

## Тема 2

# Оценка структурной и временной сложности алгоритмов и программ

Черников Б.В.

1

## Временная сложность алгоритма. O-символика

Временная сложность алгоритма зависит от количества входных данных

Обычно говорят, что временная сложность алгоритма имеет порядок  $T(n)$  от входных данных размера  $n$

Точно определить величину  $T(n)$  на практике представляется довольно трудно

Поэтому прибегают к асимптотическим отношениям с использованием O-символики

Если число тактов (действий), необходимое для работы алгоритма, выражается как  $11n^2 + 19n - \log n + 3n + 4$ , то это алгоритм, для которого  $T(n)$  имеет порядок  $O(n^2)$

Фактически, из всех слагаемых **оставляется только то, которое вносит наибольший вклад при больших  $n$**  (в этом случае остальными слагаемыми можно пренебречь), и игнорируется коэффициент перед ним

Черников Б.В.

2

Когда используют обозначение  $O()$ , имеют в виду не точное время исполнения, а только **его предел сверху**, причем с точностью **до постоянного множителя**

Когда говорят, например, что алгоритму требуется время порядка  $O(n^2)$ , имеют в виду, что время исполнения задачи растёт не быстрее, чем квадрат количества элементов

| $n$        | $\log n$ | $n \cdot \log n$ | $n^2$               |
|------------|----------|------------------|---------------------|
| 1          | 0        | 0                | 1                   |
| 16         | 4        | 64               | 256                 |
| 256        | 8        | 2 048            | 65 536              |
| 4 096      | 12       | 49 152           | 16 777 216          |
| 65 536     | 16       | 1 048 565        | 4 294 967 296       |
| 1 048 476  | 20       | 20 969 520       | 1 099 301 922 576   |
| 16 775 616 | 24       | 402 614 784      | 281 421 292 179 456 |

Черников Б.В.

3

Если операция выполняется за фиксированное число шагов, не зависящее от количества данных, то принято писать  $O(1)$

Основание логарифма здесь не пишется

Пусть есть  $O(\log_2 n)$ .

Однако  $\log_2 n = \log_3 n / \log_3 2$ , а  $\log_3 2$ , как и любую константу, символ  $O()$  не учитывает. Поэтому  $O(\log_2 n) = O(\log_3 n)$

Время выполнения алгоритма зависит не только от количества входных данных, но и от их значений

Чтобы учитывать этот факт, сохраняя при этом возможность анализировать алгоритмы независимо от данных, различают:

- ♦ **максимальную сложность**  $T_{\max}(n)$  – сложность наиболее неблагоприятного случая, когда алгоритм работает дольше всего
- ♦ **среднюю сложность**  $T_{\text{mid}}(n)$  – сложность алгоритма в среднем
- ♦ **минимальную сложность**  $T_{\min}(n)$  – сложность в наиболее благоприятном случае, когда алгоритм справляется быстрее всего

Черников Б.В.

4

### Базовые принципы оценки временной сложности алгоритма :

- ♦ Время выполнения операций присваивания, чтения, записи обычно имеют порядок  $O(1)$ . **Исключение** – когда операнды представляют собой массивы или вызовы функций
- ♦ Время выполнения **последовательности операций** совпадает с наибольшим временем выполнения операции в ней (правило сумм – если одна операция имеет порядок  $O(f(n))$ , а другая – порядок  $O(g(n))$ , то общее время будет иметь порядок  $O(\max(f(n), g(n)))$ )
- ♦ Время выполнения **конструкции ветвления (if-then-else)** состоит из времени вычисления логического выражения (обычно имеет порядок  $O(1)$ ) и **наибольшего** из времени, необходимого для выполнения операций, исполняемых при **истинном значении** логического выражения и при **ложном значении** логического выражения
- ♦ Время выполнения **цикла** состоит из времени вычисления **условия** прекращения цикла (обычно имеет порядок  $O(1)$ ) и произведения количества выполненных итераций цикла на **наибольшее возможное время выполнения операций тела цикла**
- ♦ Время выполнения операции **вызова процедур** определяется как **время выполнения вызываемой процедуры**
- ♦ При наличии в алгоритме операций **безусловного перехода** необходимо учитывать изменения последовательности операций, осуществляемых с использованием этих операций безусловного перехода

Черников Б.В.

