Алгоритмы

**Алгоритм** - набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий, при любом наборе исходных данных.

**Бинарный поиск**

**Двоичный (бинарный) поиск** (также известен как метод деления пополам и дихотомия) - классический алгоритм поиска элемента в **отсортированном** массиве (векторе), использующий дробление массива на половины.

Алгоритм двоичного поиска похож на то, как мы ищем слово в словаре. Открываем словарь посередине, смотрим в какой из половин будет нужное нам слово. Допустим, в первой. Открываем первую часть посередине, продолжаем выбирать и отбрасывать половинки, пока не найдем нужное слово.

В стандартной библиотеке Java уже имеется реализация двоичного поиска, который может быть расширен через интерфейс Comparator. Реализация находится в классе java.util.Arrays, в семействе методов binarySearch

При необходимости собственной реализации можно взять за основу код из библиотеки Java. Реализация двоичного поиска и сортировки слиянием не так банальна, как кажется, многие реализации содержат ошибки и ограничения.

Типичные ошибки при реализации двоичного поиска:

* Не работает с массивом из 0/1/2 элементов;
* Не находит первый или последний элемент;
* Некорректно работает, если элемента в массиве нет;
* Некорректно работает, если в массиве есть повторяющиеся элементы;
* Обращение к элементам за пределами массива;
* Переполнение переменной при вычислении среднего индекса (была в JDK);

**Сложность алгоритмов**

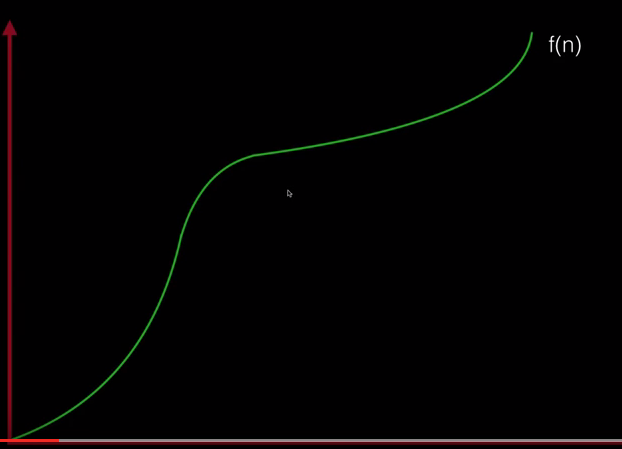
Вычислительная сложность - понятие в информатике и теории алгоритмов, обозначающее функцию зависимости объема работы, которая выполняется некоторым алгоритмом, от размера входных данных.

Теория сложности вычислений возникла из потребности сравнивать быстродействие алгоритмов, чётко описывать их поведение (время исполнения и объём необходимой памяти) в зависимости от размера входа.

Сложность можно рассматривать с разных точек зрения:

* **Временная сложность**. Количество элементарных операций, затраченных алгоритмом для решения конкретного экземпляра задачи, зависит не только от размера входных данных, но и от самих данных. Например, количество операций алгоритма сортировки вставками значительно меньше в случае, если входные данные уже отсортированы. Поэтому рассматривается временная сложность в **лучшем** и в **худшем** случае
* **Пространственная сложность**. Здесь говорят не о количестве элементарных операций, а об объёме используемой памяти.

