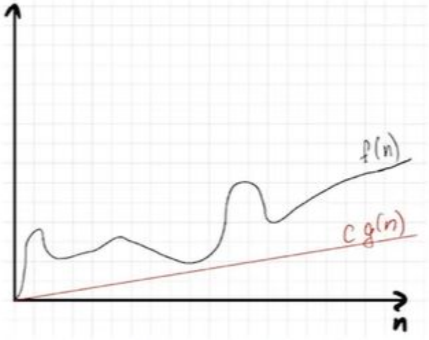
Лекция 10
Анализ алгоритмов. Базовые структуры данных

Немного кванторов

Рассмотрим функции $f, g : \mathbb{N} \to \mathbb{R}_+$

Тогда, мы будем говорить

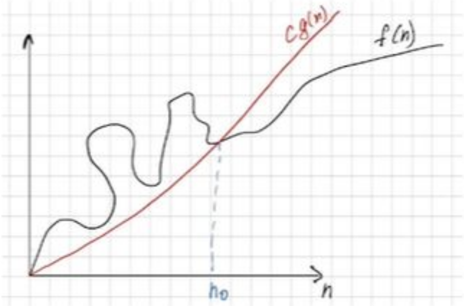
$$\Omega(g(n)) = \{f(n), \exists c > 0, n_0 \in \mathbb{N} : \forall n \geq n_0, 0 \leq cg(n) < f(n)\}$$



Рассмотрим функции $f, g : \mathbb{N} \to \mathbb{R}_+$

Тогда, мы будем говорить

$$\mathcal{O}(g(n)) = \{f(n), \exists c > 0, n_0 \in \mathbb{N} : \forall n \geq n_0, 0 < f(n) < cg(n)\}$$



Рассмотрим функции $f, g : \mathbb{N} \to \mathbb{R}_+$

Тогда, мы будем говорить

$$\Theta(g(n)) =$$
$$\{f(n), \exists c_1 > 0, c_2 > 0, n_0 \in \mathbb{N} : \forall n \geq n_0, 0 \leq c_1 g(n) < f(n) < c_2 g(n)\}$$

Полезные утверждения

Утверждение

Для любых $f, g : \mathbb{N} \to \mathbb{R}_+$ выполено следующее:

$$f(n) = \Theta\big(g(n)\big) \Leftrightarrow \begin{matrix} f(n) = \mathcal{O}\big(g(n)\big) \\ f(n) = \Omega\big(g(n)\big) \end{matrix}$$

В случае, если мы будем встречать алгоритмы, то время их работы можно написать как $\mathcal{O}\big(\log n\big)$.

Мы не пишем основание логарифма, так как $\log_a n = \frac{\log_c n}{\log_c a}$, а $\log_c a$ - константа

Используя асимптотические обозначения, мы пренебрегаем постоянными множителями, не учитывая их при вычислении функций.

При таком подходе функции $f(n) = 0{,}001n^2$ и $g(n) = 1000n^2$ для нас одинаковы, несмотря на то, что значение функции $g(n)$ в миллион раз больше значения функции $f(n)$ для любого n.

RAM – модель вычисления

Модель вычисления определяет:

- допустимые операции
- сложность

в рамках этой модели мы будем считать, что за $\Theta(1)$ выполняются следующие действия:

- Простейшие арифметические операции $(+, -, /, \%, *)$
- Считать и записать 1 число

С помощью RAM-модели можно подсчитать количество шагов, требуемых алгоритму для исполнения любого экземпляра задачи. Но чтобы получить общее представление о том, насколько хорошим или плохим является алгоритм, нам нужно знать, как он работает со всеми экземплярами задачи.

Чтобы понять, что означает наилучший, наихудший и средний случай сложности алгоритма (т. е. время его исполнения в соответствующем случае), нужно рассмотреть исполнение алгоритма на всех возможных экземплярах входных данных. В случае задачи сортировки множество входных экземпляров состоит из всех возможных компоновок ключей *n* по всем возможным значениям *n*.

Оценка рекуррентных соотношений (мастер-теорема)

Пусть $a \geq 1$ и $b > 1$ - константы, $f(n)$ - функция, а $T(n)$ - опредлена на множестве неотрицательных чисел с помощью рекуррентного отношения

$$T(n) = aT(n/b) + f(n)$$

Тогда $T(n)$ имеет следующие асимптотические границы:

- Если $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ для некоторой константы $\epsilon > 0$, то $T(n) = \Theta(n^{\log_b a})$
- Если $f(n) = \Theta(n^{\log_b a})$, то $T(n) = \Theta(n^{\log_b a} \log n)$
- Если $f(n) = \Omega(n^{\log_b a + \epsilon})$ для некоторой константы $\epsilon > 0$ и если $af(n/b) \leq cf(n)$ для некоторой константы $c < 1$ и для всех достаточно больших n, то $T(n) = \Theta(f(n))$

Простые структуры данных

Массив, список, стек, дек, очередь

Полезные ссылки

https://academy.yandex.ru/handbook/python/article/volshebnye-metody-pereopredelenie-metodov-nasledovanie