5

## Понятие структурной сложности



Разнообразие поведения алгоритма (программы) и связей между его входными и результирующими данными определяется **набором маршрутов** (чередующихся последовательностей вершин и дуг графа управления), по которым он исполняется

Сложность программного модуля связана не столько с размером (числом команд) программы, сколько с **числом маршрутов ее исполнения и их сложностью**

Черников Б.В.

6

Маршруты возможной обработки данных определяют сложность разработки программы

Данную метрику сложности можно использовать для оценки трудоемкости тестирования и сопровождения модуля, а также для оценки потенциальной надежности его функционирования

Маршруты исполнения программного модуля

Вычислительные маршруты

Маршруты принятия логических решений

Черников Б.В.

7

Сложность вычислительных маршрутов оценивается формулой

$$S_1 = \sum_{i=1}^m \sum_{j=1}^{l_i} v_j$$

$m$  – количество маршрутов исполнения программы

$l_i$  – число данных обрабатываемых в  $i$ -ом маршруте

$v_j$  – число значений обрабатываемых данных  $j$ -го типа ( $2 \leq v_j \leq 5$ )

Общее число арифметических операций не выходит за пределы 5–10%, поэтому вычислительные маршруты не определяют структурную сложность программ

Сложность маршрутов принятия логических решений оценивается формулой

$$S_2 = \sum_{i=1}^m p_i$$

$p_i$  – число ветвлений или число проверяемых условий в  $i$ -ом маршруте

Общая сложность программы

$$S = c \cdot \sum_{i=1}^m \left( \sum_{j=1}^{l_i} v_j + p_i \right)$$

$c$  – коэффициент пропорциональности

Черников Б.В.

8

## Критерии выделения маршрутов

Наилучший критерий позволит выделить все реальные маршруты исполнения программы при любых сочетаниях исходных данных



### Критерий 1

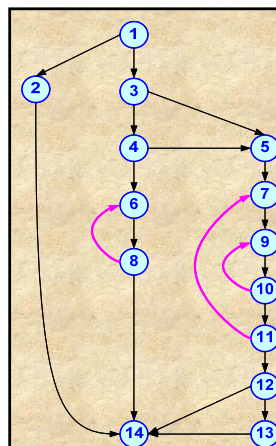
Граф программы по управлению должен быть покрыт минимальным набором путей, проходящих через каждый оператор ветвления по каждой дуге. Повторная проверка дуг не оценивается и считается избыточной. В процессе проверки гарантируется выполнение всех передач управления между операторами программы и каждого оператора не менее одного раза

Поток управления – последовательность выполнения различных модулей и операторов программы

Граф потока управления – ориентированный граф, моделирующий поток управления программой

Черников Б.В.

9



Рассмотрим структурную сложность типичной программы управления, содержащей 100 – 200 команд, граф потока управления которой содержит 14 вершин, 20 дуг и 3 цикла

Черников Б.В.

10



Черников Б.В.

11



### Критерий 2

Основан на анализе базовых маршрутов в программе, формируемых и оцениваемых на основе цикломатического числа графа потока управления

Цикломатическое число исходного графа программы

$$Z = m - n + 2 \cdot p$$

$m$  – общее число дуг в графе;  $n$  – общее число вершин в графе;

$p$  – число связанных компонентов графа

Число связанных компонентов графа  $p$  равно количеству дуг, необходимых для превращения исходного графа в максимально связанный (или полностью связанный) граф

Максимально связанным (полностью связанным) называется граф, у которого любая вершина доступна из любой другой вершины

Максимально связанный (полностью связанный) граф получается из исходного графа путем замыкания конечных и начальной вершин

Черников Б.В.

