Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 5

ПРИМЕРЫ АНАЛИЗА ТРУДОЕМКОСТИ И ЕМКОСТНОЙ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, 8 916 589 94 04

Трудоемкость конструкции «Цикл по счетчику»

Запишем цикл **For** в элементарных операциях нашей модели вычислений:

```
For i ← 1 to n
   Тело цикла
end For i
```

```
   i ← 1

Next   Тело цикла

   i ← i + 1
   If i <= n then Next
```

Инициализация цикла требует 1 операцию i ← 1, и на каждом проходе цикла выполняется i ← i + 1 и If i <= n, что дает 3 операции, итого — **1+3n** элементарных операций на обслуживание цикла **For.**

Пример 0 — Алгоритм нахождения суммы элементов массива

Сумма элементов массива:

```
Sum (A, n; S)
S ← 0                           1
For i ← 1 to n                  1+3n
   S ← S + A[i]                 3
end For i
Return (S)
End.
```

Трудоемкость определяется следующей формулой

$$f_A(n) = 1 + 1 + n(3 + 3) = 6n + 2 = \Theta(n).$$

Вопрос 1 — можно ли для этой задачи улучшить асимптотическую оценку?

Вопрос 2 — можно ли предложить алгоритм (в данной модели вычислений), мультипликативная константа которого меньше, чем 6?

Пример 1 — Алгоритм умножения квадратных матриц

Алгоритм умножения двух квадратных матриц. Запись этого классического алгоритма имеет вид:

```
MultM (A, B, n; C)
  For i ← 1 to n                        1+3n
    For j ← 1 to n                      1+3n
      Sum ← 0                              1
      For k ← 1 to n                    1+3n
        Sum ← Sum + A[i,k]*B[k,j]          7
      end For k
      C[i,j] ← Sum                         3
    end For j
  end For i
Return (C)
End.
```

Все циклы не зависят друг от друга, что позволяет непосредственно применить формулу для трудоемкости конструкции «Цикл по счетчику», в результате, с учетом трудоемкости в строках, получаем

$$f_A(n) = 1 + n\big(3 + 1 + n(3 + 1 + 1 + n(3 + 7) + 3)\big) = 10n^3 + 8n^2 + 4n + 1.$$

$$f_A(n) = \Theta(n^3).$$

Здесь n — длина входа, линейная размерность матрицы — общепринятое соглашение при анализе алгоритмов умножения квадратных матриц. Входом алгоритма являются два массива, содержащие по n^2 элементов, такой же массив хранит вычисленный результат и алгоритм требует четыре ячейки для счетчиков циклов и хранения суммы, и одну ячейку для хранения n, таким образом:

$$V_A(n) = 3n^2 + 5.$$

Ресурсная сложность алгоритма задается формулой

$$\mathfrak{R}_{\mathrm{c}}(A) = \langle \Theta(n^3),\ \Theta(n^2) \rangle.$$

Пример 2 — Алгоритм вычисления расстояний между n точками.

Алгоритм вычисляет матрицу квадратов расстояний между заданными точками. Вход: массивы координат точек X и Y и их размер. Результат — матрица S, размерностью $n \times n$ элементов. Запись алгоритма имеет вид:

```
Lcalc (X, Y, n; S)
  For i ← 1 to n                   1+3n
    S[i,i]← 0                      3
    For j ← i+1 to n               2+3n
      dx ← (X[i]–X[j])             4
      dy ← (Y[i]–Y[j])             4
      S[i,j] ← dx*dx + dy*dy       6
      S[j,i] ← S[i,j]              5
    end For j
  end For i
Return (S)
End.
```

