Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №3

“ Побудова оберненої матриці методом мінорів”

Варіант №4

Виконав студент 2-го курсу

Групи 3

Гончаренко Ілля Сергійович

Київ - 2023

Завдання:

Реалізація побудови оберненої матриці в стилі vector<vector<T>> методом мінорів для дійснозначних чисел.

Предметна область: Комп'ютерна графіка. Графічний додаток.

Примітки: У графічних додатках обернені матриці використовуються для трансформацій об'єктів, таких як обертання, масштабування чи зміщення.

Теорія

Метод мінорів — це один із способів знаходження оберненої матриці. Цей метод базується на визначенні мінорів та алгебраїчних доповнень.

Основна ідея полягає в тому, що кожен елемент оберненої матриці обчислюється за допомогою відповідного алгебраїчного доповнення та ділення на детермінант вихідної матриці. Цей метод дозволяє знаходити обернену матрицю за допомогою визначників та алгебраїчних доповнень, але він може бути витратним з точки зору обчислювальних ресурсів, особливо для великих матриць.

‘vector’ - це динамічний масив, який може змінювати свій розмір під час виконання програми. Вектор у векторі (vector of vectors) можна використовувати для представлення двовимірних структур даних, таких як матриці, де кожен зовнішній вектор представляє рядок матриці, а внутрішні вектори містять її елементи.

Дійснозначна матриця - це матриця, всі елементи якої є дійсними числами. Дійсні числа включають у себе усі раціональні та ірраціональні числа, які не мають уявної частини. Отже, дійснозначна матриця не містить комплексних чисел.

Детермінант матриці - це числова характеристика квадратної матриці, яка визначається за певним способом. Для квадратної матриці A розмірності n×n, детермінант позначається як det(A) або |A|.

Алгебраїчне доповнення (або алгебраїчний додаток) елемента матриці - це число, яке отримується множенням відповідного мінора на відповідний множник, залежний від позначення рядка та стовпця елемента.

Алгоритм

1. Обчислити визначник $|A|$ оригінальної матриці A.
2. Для кожного елемента A_{ij} оригінальної матриці A, обчислити його алгебраїчне доповнення C_{ij} (мінор матриці, яка отримується видаленням рядка i та стовпчика j).
3. Знайти кожен елемент оберненої матриці A_{ij}^{-1} за допомогою виразу:
   $A_{ij}^{-1} = \frac{1}{|A|} \cdot (-1)^{i+j} \cdot |C_{ij}|$
4. Сформувати нову матрицю, в якій кожен елемент A_{ij}^{-1} визначений згідно попереднього кроку.

Важливо врахувати, що визначник матриці |A| повинен бути відмінним від нуля, оскільки для нульового визначника оберненої матриці не існує. Також цей метод не є найефективнішим для великих матриць, і в практичних застосуваннях часто використовують більш оптимізовані алгоритми.

Цей код описує вище перечислений алгоритм:

Function знаходження_мінора(matrix, row, col):

//Створити нову матрицю minor розміром (n-1) x (n-1)

//Заповнити minor елементами з matrix, виключаючи рядок row та

//заповнити col

return minor;

Function обернена_матриця(matrix):

if matrix не є квадратною матрицею:

print: "Матриця не є квадратною"

break;

//Знайти детермінант матриці det_matrix

if det_matrix = 0:

print: "Детермінант матриці дорівнює нулю, обернена матриця не існує"

break;

//Створити нову порожню матрицю inverse_matrix того ж розміру, що й matrix

//Для кожного елемента matrix з індексами i та j:

//Знайти мінор для елемента matrix[i][j]

//Обчислити доповнений мінор minor_ij як детермінант мінора, помножений на (-1)^(i + j)

inverse_matrix[j][i] = minor_ij / det_matrix

return inverse_matrix

Складність алгоритму

Складність побудови оберненої матриці методом мінорів є високою і зазвичай дорівнює приблизно O(n!), де n - розмірність квадратної матриці. Така висока складність обумовлена необхідністю обчислення визначника та матриці алгебраїчних доповнень, що займає

час та ресурси, особливо для великих матриць.

Для матриці розміром n x n загальна кількість операцій буде порядку O(n^4), оскільки для кожного з n^2 елементів оберненої матриці потрібно знайти визначник підматриці, що вже потребує обчислення визначника, який вже сам по собі може займати O(n^3) операцій. Таким чином, цей метод не є ефективним для великих матриць через велику обчислювальну складність.

