Оценка сложности О-нотация

1 Основные обозначения в курсе

- $\forall$ — «для каждого», «для любого», «для всех»
- $\exists$ — «существует», «найдется»
- : — «такой, что», «такие, что»
- $\rightarrow$ — «выполняется»
- $\Rightarrow$ — «следует»

Лекция опирается на книгу [1], в частности на некоторые определения.

2 Методы сравнения программ

- Время работы
- Количеством использованной памяти

Как можно сравнивать программы решающие одну и туже задачу? Первое что хочется сказать это сколько времени программа отрабатывает на компьютере, но это не совсем так, так как вместе с вашей программой на нем работает еще очень много других программ. Для этого будем использовать не просто время работы программы, а время которое затратил процессор на нашу программу. Дальше всегда будем предполагать, что время работы программы это время затраченной процессором на выполнения только нашей программы.

От чего зависит время работы программы? Конечно же время работы программы зависит от мощности компьютера и тому подобное, но так-как мы договорились, что время работы это количество времени которое затратит процессор, то здесь уже явная зависимость от компьютера отпадает. На самом деле нас интересует зависимость времени работы программы от входных данных, а точнее от их размера.

В дальнейшем, чтобы отойти немного от компьютера введем понятия алгоритм.

Определение 1: Алгоритм - это некоторая последовательность действий, которая преобразует начальные данные (input data) в выходные данные (output data).

Введя понятия алгоритм мы будем сравнивать не программы выполняемые на компьютере а некоторые математические модели.

Для простоты сравнения алгоритмов рассматривается асимптотическое сравнение двух алгоритмов. То есть как ведет себя алгоритм при изменении размера входных данных.

Определение 2: Говорят, что $f(n) \in O(g(n))$, если

$$\exists C, N : \forall n > N \to f(n) < Cg(n), \tag{1}$$

Определение 3: Говорят, что $f(n) \in \Omega(g(n))$, если

$$\exists C, N : \forall n > N \to f(n) > Cg(n), \tag{2}$$

Определение 4: Говорят, что $f(n) \in \Theta(g(n))$, если $f(n) \in \Omega(g(n))$ и $f(n) \in O(g(n))$.

Введем некоторые функции, которые будут полезны для оценки сложности алгоритмов. Первой функцией является функция времени работы алгоритма от размера входа. Второй функцией является функция памяти используемой алгоритмом от размера входа.

Определение 5: $T(n)$ — это время работы алгоритма на входе (input data) длины n, где под длинной n подразумевается длина двоичной записи входных данных.

Определение 6: $M(n)$ — это объем дополнительной памяти требуемой алгоритмом на входе (input data) длины n, где под длинной n подразумевается длина двоичной записи входных данных. Под дополнительной памятью подразумевается вся память кроме входа.

3 Примеры асимптотик

3.1 Задача

$sum = 0$
for $0 < i < n$ *do*
....$sum+ = 1$

Пусть время затраченное на прибавление единицы равно t. Найти асимптотическое время работы алгоритма $T_1(n)$ для этой задачи.

$$T_1(n) = \Theta(n \cdot t)$$

3.2 Задача

$sum = 0$
for $0 < i < n$ *do*
....for $0 < j < n$ *do*
.....$sum+ = 1$

Пусть время затраченное на прибавление единицы равно t. Найти асимптотическое время работы алгоритма $T_2(n)$ для этой задачи.

$$T_2(n) = \Theta(n^2 \cdot t)$$

3.3 Задача

def f(char*str)
....if(len(str) > 0)
.....return f($str + 1$)
....else
.....return 0

Пусть время затраченное на прибавление единицы к индексу строки равно t_1, время нахождения длины строки равно t_2. Найти асимптотическое время работы алгоритма $T_3(n)$ для этой задачи, где n это длина строки.

$$T_3(n) = O(n \cdot (t_1 + t_2))$$

3.4 Задача

$min = 2^{32}$
arr — массив размера n, где все числа меньше 2^{32}
for $0 < i < n$ *do*
....if($arr[i] < min$)
.....$min = arr[i]$

Пусть время затраченное на сравнение двух чисел равно t. Найти асимптотическое время работы алгоритма $T_4(n)$.

$$T_4(n) = O(n \cdot t)$$

3.5 Задача

arr — массив размера n
for $0 < i < n$ *do*
....for $0 < j < n$ *do*
.....if($arr[j] > arr[j + 1]$)
.....swap($arr[j], arr[j + 1]$)

Пусть время затраченное на сравнение двух чисел равно t_1, а время затрачиваемое на перемещения двух рядом стоящих элементов массива это t_2. Найти асимптотическое время работы алгоритма $T_5(n)$.

$$T_5(n) = O(n^2 \cdot (t_1 + t_2))$$

Список литературы

[1] *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. 3-е издание. Москва, 2013.