В этом алгоритме внутренний цикл зависит от внешнего, поэтому для получения функции трудоемкости нам необходимо просуммировать количество проходов внутреннего цикла $(n - i)$ при изменении внешнего (i), в результате получаем

$$\sum_{i=1}^{n}(n - i) = n^2 - \frac{n(n+1)}{2} = \frac{n^2 - n}{2}.$$

На основе полученного результата, применяя сведение конструкции цикла по счетчику к базовым операциям, получаем трудоемкость этого алгоритма

$$f_A(n) = 1 + n(3 + 3 + 2) + \frac{n^2 - n}{2}(3 + 4 + 4 + 6 + 5) = 11n^2 - 3n + 1 = \Theta(n^2).$$

Вход два одномерных массива — **`X, Y`**, и **`n,`** выход алгоритма — двумерный массив **`S`**, дополнительные ячейки — **`i, j, dx, dy`**,

$V_A(n) = n^2 + 2 \cdot n + 5$, а ресурсная сложность имеет вид

$$\Re_c(A) = \langle \Theta(n^2),\ \Theta(n^2) \rangle.$$

Пример 3 — Алгоритм возведения в целую степень

Задача о возведении числа в целую степень — это задача с фиксированной длиной входа. Мы вычисляем $y = x^m$ для целого неотрицательного значения m. Рассмотрим примитивное решение этой задачи, которое использует последовательное умножение. Запись алгоритма Pow_A1 имеет вид:

Pow_A1 (x, m; y)	
y ← 1	1
For i ← 1 **to** m	1+3m
y ← y*x	2m
Return (y)	
End.	

Поскольку операция умножения является базовой, то значение x не влияет на трудоемкость и

$$f_{A1}(D) = f_{A1}(x, m) = f_{A1}(m) = 5m + 2,$$

Значение m является параметром — это значение одной из ячеек входа.

Пример 4 — Алгоритм поиска максимума в массиве

Рассмотрим задачу поиска максимального элемента и его индекса в массиве. Запись стандартного алгоритма, решающего эту задачу, в принятом алгоритмическом базисе имеет вид (справа в строке приведено задаваемое этой строкой число базовых операций):

```
Max_A0 (S, n; Max, iMax)
  Max ← S[1]                        2
  iMax ← 1                          1
  For i ← 2 to n                    1+3(n-1)
    If Max < S[i]                   2)
      then
        Max ← S[i]                  2
        iMax ← i                    1
  end For
Return (Max, iMax)
End.
```

Входом данного алгоритма является массив из n элементов — а трудоемкость алгоритма зависит как от количества чисел в массиве, так и от порядка их расположения Полный анализ для фиксированной размерности множества исходных данных предполагает получение функций трудоемкости для худшего, лучшего и среднего случаев. Для данного алгоритма фрагментом, определяющим параметрическую зависимость трудоемкости, является блок **`then`**, содержащий строки `Max ← S[i]` и `iMax ← i`. Обозначим общее количество выполнений этого блока, задаваемое алгоритмом **`Max_A0`**, через m, заметим, что сам блок содержит три операции.

Худший случай. Значение m будет максимально, а именно $m = n - 1$, когда на каждом проходе цикла выполняется переприсваивание текущего максимума — это ситуация, когда элементы исходного массива различны и отсортированы по возрастанию. В этом случае трудоемкость алгоритма равна:

$$f_A^{\wedge}(n) = 2 + 1 + 1 + (n-1)(3+2) + (n-1)(3) = 8n - 4 = \Theta(n).$$

Лучший случай. Значение m будет минимально, и равно нулю ($m = 0$), в том случае, если первый элемент массива является максимальным. Трудоемкость алгоритма в этом случае равна:

$$f_A^{\vee}(n) = 2 + 1 + 1 + (n-1)(3+2) = 5n - 1 = \Theta(n)$$

Расширенная задача — найти минимальный и максимальный элемент в массиве чисел, речь идет как о значениях соответствующих элементах массива, так и об их индексах. Эта задача достаточно часто встречается как вспомогательная в целом ряде задач статистической обработки данных, сортировки, при работе с матрицами и т. д.

Вопрос — как найти минимальный и максимальный элементы с почти таким же коэффициентом — $8n$ в худшем случае?