Мова реалізації алгоритму C++

Модулі програми:

- **std::vector<std::vector<double>>matrixAddition(const std::vector<std::vector<double>>&A,const std::vector<std::vector<double>>& B)**

Ця функція виконує операцію додавання двох матриць. Вхідними параметрами є дві матриці A і B, обидві представлені у вигляді вектора векторів подвійного типу даних (std::vector<std::vector<double>>).

Функція спочатку визначає розмір матриці A (або B, оскільки вони повинні мати однаковий розмір для виконання операції додавання), і створює новий вектор векторів C з таким самим розміром.

Потім вона використовує вкладені цикли для ітерації по кожному елементу матриць A і B та виконує додавання їх відповідних елементів. Результат зберігається у відповідному елементі матриці C.

На завершення функція повертає отриману матрицю C, що представлена як вектор векторів. Таким чином, якщо ви викличете цю функцію із двома матрицями A і B, вона поверне нову матрицю C, яка є результатом їх додавання.

- **std::vector<std::vector<double>>matrixSubtraction(const std::vector<std::vector<double>>&A,const std::vector<std::vector<double>>& B)**

Ця функція виконує операцію віднімання двох матриць. Вхідними параметрами є дві матриці A і B, обидві представлені у вигляді вектора векторів подвійного типу даних (std::vector<std::vector<double>>).

Функція спочатку визначає розмір матриці A (або B, оскільки вони повинні мати однаковий розмір для виконання операції віднімання), і створює новий вектор векторів C з таким самим розміром.

Потім вона використовує вкладені цикли для ітерації по кожному елементу матриць A і B та виконує віднімання відповідних елементів. Результат зберігається у відповідному елементі матриці C.

На завершення функція повертає отриману матрицю C, що представлена як вектор векторів. Таким чином, якщо ви викличете цю функцію із двома матрицями A і B, вона поверне нову матрицю C, яка є результатом їх віднімання.

- **std::vector<std::vector<double>> strassenMultiplication(const std::vector<std::vector<double>>&A,const std::vector<std::vector<double>>& B)**

Функція реалізує алгоритм множення матриць з використанням алгоритму Штрассена, який є спеціальним алгоритмом множення матриць і призначений для використання при обчисленнях над великими матрицями. Алгоритм Штрассена спрощує процес множення,

розбиваючи великі матриці на менші підматриці і обчислюючи їх добутки рекурсивно.

Основні кроки функції:

1) Якщо розмір вхідних матриць менший або рівний 2, то використовується звичайний алгоритм множення матриць для невеликих матриць.
2) Якщо розмір матриці більший за 2, то матриця A і B розбивається на чотири підматриці: A11, A12, A21, A22 та B11, B12, B21, B22.
3) Обчислюються сім допоміжних матриць P1 до P7 за допомогою рекурсивних викликів функції strassenMultiplication. Ці матриці обчислюються за допомогою операцій додавання, віднімання та множення.
4) Обчислюються результатні підматриці C11, C12, C21, C22 за допомогою операцій додавання та віднімання, використовуючи значення P1 до P7.
5) Складається результатна матриця C з отриманих підматриць C11, C12, C21, C22.

Функція повертає отриману матрицю C, що є результатом множення матриць A і B за алгоритмом Штрассена.

- **T determinant(std::vector<std::vector<T>>& matrix)**

  Ця функція обчислює визначник квадратної матриці. Вхідним параметром є матриця matrix типу T (загальний тип даних, оскільки використовується шаблон).

  Основні кроки функції:

1) Перевіряється, чи є матриця квадратною (кількість рядків дорівнює кількості стовпців) і чи не є порожньою. Якщо матриця не є

квадратною або порожньою, повертається 0, оскільки визначник не визначений для таких матриць.

2) Якщо розмір матриці дорівнює 1, повертається єдиний елемент матриці як визначник.
3) Якщо розмір матриці більший за 1, визначник обчислюється рекурсивно. Для кожного стовпця першого рядка матриці обчислюється визначник за допомогою відповідної підматриці, яка утворюється вилученням рядка і стовпця, де перший елемент розглядується. Коефіцієнт sign змінюється між додаванням і відніманням для кожного ітераційного кроку.
4) Сумуються отримані визначники за всі стовпці першого рядка з врахуванням знаку, і отриманий результат повертається як визначник вхідної матриці.

Ця реалізація використовує рекурсивний підхід та визначає визначник матриці за допомогою рекурсивного розкладання по першому рядку.

- **std::vector<std::vector<T>> inverseMatrix(std::vector<std::vector<T>>& matrix)**

Ця функція обчислює обернену матрицю для вхідної квадратної матриці matrix. Вихідною є нова матриця, яка є оберненою до вхідної матриці.