12

Для рассматриваемого графа **полносвязный граф** получается путем добавления дуги (14, 1)

$Z = m - n + 2 \times p = 20 - 14 + 2 \times 1 = 8$

Критерий 2 (по цикломатическому числу) требует однократной проверки или тестирования **каждого линейно-независимого цикла** и **каждого линейно-независимого ациклического участка** программы

Каждый линейно-независимый ациклический маршрут или цикл **отличается от всех остальных** хотя бы одной вершиной или дугой

При использовании критерия 2 количество проверяемых маршрутов равно **цикломатическому числу**

Линейно-независимые **циклические** маршруты

- m1: 6 - 8
- m2: 9 - 10
- m3: 7 - 9 - 10 - 11

Линейно-независимые **ациклические** маршруты

- m4: 1 - 2 - 14
- m5: 1 - 3 - 4 - 6 - 8 - 14
- m6: 1 - 3 - 5 - 7 - 9 - 10 - 11 - 12 - 14
- m7: 1 - 3 - 4 - 5 - 7 - 9 - 10 - 11 - 12 - 13 - 14
- m8: 1 - 3 - 4 - 5 - 7 - 9 - 10 - 11 - 12 - 14

Черников Б.В. 3

**Определение структурной сложности по второму критерию**

m1: 6 - 8 = 1

m2: 9 - 10 = 1

m3: 7 - 9 - 10 - 11 = 2

m4: 1 - 2 - 14 = 1

m5: 1 - 3 - 4 - 6 - 8 - 14 = 4

m6: 1 - 3 - 5 - 7 - 9 - 10 - 11 - 12 - 14 = 5

m7: 1 - 3 - 4 - 5 - 7 - 9 - 10 - 11 - 12 - 14 = 6

m8: 1 - 3 - 4 - 5 - 7 - 9 - 10 - 11 - 12 - 13 - 14 = 6

**Итого - 26**

Для правильно структурированных программ цикломатическое число **Z** можно определить путем подсчета числа **вершин  $n_v$** , в которых происходит **ветвление**

$Z = n_v + 1$

В нашем примере  $Z = 7 + 1 = 8$  (ветвления имеют место в вершинах графа 1, 3, 4, 8, 10, 11, 12)

Черников Б.В. 14

**Исследования графов реальных программных модулей с достаточно большим фиксированным числом вершин показали:**

- Суммарная сложность тестов почти не зависит от детальной структуры графа и в основном определяется числом предикатов - ветвлений графа
- При неизменном числе вершин в широких графах имеется большее количество маршрутов, чем в узких графах, но маршруты в среднем короткие. В узких графах число маршрутов сокращается по сравнению с широкими, но маршруты становятся длиннее. В результате величина  $S_2$  при изменении структуры графов изменяется меньше, чем цикломатическое число и сильнее коррелирована с числом вершин

В.В. Липаев показал:

- для  $Z \leq 10$  модули корректно проверяемы и число ошибок в таких модулях будет минимальным
- приемлемыми значениями считаются  $10 \leq Z \leq 30$
- при  $Z > 30$  устранить ошибки в процессе тестирования практически невозможно

Черников Б.В. 15

Для автоматического анализа графов по второму критерию с помощью ЭВМ используются **матрицы смежности и достижимости графов**

**Матрица смежности** - это квадратная матрица, в которой 1 располагается в позиции (i, j), если в графе имеется дуга (i, j)

Черников Б.В. 16

**Матрица достижимости** - это квадратная матрица, в которой 1 располагается в позиции, соответствующей дуге (i, j), причем 1 ставится для всех вершин, до которых можно «достать»

Черников Б.В. 17

**Критерий 3**

Основан на выделении полного состава базовых структур графа программы и заключается в анализе хотя бы один раз каждого из реальных ациклических маршрутов исходного графа программы и каждого цикла, достижимого из всех этих маршрутов

