Алгоритмы

Алгоритм – это набор команд, необходимых для решения той или иной задачи. Определяет шаги, согласно которым она будет выполняться.

# Свойства алгоритма

Хороший алгоритм должен быть:

* Правильным
* Надежным
* Эффективным

Чтобы проверить эффективность алгоритма, программисты задаются вопросом:

*«Как изменится производительность кода, если скорректировать размерность задачи? Если удвоить кол-во переменных в алгоритме, не удвоится ли время его работы? Каков наихудший вариант? Какова вероятность его возникновения? Если запустить алгоритм с использованием большого объема произвольных данных, то какова будет его средняя производительность?»*

## Асимптотическая сложность алгоритма

**Асимптотическая сложность**(производительность) определяется функцией, которая указывает, насколько ухудшается работа алгоритма с усложнением поставленной задачи.

Имеет форму:

O(N) – что означает, по мере увеличения входных данных время работы алгоритма возрастает линейно. Если данных станет в 2 раза больше, то производительность замедлится приблизительно в 2 раза.

Выражение О(N^2) читается как «порядок N в квадрате». Например, алгоритм быстрой сортировки имеет наихудшую производительность порядка N^2.

5 Правил расчета асимптотической сложности алгоритма:

1. Если для математической функции F алгоритму необходимо выполнить определенные действия F(N) раз, то для этого ему понадобится сделать O(F(N)) шагов.
2. Если алгоритм выполняет одну операцию, состоящую из O(F(N)) шагов, а затем вторую операцию, включающую O(G(N)) шагов, то общая производительность алгоритма для функций F и G составит O(F(N) + G(N)).
3. Если алгоритму необходимо сделать O(F(N) + G(N)) шагов и область значений N функции F(N) больше, чем у G(N), то асимптотическую сложность можно упростить до O(F(N)).
4. Если алгоритму внутри каждого шага O(F(N)) одной операции приходится выполнять еще O(G(N)) шагов другой операции, то общая производительность для алгоритма составит O(F(N)\*G(N)).
5. Постоянными множителями(константами) можно пренебречь. Если С является константой, то О(C\*F(N)) или O(F(C\*N)) можно записать как O(F(N)).

Например:

**Boolean: ContainsDuplicates(int [] array)**

**{**

**For (i==0;i<N;i++) // O(N)**

**{**

**For(j==0;j<N;j++) //O(N)**

**{**

**IF(I != j) // Если срабатывает, то O(1)**

**IF(array[i] == array[j]) // Если срабатывает, то O(1)**

**Return True; // Если срабатывает, то O(1)**

**}**

**}**

**Return False; // Если срабатывает, то O(1)**

**}**

**Алгоритм содержит два цикла, один из которых является вложенным. А значит общая производительность алгоритма составит O(N\*N) = O(N2). Пренебрегая дополнительным шагом в выражении Return, предположим что срабатывают оба оператора IF, тогда внутренний цикл будет пройден за O(2N) шагов. Следовательно, общая производительность алгоритма составит O(N\*2N) = O(2N2). Последнее правило позволяет пренебречь коэффициентом 2 и записать производительность алгоритма в виде O(N2).**

Асимптотическая сложность дает представление о теоретическом поведении алгоритма. Практические результаты могут отличаться. Предположим, производительность алгоритма равна O(N); если не пренебрегать целыми числами, реальное количество выполняемых шагов составит приблизительно 100000000+N. В этом случае, пока N не слишком велико, значение константы лучше учитывать.

## Обычные функции рабочего цикла.

Рассмотрим некоторые функции, чтобы иметь общее представление об их уместности:

* **O(1)**

**Алгоритм выполняется за один и тот же промежуток времени, независимо от сложности задачи. Как правило речь идет об ограниченном круге команд, поскольку за O(1) времени невозможно даже просмотреть все входные данные.**

**Например, в какой-то момент, алгоритму быстрой сортировки необходимо выбрать из массива чисел то, которое будет равно среднему значению.**

**Integer: DividingPoint (int[] array){**

**Integer: number1 = array[0];**

**Integer: number2 = array[N];**

**Integer: number3 = array[N/2];**

**IF (<number1 между number2 и number3>) Return number1;**

**IF (<number2 между number2 и number3>) Return number2; Return number3;}**

**Алгоритм выбирает 3 величины (в начале, в конце и в середине массива), сравнивает их между собой и возвращает среднее значение.**

**Поскольку алгоритм выполняет лишь несколько определенных шагов, его производительность равняется O(1), а время**