**Асимптотические обозначения сложности**

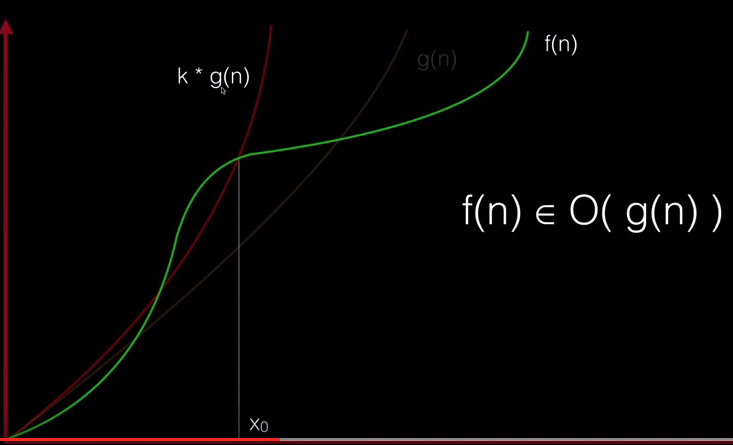
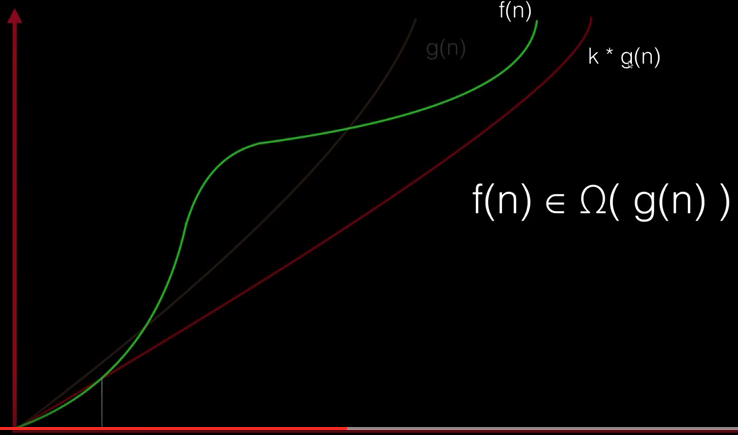
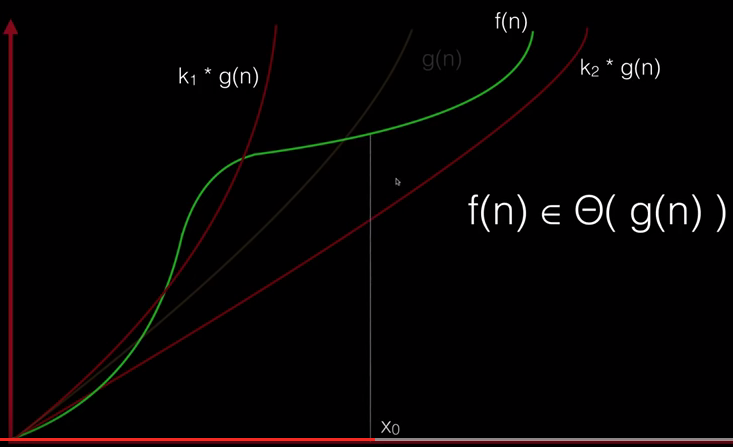
Пускай f(n) – функция, которая описывает зависимость количества работы от количества данных  


Точное значение временной сложности зависит от определения понятия **элементарных операций**, поэтому часто невозможно определить точный вид f(n).

Кроме того, при увеличении размера входных данных **вклад** постоянных множителей и слагаемых низших порядков, фигурирующих в выражении для точного времени работы, становится крайне **незначительным**.

Поэтому достаточно знать какими функциями ограничена реальная сложность f(n). Эти ограничивающие функции задают порядок сложности и называются асимптотической сложностью алгоритма.

Для записи сложности алгоритмов используются такие основные асимптотические обозначения:

* **f** ограничена сверху функцией **g** (с точностью до постоянного множителя) асимптотически (нотация Big O)  
  
* **f** ограничена снизу функцией **g** (с точностью до постоянного множителя) асимптотически  
  
* **f** ограничена снизу и сверху функцией **g** асимптотически  
  

**Примеры вычисления сложности**

Рассмотрим двоичный поиск. Массив разбивается пополам на каждом вызове до тех пор, пока мы не достигнем единицы. Давайте запишем количество элементов в массиве на каждом вызове:

0-я итерация: n

1-я итерация: n / 2

2-я итерация: n / 4

3-я итерация: n / 8

…

i-я итерация: n / 2i

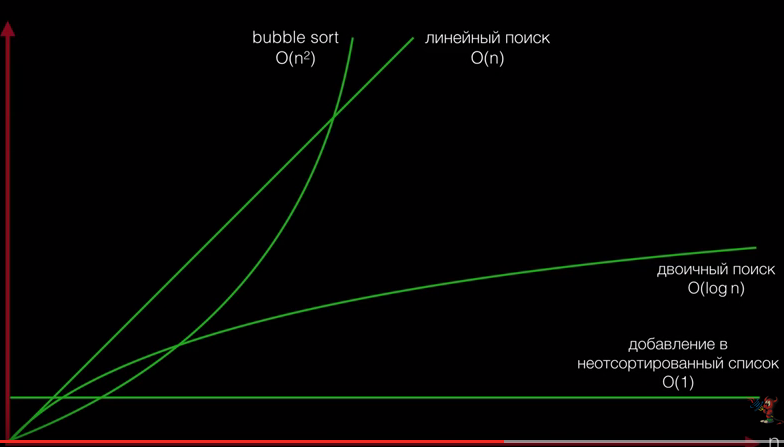
…

последняя итерация: 1

Чтобы узнать, на каком шаге закончится алгоритм нужно решить уравнение:

1 = n /2i; 2i  = n; I = log(n);

Количество итераций, необходимых для бинарного поиска, равняется log(n), то есть сложность двоичного поиска O(log(n)).



Рекурсия

Рекурсия – это определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса. То есть ситуация, когда объект является частью самого себя.

В программировании – это функция которая вызывает сама себя, но с другими аргументами.

1. <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC>
2. <http://habrahabr.ru/post/91605/>
3. <https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C>
4. <https://habrahabr.ru/post/195996/>
5. <https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F>
6. <https://ru.wikibooks.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2>