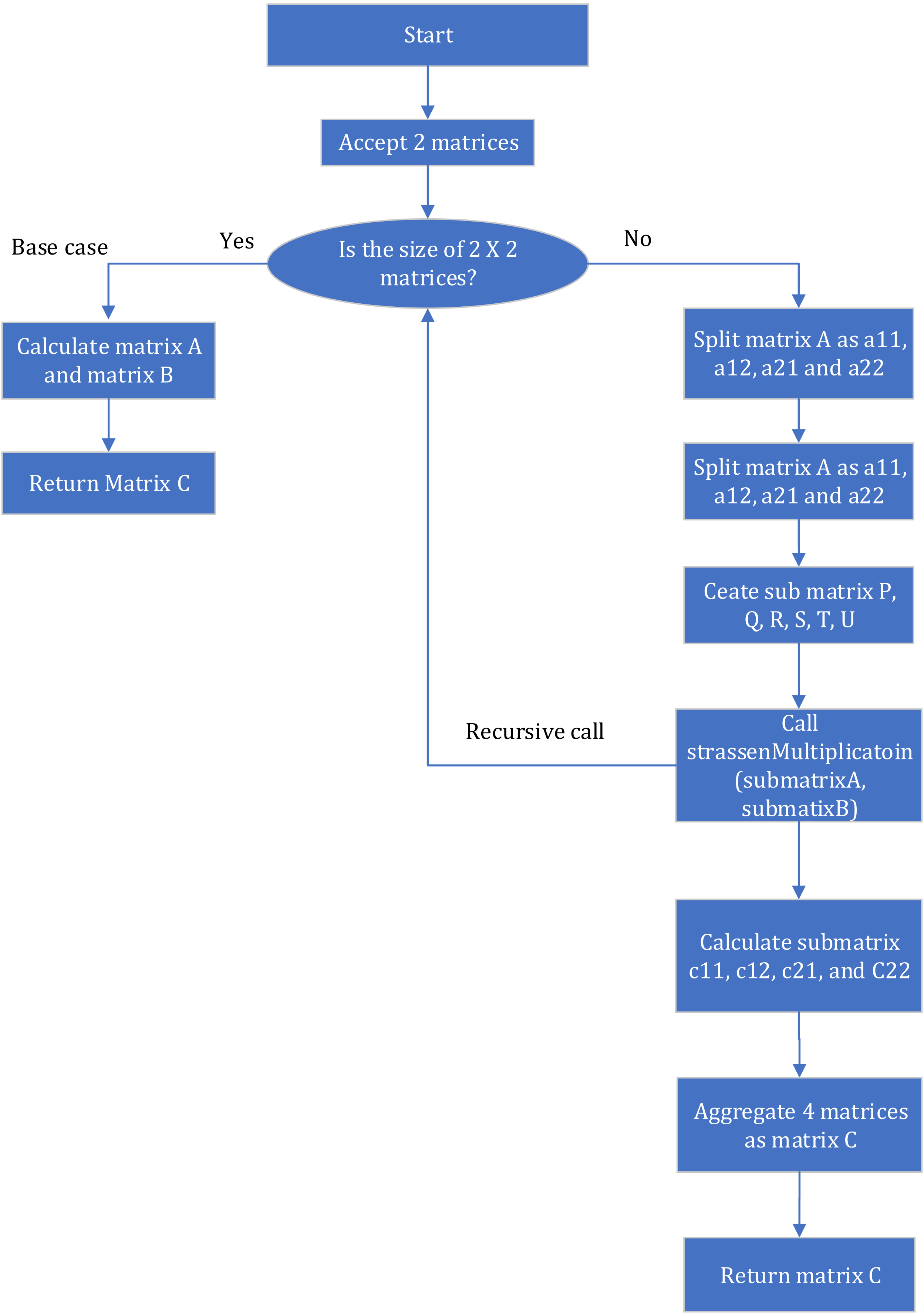
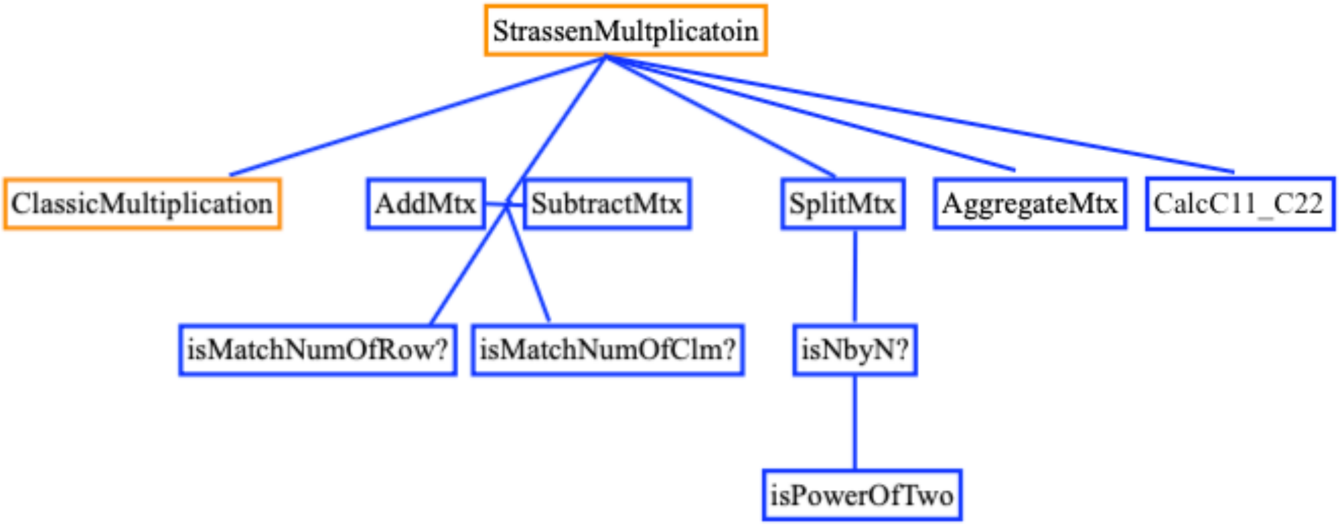

