# Практическая работа № 4. Оценка сложности и отладка

# математических алгоритмов

**Цель практической работы**

Закрепление теоретических знаний по оценке сложности и отладке математических алгоритмов.

**Постановка задачи**

**Сложность алгоритма** — это функция, описывающая зависимость времени выполнения или объема используемой памяти от размера входных данных. Оценивается в терминах асимптотического поведения при стремлении n → ∞.

**Виды сложности**

1. **Временная сложность (Time Complexity)**
   * Определяет количество **элементарных операций**, выполняемых алгоритмом.
   * Измеряется в **"О-большом"** (асимптотическая верхняя граница).
2. **Пространственная сложность (Space Complexity)**
   * Определяет объем **памяти**, используемой алгоритмом.
   * Учитывает дополнительную память (кроме входных данных).

**Асимптотические обозначения**

Для формализации оценки сложности используются:

* **O(f(n))** — верхняя граница ("не хуже, чем").
* **Ω(f(n))** — нижняя граница ("не лучше, чем").
* **Θ(f(n))** — точная оценка (совпадение O и Ω).

| **Сложность** | **Описание** | **Пример алгоритма** |
| --- | --- | --- |
| **O(1)** | Константное время | Доступ к элементу массива |
| **O(log n)** | Логарифмическое время | Бинарный поиск |
| **O(n)** | Линейное время | Поиск в неотсорт. массиве |
| **O(n log n)** | Линейно-логарифмическое время | Быстрая сортировка |
| **O(n²)** | Квадратичное время | Пузырьковая сортировка |
| **O(2ⁿ)** | Экспоненциальное время | Перебор всех подмножеств |

**Отладка** — это процесс выявления, локализации и устранения ошибок (дефектов) в программном коде с использованием специализированных инструментов и методик.

**Ключевые аспекты:**

* **Диагностика** (анализ поведения программы);
* **Верификация** (проверка соответствия ожидаемому результату);
* **Коррекция** (исправление ошибок).

**Инструменты:**

* **Отладчики** (Debugger, например, в Visual Studio);
* **Профилировщики** (для анализа производительности);
* **Логирование** (запись состояния программы).

**Точка останова** — это маркер в коде, при достижении которого выполнение программы приостанавливается, позволяя разработчику исследовать:

* **значения переменных**,
* **стек вызовов**,
* **состояние памяти**.

**Виды точек останова:**

* **Условные** (срабатывают при выполнении условия);
* **По количеству попаданий** (активируются после N-ного вызова);
* **На данные** (при изменении значения переменной).

**Стек вызовов** — это структура данных, отражающая последовательность вызовов методов, приведших к текущей точке выполнения программы.

**Использование при отладке:**

* Позволяет отследить **путь выполнения**;
* Выявляет **рекурсивные вызовы**;
* Помогает находить **источники исключений**.

**Пошаговое выполнение** — это режим отладки, при котором программа выполняется по одной инструкции за раз.

**Основные команды:**

* **Step Into (F11)** — вход в вызываемый метод;
* **Step Over (F10)** — выполнение текущего метода без входа в вызываемые;
* **Step Out (Shift+F11)** — выход из текущего метода.

**Окно просмотра переменных** — инструмент отладчика, отображающий текущие значения переменных и выражений в режиме реального времени.

**Функциональность:**

* **Изменение значений** переменных во время отладки;
* **Вычисление выражений**;
* **Отслеживание сложных структур данных**.

**Задание на практическую работу**

1. Разработать консольное приложение для выполнения заданий.
2. Определить сложность алгоритмов.
3. Провести отладку алгоритмов.
4. Предусмотреть понятный, аккуратный интерфейс ввода и вывода данных на консоль.

**Задание 1.**

**Анализ сложности алгоритма "Решето Эратосфена"**

* + Реализовать алгоритм поиска всех простых чисел до заданного N.
  + Определить временную и пространственную сложность.
  + Провести отладку и выписать значения массива на каждой итерации внешнего цикла.

