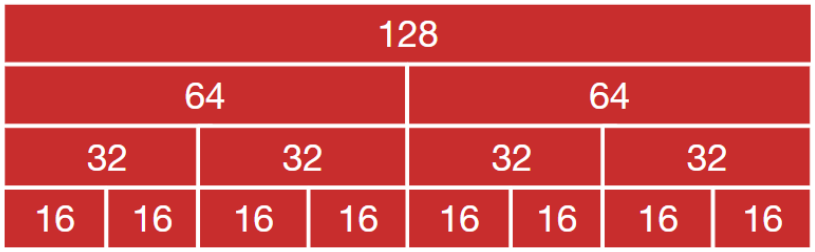
Оценка бинарного поиска

Бинарный поиск - это эффективный алгоритм поиска, который работает путем деления массива на две равные части и последующего сравнения искомого элемента с элементом в центре. В **лучшем случае** бинарный поиск находит элемент за одно сравнение, что дает асимптотическую сложность **O(1)**. Это происходит, когда искомый элемент находится в середине массива.

Для анализа **худшего случая** рассмотрим массив из 128 элементов:

1-е сравнение: Сравниваем с центральным элементом, остается 64 элемента.
2-е сравнение: Сравниваем с центральным элементом новой половины, остается 32 элемента.
3-е сравнение: Остается **16** элементов после сравнения.
4-е сравнение: Остается **8** элементов.
5-е сравнение: Остается **4** элемента.
6-е сравнение: Остается **2** элемента.
7-е сравнение: Остается **1** элемент.

В худшем случае бинарный поиск на массиве из **128** элементов потребует **7** сравнений. По определению логарифма - $\mathbf{\log_2 128 = 7}$, и асимптотическая сложность будет **O(log n)**, где **n** - количество элементов в массиве:



Асимптотический анализ

Асимптотический анализ позволяет оценить скорость роста времени работы алгоритма относительно увеличения объема входных данных. Это подход, который игнорирует точные значения времени выполнения или количества операций, сосредотачиваясь на общем тренде роста. Асимптотический анализ дает общее представление о производительности алгоритма, позволяя сравнивать алгоритмы на основе их фундаментальной эффективности, а не на основе тестирования с конкретными данными.

Как в случае с сортировкой выбором, обозначение $\mathbf{O(n^2)}$ указывает, что время выполнения алгоритма увеличивается квадратично по отношению к количеству входных данных. Это означает, что удвоение количества входных данных приведет к увеличению времени выполнения в четыре раза.

В выражении типа **$5n^2 + 3n + 8$**, асимптотическая сложность будет оцениваться как **$O(n^2)$**. Здесь ведущий член **n^2** определяет основную тенденцию роста времени выполнения, в то время как линейные члены **(3n)** и константы **(8)** игнорируются, поскольку их влияние становится незначительным для больших значений **n**.

Асимптотический анализ особенно важен при работе с большими объемами данных, где различия в асимптотической эффективности алгоритмов становятся критически значимыми. Асимптотический анализ позволяет легко сравнивать алгоритмы. Например, алгоритм с **O(n log n)** будет эффективнее, чем алгоритм с **$O(n^2)$**, для больших значений **n**.

O(1) - получение элемента по индексу в массиве
O(log n) - бинарный поиск
O(n) - линейный поиск
O(n log n) - сортировка слиянием
$O(n^2)$ - сортировка выбором, пузырьковая сортировка