Nguyễn Ngọc Trà My – 47.01.103.068

**Bài thực hành**

Bài thực hành 2: Cài đặt chứng minh độ phức tạp, giải thuật Strassen cho nhân ma trận

**Input:**

Hai ma trận A và B cùng kích thước nxn

**Output:**

Ma trận kết quả C là tích của ma trận A và B

**Cách xử lý**

* Chia ma trận A và B thành các ma trận con theo công thức của Strassen
* Chia A thành A11, A12, A21, A22
* Chia B thành B11, B12, B21, B22
* Tính các ma trận con T1, T2, T3, T4, T5, T6, T7 theo công thức của Strassen.

T1 = (A11 + A22) \* (B11 + B22)

T2 = (A21 + A22) \* B11

T3 = A11 \* (B12 - B22)

T4 = A22 \* (B21 - B11)

T5 = (A11 + A12) \* B22

T6 = (A21 - A11) \* (B11 + B12)

T7 = (A12 - A22) \* (B21 + B22)

* Tính các ma trận con C11, C12, C21, C22 theo công thức của Strassen.

C11 = T1 + T4 - T5 + T7

C12 = T3 + T5

C21 = T2 + T4

C22 = T1 + T3 - T2 + T6

* Tạo ma trận kết quả C bằng cách ghép các ma trận con theo công thức của Strassen.
* Trả về ma trận kết quả C.

**Code**

#include <iostream>

#include <vector>

using namespace std;

// Hàm cộng hai ma trận

vector<vector<int>> matrixAddition(const vector<vector<int>>& A, const vector<vector<int>>& B) {

int n = A.size();

vector<vector<int>> result(n, vector<int>(n, 0));

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

result[i][j] = A[i][j] + B[i][j];

}

}

return result;

}

// Hàm trừ hai ma trận

vector<vector<int>> matrixSubtraction(const vector<vector<int>>& A, const vector<vector<int>>& B) {

int n = A.size();

vector<vector<int>> result(n, vector<int>(n, 0));

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

result[i][j] = A[i][j] - B[i][j];

}

}

return result;

}

// Hàm nhân ma trận theo giải thuật Strassen

vector<vector<int>> strassenMultiplication(const vector<vector<int>>& A, const vector<vector<int>>& B) {

int n = A.size();

// Trường hợp ma trận 1x1

if (n == 1) {

return {{A[0][0] \* B[0][0]}};

}

// Chia ma trận A và B thành các ma trận con

int halfSize = n / 2;

vector<vector<int>> A11(A.begin(), A.begin() + halfSize);

vector<vector<int>> A12(A.begin(), A.begin() + halfSize);

vector<vector<int>> A21(A.begin() + halfSize, A.end());

vector<vector<int>> A22(A.begin() + halfSize, A.end());

vector<vector<int>> B11(B.begin(), B.begin() + halfSize);

vector<vector<int>> B12(B.begin(), B.begin() + halfSize);

vector<vector<int>> B21(B.begin() + halfSize, B.end());

vector<vector<int>> B22(B.begin() + halfSize, B.end());

// Tính các ma trận con theo công thức Strassen

vector<vector<int>> T1 = strassenMultiplication(matrixAddition(A11, A22), matrixAddition(B11, B22));

vector<vector<int>> T2 = strassenMultiplication(matrixAddition(A21, A22), B11);

vector<vector<int>> T3 = strassenMultiplication(A11, matrixSubtraction(B12, B22));

vector<vector<int>> T4 = strassenMultiplication(A22, matrixSubtraction(B21, B11));

vector<vector<int>> T5 = strassenMultiplication(matrixAddition(A11, A12), B22);

vector<vector<int>> T6 = strassenMultiplication(matrixSubtraction(A21, A11), matrixAddition(B11, B12));

vector<vector<int>> T7 = strassenMultiplication(matrixSubtraction(A12, A22), matrixAddition(B21, B22));

// Tính các ma trận kết quả theo công thức Strassen

vector<vector<int>> C11 = matrixAddition(matrixSubtraction(matrixAddition(T1, T4), T5), T7);

vector<vector<int>> C12 = matrixAddition(T3, T5);

vector<vector<int>> C21 = matrixAddition(T2, T4);

vector<vector<int>> C22 = matrixAddition(matrixSubtraction(matrixAddition(T1, T3), T2), T6);

// Tạo ma trận kết quả C bằng cách ghép các ma trận con

vector<vector<int>> C(n, vector<int>(n, 0));

for (int i = 0; i < halfSize; i++) {

for (int j = 0; j < halfSize; j++) {

C[i][j] = C11[i][j];

C[i][j + halfSize] = C12[i][j];

C[i + halfSize][j] = C21[i][j];

C[i + halfSize][j + halfSize] = C22[i][j];

}

}

return C;

}

int main() {

vector<vector<int>> A = {{1, 2}, {3, 4}}; // Ma trận A

vector<vector<int>> B = {{5, 6}, {7, 8}}; // Ma trận B

vector<vector<int>> C = strassenMultiplication(A, B); // Nhân ma trận A và B

cout << "Ma tran C (A x B):" << endl;

for (const auto& row : C) {

for (int element : row) {

cout << element << " ";

}

cout << endl;

}

return 0;

}

**Kết quả**

A picture containing text, screenshot, display, font

Description automatically generated