**Задание 2.**

**Оценка сложности быстрого возведения в степень**

* + Реализовать итеративный алгоритм бинарного возведения в степень.
  + Доказать логарифмическую сложность O(log n).
  + Провести отладку и выписать текущие значения основания, степени и результата на каждом шаге.

**Задание 3.**

**Сложности алгоритма сортировки**

* + Реализовать пузырьковую сортировку для одного массива.
  + Провести отладку и выписать состояние массива после каждой итерации (первые 5 элементов).

**Задание 4.**

**Отладка алгоритма поиска НОД (алгоритм Евклида)**

* + Реализовать итеративный и рекурсивный варианты.
  + Определить сложность в худшем случае.
  + Провести отладку и выписать значения входных параметров и промежуточных результатов на каждом шаге.

**Пример разработки приложения**

**Задание.**

Решето Эратосфена — древний алгоритм для нахождения всех простых чисел до заданного предела N. Работает следующим образом:

1. Создаётся список чисел от 2 до N.
2. Первое непомеченное число (2) объявляется простым.
3. Все кратные этому числу помечаются как составные.
4. Процесс повторяется для следующего непомеченного числа.
5. Алгоритм завершается, когда квадрат текущего числа превышает N.

**Шаг 1. Реализация на C#**

Console.Write("Введите N: ");

int n = int.Parse(Console.ReadLine());

bool[] isPrime = new bool[n + 1];

for (int i = 2; i <= n; i++)

{

isPrime[i] = true;

}

for (int p = 2; p \* p <= n; p++)

{

if (isPrime[p])

{

for (int i = p \* p; i <= n; i += p)

{

isPrime[i] = false;

}

}

}

Console.WriteLine("Простые числа до {0}:", n);

for (int i = 2; i <= n; i++)

{

if (isPrime[i])

{

Console.Write(i + " ");

}

}

**Шаг 2. Анализ сложности**

1. **Временная сложность**:
   * Внешний цикл: O(√n) (до √n)
   * Внутренний цикл: для каждого простого p выполняется n/p операций
   * Общая сложность: O(n log log n) (доказательство через гармонический ряд простых чисел)
2. **Пространственная сложность**:
   * O(n) (используется массив размера n)

**Шаг 3. Установка точек останова**

Установим точку останова

bool[] isPrime = new bool[n + 1]; // Точка 1 (инициализация)

for (int p = 2; p \* p <= n; p++) // Точка 2 (начало внешнего цикла)

{

if (isPrime[p]) // Точка 3 (проверка простоты)

{

for (int i = p \* p; i <= n; i += p) // Точка 4 (начало внутреннего цикла)

{

isPrime[i] = false; // Точка 5 (вычеркивание)

}

}

}

**Шаг 4. Запуск отладки (F5)**

1. Введите значение N = 30
2. Программа остановится на первой точке останова

**Шаг 5. Пошаговое выполнение**

Используйте:

* **F10** (Step Over) - выполнить текущую строку
* **F11** (Step Into) - войти в метод
* **Shift+F11** (Step Out) - выйти из метода

**Шаг 6. Анализ состояния программы**

При остановке проверяйте **локальные переменные** (окно Locals):

* + Текущее значение p
  + Состояние массива isPrime

**Шаг 7. Конкретный пример отладки для N=10**

1. После инициализации (Точка 1):
   * isPrime = [F,F,T,T,T,T,T,T,T,T,T] (индексы 0-10)
2. Первая итерация (p=2):
   * Точка 3: isPrime[2] = true → заходим в цикл
   * Точка 5: вычеркиваем 4,6,8,10
   * Состояние: isPrime[4]=false, isPrime[6]=false и т.д.
3. Вторая итерация (p=3):
   * isPrime[3] = true → вычеркиваем 9
   * (p=4 пропускаем, т.к. isPrime[4]=false)

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»