Основні кроки функції:

1) Визначається розмір матриці n.
2) Створюється нова матриця invMatrix такого ж розміру, яка буде містити обернену матрицю.
3) Обчислюється визначник вхідної матриці за допомогою функції determinant.
4) Якщо визначник рівний нулю, функція виводить повідомлення про те, що обернена матриця не існує, і повертає порожню матрицю.

5) Використовуючи рекурсивний підхід, для кожного елемента оберненої матриці обчислюється відповідна підматриця, і потім визначається елемент оберненої матриці за формулою:

$$\text{invMatrix}[j][i] = (-1)^{i+j} \cdot \frac{\text{determinant(submatrix)}}{\det}$$

де i та j - індекси елемента, а `submatrix` - підматриця, отримана виключенням рядка і стовпця, де розташований поточний елемент.

6) Отримана обернена матриця повертається як результат функції.

- **void printMatrix(std::vector<std::vector<T>>& matrix)**

Ця функція виводить матрицю на екран (консоль). Вхідним параметром є матриця типу T (загальний тип даних, оскільки використовується шаблон).

Основний алгоритм функції:

1) Функція використовує цикл для ітерації по кожному рядку матриці.
2) У внутрішньому циклі для кожного елемента рядка виводиться значення елемента на екран, відокремлене табуляцією.
3) Після виведення всіх елементів рядка вставляється рядок нового рядка для подальшого виведення наступного рядка матриці.

Ця функція допомагає візуалізувати матрицю при виведенні у консоль, щоб користувач міг переглядати її елементи в зручному форматі.

Інтерфейс користувача

Цей код представляє собою консольний інтерфейс, що використовує різні функції для роботи з матрицями. Основні операції, які виконуються у цьому інтерфейсі:

1. Ініціалізація матриць A і B з числовими значеннями для демонстрації роботи алгоритмів.
2. Множення матриць A і B за допомогою алгоритму Штрассена.
3. Виведення на екран початкових матриць A і B, а також результату їхнього множення методом Штрассена.
4. Обчислення оберненої матриці від результату множення (C) за допомогою методу мінорів.
5. Виведення на екран оберненої матриці.

Це дозволяє користувачеві побачити початкові матриці, результат множення методом Штрассена та обернену матрицю отриману від цього результату. Користувач може вводити свої матриці або використовувати дані, які вже запрограмовані.

Тестові приклади

1. Тест для двох 2x2 матриць:

   A = {{1, sqrt(2)}, B = {{7, 1},

   {-2, 6.5}} {1, 2}}

   Очікуваний результат (C):

   {{0.090707, -0.0315696},

{0.0618457 0.0693844}}

2. Тест для двох 1x1 матриць:

   A = {{3}} B = {{4}}

   Очікуваний результат (C):

   {{0}}

3. Тест для матриці розміру 2x2 та матриці розміру 4x4:

   A = {{1, 2}, B = {{1, 2, 3, 4},

   {3, 4}} {5, 6, 7, 8}}

   Очікуваний результат: Помилка - матриці несумісні

4. Тест для двох пустих матриць:

   A = {} B = {}

   Очікуваний результат: Помилка - матриці порожні

Функція знаходження рберненої матриці методом мінорів працює коректно для випадку множення двох 2x2 матриць, як показано в першому тесті. Очікуваний результат співпадає з отриманим результатом. Функція також коректно обробляє множення двох 1x1 матриць. Очікуваний результат співпадає з отриманим результатом. Функція виявляє несумісність матриць для множення і повертає відповідне повідомлення про помилку, як у випадку множення 2x2 матриці на 4x4 матрицю. Функція також правильно обробляє випадок, коли обидві вхідні матриці порожні. Очікуваний результат співпадає з повідомленням про помилку.

Висновки

Отже, у даній роботі, було описано алгоритм знаходження оберненої матриці методом мінорів для для дійснозначної матриці в стилі vector<vector<T>>.

Цей код представляє собою реалізацію алгоритму множення матриць методом Штрассена, та знаходження оберненої матриці методом мінорів а також функцій для додавання та віднімання матриць. Та він працює для валідних вхідних даних.

Використані літературні джерела

- https://xreferat.com/54/2702-1-znahodzhennya-oberneno-matric-za-formuloyu.html
- https://www.youtube.com/watch?v=Y_WGlrCsmpU
- https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D1%80%D0%BD%D0%B5%D0%BD%D0%B0_%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8F
- https://ru.frwiki.wiki/wiki/Algorithme_de_Strassen