Урок Спринт 1.11

Эффективность алгоритмов

Сортировки

Алгоритмы сортировки упорядочивают элементы коллекций по возрастанию или убыванию.

В повседневной жизни мы пользуемся многими наборами данных, которые уже отсортированы для нашего удобства. Например, книги в библиотеках стоят по алфавиту — так их проще найти.

Эффективность алгоритмов

Способ сортировки у каждого алгоритма может различаться. Поэтому важно знать, насколько эффективно он работает. Эту эффективность можно оценить по двум критериям:

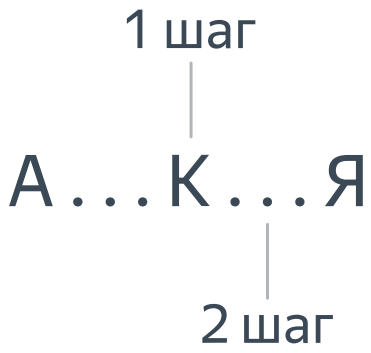
- Время работы
- Потребляемая память

Время выполнения описывает, сколько операций должен выполнить алгоритм, прежде чем он завершится. Оценить это время можно с помощью O-нотации (*Big O notation*).

Например, нам нужно найти самого старшего одноклассника. Как мы можем это сделать? Кажется, проще всего последовательно перебрать список всех учеников. Такой алгоритм называется линейным поиском. Если в классе n учеников, алгоритм обладает линейной сложностью $O(n)$. Ему нужно выполнить n операций, чтобы проверить всех одноклассников.

Возьмём другой пример: как найти в классном журнале одноклассника с фамилией, которая начинается на букву Ф? Так как данные в журнале уже отсортированы, эффективнее было бы воспользоваться другим алгоритмом.

Сначала заглянем в середину списка — и найдём там фамилию, например, на букву К. Значит, фамилии выше нам просматривать уже не нужно. Далее смотрим в середину второй половины списка и снова отбрасываем ненужную половину. Таки образом на каждом шаге мы сокращаем список фамилий в два раза:



Мы продолжаем искать до тех пор, пока не найдём нужную нам фамилию, либо пока в списке не останется только одна фамилия. Такой поиск называется бинарным, или двоичным. В среднем такому алгоритму нужно $O(\log n)$ операций до завершения.

Время выполнения также зависит от характера и размера входных данных. У каждого алгоритма поэтому есть худшее, среднее и лучшее значение вычислительной сложности.

Теперь о втором критерии — потребляемой памяти. Он показывает, сколько дополнительной памяти нужно алгоритму для работы. Например, если алгоритм принимает на вход массив размера n и создаёт новый массив размера n для каждого элемента, потребляемая память оценивается как n^2.