Algorithm C, Strassen’s Multiplication pseudocode.

A diagram of a multiplicatoin

Description automatically generated

Let ClasssicMultiplication // Method a)

From DevideConquerMultiplication

* AddMtx
* SplitMtx
* isNbyN
* AggregateMtx
* IsPowerOfTwo

//Input: two matrices

//Process: Subtract two matrix

//Output: result of matrix

subtractMtx

1. Start
2. Accept two matrices, mtxA and mtxB
3. Check mtxA and mtxB has the same number of row // Call isMatchNumOfRow

If not, return null

1. Check mtxA and mtxB has the same number of column //Call isMatchNumOfClm

If not, return null

1. Create a new matrix, mtxC
2. Set loop which iterates form the first row to the last row of mtxA and mtxB
   1. Set loop which iterates from the first column to the last columns to mtxA and mtxB
      1. Subtract value in the corresponding row and column of mtxA and mtxC

Eg. mtxC[0][0] = mtxA[0][0] - mtxB[0][0]

* + 1. Store the result to mtxC corresponding row and column

1. Return mtxC
2. Stop

//Input: 2 four matrices array lists

//Process: Calculate all the matrices for C11 and C22

//Output: result of matrix

calcC11\_C22

1. Start
2. Create a new matrix subMtxC
3. Calculate as follow
   1. subMtxC += mtx1 // Now subMtxC == mtx1
   2. subMtxC += mtx2
   3. subMtxC -= mtx3
   4. subMtxC += mtx4
4. Return subMtxC
5. End

//Input: 2 matrices

//Process: multiple two matrix with the Strassen formula recursively

//Output: 1 matrix

StrassenMultiplicatoin

1. Start
2. Accept 2 matrices, mtxA and mtxB
3. If the size of mtxA and size of mtxB are 2 // Base case

Calculate 2 by 2 matrix // Call classicMultiplicatoin

Return the new matrix C, mtx\_c

else

1. Split mtxA into sub matrix as a11, a12, a21, and a22 // Call SplitMtx
2. Split mtxB into submatrix as b11, b12, b21, and b22 // Call SplitMtx

//Make P

1. Create sub matrix mtxPL
2. Create sub matrix mtxPR
3. Create matrix mtxP // Recursive call strassenMultiplicatoin(mtxPL, mtxPR)

//Make Q

1. Create sub matrix mtxQL
2. Create matrix mtxP // Recursive call strassenMultiplicatoin(mtxQL, b11)

//Make R

1. Create sub matrix mtxRR
2. Create matrix mtxR // Recursive call strassenMultiplicatoin(a11, mtxRR)

//Make S

1. Create sub matrix mtxSR
2. Create matrix mtxS // Recursive call strassenMultiplicatoin(a22, mtxSR)

//Make T

1. Create sub matrix mtxTL
2. Create matrix mtxT // Recursive call strassenMultiplicatoin(mtxTL, b22)

//Make U

1. Create sub matrix mtxUL
2. Create sub matrix mtxUR
3. Create matrix mtxU // Recursive call strassenMultiplicatoin(mtxUL, mtxUR)

//Make V

1. Create sub matrix mtxVL
2. Create sub matrix mtxVR
3. Create matrix mtxV // Recursive call strassenMultiplicatoin(mtxVL, mtxVR)

//Make C11

1. Calculate mtxP + mtxS – mtxT + mtxV as c11

// Call CalcC11\_C22(mtxP, mtxS, mtxT, mtxV)

//Make C12

1. Calcluate R+T as c12// Call AddMtx(mtxR, mtxT)

//Make C21

1. Calculate Q + S as c21// Call AddMtx(mtxQ, mtxS)

//Make C22

1. Calculate mtxP + mtxR – mtxQ + mtxU as c22

// Call CalcC11\_C22(mtxP, mtxR, mtxQ, mtxU)

//Matrix C

1. Aggregate sub matrix c11, c12, c21 and c22 //Call AggregateMtx(c11, c12, c21,c22)
2. Return mtxC
3. Stop