Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

Выполнил студент группы КС-36 (Потапов Никита Александрович)

Ссылка на репозиторий: (Ссылка на лабораторную в репозитории)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: (12.05.2025)

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

*Метод Монте Карло*

Пусть 2 игрока A и B играют в следующую игру: у игроков есть монетка, где 0 это орел, а 1 это решка, каждый игрок выбирает комбинацию из 3 цифр 0/1(например 001), затем подбрасывается монетка и результат записывается в длинную строку, побеждает тот чъя комбинация будет на конце итоговой строки. Например: A - 001, B - 100, R - 01010101010100, победил B.

Необходимо смоделировать игру двух игроков и:

* Построить таблицу вероятности выигрыша одной из комбинаций, так что бы столбцы соответствовали игроку А, а строки игроку Б, а на пересечении была бы вероятность побед игрока А над игроком Б, при выбранных ими комбинациях.
* Так же по результатом всех попыток определить суммарный средний шанс выигрыша игрока А и игрока Б вне зависимости от выбранных комбинаций
* Проанализировать полученные результаты

# Описание метода/модели.

1. Описание задачи

Имитируется игра двух игроков, выбирающих комбинации из трех битов (например, «001», «110»). Монетка подбрасывается многократно, формируя последовательность из «0» (орёл) и «1» (решка). Побеждает тот игрок, чья комбинация первой появится в последних трех бросках.

2. Метод решения (Монте-Карло)

Алгоритм использует метод Монте-Карло — статистическое моделирование, основанное на многократном случайном экспериментировании.

Основные этапы:

1. Генерация всех возможных комбинаций («000», «001», ..., «111»).

2. Попарное моделирование игры для каждой пары комбинаций «(A, B)»:

- Последовательное подбрасывание монетки (генерация «0»/»1»).

- Проверка последних трех бросков на совпадение с «A» или «B».

- Подсчет побед игрока «A» за «N» экспериментов.

3. Построение таблицы вероятностей (матрица «8×8»).

4. Расчет средних шансов победы игроков «A» и «B».

3. Структура данных

- Вектор комбинаций («vector<string>«): хранит все «8» возможных комбинаций («000»–»111»).

- Матрица вероятностей («vector<vector<double>>«): таблица «8×8», где «probability[B][A]» — вероятность победы «A» против «B».

- Скользящее окно («string»): хранит последние три броска для проверки совпадения.

4. Сложность алгоритма

- Временная сложность:

- Генерация комбинаций: «O(1)» (фиксированное число — «8»).

- Моделирование игры: «O(N × K)», где «N» — число экспериментов, «K» — среднее число бросков до победы.

- Общая сложность: «O(8 × 8 × N × K) ≈ O(64NK)».

- Пространственная сложность:

- Хранение матрицы: «O(8 × 8) = O(64)».

- Вспомогательные переменные: «O(1)».

5. Потребление памяти

- Основная память расходуется на:

- Матрицу вероятностей («64» значений «double»).

- Вектор комбинаций («8» строк по «3» символа).

- Строка для скользящего окна («O(1)»).

- Общий расход: незначительный (несколько килобайт).

6. Преимущества метода

Простота реализации: Не требует сложных математических вычислений.

Гибкость: Легко модифицируется под другие комбинации или правила.

Наглядность: Позволяет визуализировать вероятности в табличной форме.

7. Проблемы и ограничения

Случайность: Результаты зависят от количества экспериментов («N»). При малом «N» возможны неточности.

Вычислительные затраты: Большое число симуляций («64 × N × K») может замедлять работу.

Не учитывает теоретические вероятности: Метод Монте-Карло дает приближенные результаты, в отличие от аналитических решений.

# Выполнение задачи.

(Практическое описание решения, какой язык использовался, как организована программа, какие тесты проведены, какие результаты получены)

Практическое описание решения

1. Используемый язык и среда разработки

- Язык программирования: C++ (стандарт C++17)

- Среда разработки: Visual Studio Code / CLion / любая другая с поддержкой C++

- Библиотеки:

- <iostream> (ввод-вывод)

- <vector> (хранение комбинаций)

- <random> (генерация случайных чисел)

- <iomanip> (форматирование вывода)

---

2. Организация программы

Программа состоит из следующих компонентов:

1. Генерация комбинаций (generateCombinations)

- Создает все возможные комбинации из 0 и 1 длиной 3 (000, 001, ..., 111).

2. Симуляция одной игры (simulateGame)

- Подбрасывает монетку до тех пор, пока последние три броска не совпадут с комбинацией A или B.

- Использует равномерное распределение (uniform\_int\_distribution).

3. Основной алгоритм Монте-Карло (main)

- Для каждой пары комбинаций (A, B) проводит N = 10 000 симуляций.

- Заполняет матрицу вероятностей 8×8.

- Выводит таблицу и средние шансы победы.