Если из некоторого ациклического маршрута исходного графа достижимы несколько элементарных циклов, то при тестировании должны исполняться все достижимые циклы

Черников Б.В. 18





### Выводы по критериям оценки сложности программ

- Критерии для оценки сложности программных модулей характеризуют в каждом случае **минимально необходимые величины проверок по каждому критерию**
- Для проверки реальных программ это количество проверок может быть **недостаточно** (например, циклы желательно проверять на одном-двух промежуточных значениях, а также на максимальном и минимальном количествах исполнения циклов)
- Оценить **достаточность проверок программы** значительно труднее, так как при этом, кроме сложности структуры, необходимо анализировать сложность преобразования каждой переменной во всем диапазоне ее изменения и при сочетаниях с другими переменными

Черников Б.В.

20

### Метрика Маккейба

Основана на анализе потока передачи управления от одного оператора к другому, что позволяет **учесть логику программы**. Программа (алгоритм, спецификация) должна быть представлена в виде управляющего ориентированного графа

$$G = (V, E)$$

с  $V$  вершинами и  $E$  дугами, где вершины соответствуют операторам, а дуги – переходам от одного оператора к другому

Граф, описывающий программу в виде вершин-операторов и дуг-переходов, называют **графом управления** или **управляющим графом программы**

Обычно учитывают только **исполнимые операторы**, исключая операторы описания данных. Линейные участки программ можно заменить одним узлом графа. Желательно преобразовать операторы цикла в эквивалентную последовательность операторов ветвления, добавив операторы суммирования (счетчики) числа повторений цикла с «верхним» или «нижним» окончанием

Черников Б.В.

21

Метрика Маккейба является **цикломатическим числом графа управления программы**

$$M = m - n + 2$$

$m$  – количество ребер графа  
 $n$  – количество вершин графа

Величину  $M$  называют **цикломатическим числом Маккейба**

**Цикломатическая сложность программы – структурная (или топологическая) мера сложности программ**, используемая для измерения качества программного обеспечения, основанная на методах статического анализа кода

Цикломатическая сложность программы равна **увеличенному на единицу цикломатическому числу графа программы**

Разработана Томасом Дж. Маккейбом в 1976 году

Черников Б.В.

22

При вычислении цикломатической сложности используется **граф потока управления программы**: узлы графа соответствуют неделимым группам команд программы и ориентированным ребрам, каждый из которых соединяет два узла и соответствует двум командам, вторая из которых может быть выполнена сразу после первой

Эта стратегия тестирования называется **основным маршрутом тестирования Маккейба**: **тестирование каждого линейного независимого маршрута через программу** – в этом случае **число тестов должно быть равно цикломатической сложности программы**

Цикломатическая сложность части программного кода – **счетное число линейно независимых маршрутов** через программный код

Исходный код не содержит никаких точек решений (IF, FOR)

Сложность = 1 (есть только один маршрут)

Код имеет единственный оператор IF, содержащий простое условие

Два пути: один – через IF как TRUE и один – как FALSE

Черников Б.В.

23

В теории графов цикломатическое число ориентированного графа

$$Z = m - n + 2 \times p$$

$m$  – количество ребер  
 $n$  – количество вершин  
 $p$  – количество компонентов связности графа

Число компонентов связности  $p$  можно рассматривать как минимально необходимое количество ребер, которые нужно добавить к графу, чтобы сделать его **полносвязным**

Справедливо считать, что для любого графа управления программы число компонентов связности равно единице ( $p = 1$ )

Подстановка  $p = 1$  в формулу определения цикломатического числа дает **цикломатическое число Маккейба** – определяет количество независимых контуров в полносвязном графе и, как следствие, количество различных путей, ведущих из начальной вершины в конечную

При оценке сложности программы с использованием цикломатического числа Маккейба действует правило: **если цикломатическое число  $Z > 10$ , программа обладает излишней сложностью и ее следует разбить на составные части с меньшим значением цикломатического числа**

Черников Б.В.

24