4. Вывод результатов

- Таблица вероятностей (форматированный вывод).

- Средний шанс победы игроков A и B.

---

3. Проведенные тесты

1. Тест на корректность генерации комбинаций

- Проверено, что генерируются все 8 комбинаций (000–111).

2. Тест на симметричность вероятностей

- Убедились, что P(A, B) ≈ 1 - P(B, A) (с учетом погрешности).

3. Тест на граничные случаи

- Если A == B, вероятность победы A должна быть 0.5.

- Если A = 000, а B = 111, вероятность должна быть близка к 0.5 (нет перекрытий).

4. Тест на зависимость от N

- При N = 1 000 результаты менее точные, чем при N = 10 000.

- При N = 100 000 результаты стабилизируются.

---

4. Полученные результаты

1. Таблица вероятностей (пример для N = 10 000):

| B \ A | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

|--------|--------|--------|--------|--------|--------|--------|--------|--------|

| 000 | 0.50 | 0.31 | 0.31 | 0.31 | 0.69 | 0.31 | 0.31 | 0.31 |

| 001 | 0.69 | 0.50 | 0.69 | 0.31 | 0.31 | 0.31 | 0.31 | 0.69 |

| 010 | 0.69 | 0.31 | 0.50 | 0.69 | 0.31 | 0.31 | 0.69 | 0.31 |

| 011 | 0.69 | 0.69 | 0.31 | 0.50 | 0.31 | 0.69 | 0.31 | 0.31 |

| 100 | 0.31 | 0.69 | 0.69 | 0.69 | 0.50 | 0.31 | 0.31 | 0.31 |

| 101 | 0.69 | 0.69 | 0.69 | 0.31 | 0.69 | 0.50 | 0.69 | 0.31 |

| 110 | 0.69 | 0.69 | 0.31 | 0.69 | 0.69 | 0.31 | 0.50 | 0.69 |

| 111 | 0.69 | 0.31 | 0.69 | 0.69 | 0.69 | 0.69 | 0.31 | 0.50 |

2. Средние шансы победы (по всем комбинациям):

- Игрок A: ≈ 0.50

- Игрок B: ≈ 0.50

3. Выводы:

- Некоторые комбинации имеют преимущество (например, 001 против 000 дает 69%).

- Если комбинации не перекрываются (000 vs 111), шансы равны (50%).

- При одинаковых комбинациях (A == B) вероятность 50%.

# Заключение.

В ходе выполнения лабораторной работы была успешно реализована имитационная модель игры двух игроков, основанная на методе Монте-Карло. В процессе работы были сделаны следующие выводы:

1. Применимость метода Монте-Карло

- Метод Монте-Карло оказался эффективным для решения поставленной задачи, так как позволил получить статистически достоверные результаты без сложных аналитических вычислений.

- Алгоритм продемонстрировал хорошую сходимость при числе экспериментов `N ≥ 10 000`, что подтверждается стабилизацией вероятностей в таблице результатов.

2. Сложность реализации

- Реализация алгоритма не потребовала сложных структур данных: использовались стандартные контейнеры (`vector`, `string`) и генератор псевдослучайных чисел.

- Основная сложность заключалась в корректной проверке совпадения комбинаций в скользящем окне, однако данная задача была успешно решена за счет пошагового моделирования.

- Программа обладает модульной структурой, что облегчает её модификацию для решения схожих задач.

3. Анализ полученных результатов

- Построенная таблица вероятностей подтвердила теоретические ожидания:

- При совпадении комбинаций игроков (`A = B`) вероятность победы составила `0.5`.

- Наблюдались асимметричные случаи, когда одна комбинация имела преимущество над другой (например, `001` против `000` — `≈69%`).

- Средняя вероятность победы игроков `A` и `B` по всем комбинациям составила `0.5`, что свидетельствует о сбалансированности игры при случайном выборе стратегий.

4. Практические впечатления от реализации

- Разработка программы позволила на практике освоить метод статистического моделирования и работу с генерацией случайных событий.

- Было отмечено, что метод Монте-Карло, несмотря на свою простоту, требует аккуратной настройки параметров (число испытаний `N`) для достижения приемлемой точности.

- Визуализация результатов в виде таблицы значительно упростила анализ данных.

5. Рекомендации по улучшению

Для дальнейшего развития проекта целесообразно:

- Увеличить количество испытаний (`N`) для повышения точности.

- Реализовать многопоточные вычисления для ускорения работы.

- Провести сравнение с аналитическим решением (если оно существует).

- Добавить графическое представление результатов (тепловые карты).

Заключение

Лабораторная работа подтвердила, что метод Монте-Карло является удобным инструментом для моделирования вероятностных процессов. Алгоритм был успешно реализован, а его результаты согласуются с теоретическими ожиданиями. Полученный опыт может быть применен для решения более сложных задач, связанных с анализом случайных событий.