Start
Accept 2 matrices
Base case
Yes
Is the size of 2 X 2 matrices?
No
Calculate matrix A and matrix B
Return Matrix C
Split matrix A as a11, a12, a21 and a22
Split matrix A as a11, a12, a21 and a22
Ceate sub matrix P, Q, R, S, T, U
Recursive call
Call strassenMultiplicatoin (submatrixA, submatixB)
Calculate submatrix c11, c12, c21, and C22
Aggregate 4 matrices as matrix C
Return matrix C

Algorithm C, Strassen's Multiplication pseudocode.



Let ClasssicMultiplication // Method a)

From DevideConquerMultiplication

- AddMtx
- SplitMtx
- isNbyN
- AggregateMtx
- IsPowerOfTwo

```
//Input: two matrices
//Process: Subtract two matrix
//Output: result of matrix
subtractMtx
0. Start
1. Accept two matrices, mtxA and mtxB
2. Check mtxA and mtxB has the same number of row // Call isMatchNumOfRow
        If not, return null
3. Check mtxA and mtxB has the same number of column //Call isMatchNumOfClm
        If not, return null
4. Create a new matrix, mtxC
5. Set loop which iterates form the first row to the last row of mtxA and mtxB
    a. Set loop which iterates from the first column to the last columns to mtxA and
       mtxB
        i. Subtract value in the corresponding row and column of mtxA and mtxC
```

```
                Eg. mtxC[0][0] = mtxA[0][0] - mtxB[0][0]
            ii. Store the result to mtxC corresponding row and column
6. Return mtxC
7. Stop

//Input: 2 four matrices array lists
//Process: Calculate all the matrices for C11 and C22
//Output: result of matrix
calcC11_C22
0. Start
1. Create a new matrix subMtxC
2. Calculate as follow
    a. subMtxC += mtx1 // Now subMtxC == mtx1
    b. subMtxC += mtx2
    c. subMtxC -= mtx3
    d. subMtxC += mtx4
3. Return subMtxC
4. End
```

```
//Input: 2 matrices
//Process: multiple two matrix with the Strassen formula recursively
//Output: 1 matrix
StrassenMultiplicatoin
0.  Start
1.  Accept 2 matrices, mtxA and mtxB
2.  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
3.  Split mtxA into sub matrix as a11, a12, a21, and a22 // Call SplitMtx
4.  Split mtxB into submatrix as b11, b12, b21, and b22 // Call SplitMtx

//Make P
5.  Create sub matrix mtxPL
6.  Create sub matrix mtxPR
7.  Create matrix mtxP // Recursive call strassenMultiplicatoin(mtxPL, mtxPR)

//Make Q
8.  Create sub matrix mtxQL
9.  Create matrix mtxP // Recursive call strassenMultiplicatoin(mtxQL, b11)

//Make R
10. Create sub matrix mtxRR
11. Create matrix mtxR // Recursive call strassenMultiplicatoin(a11, mtxRR)


//Make S
12. Create sub matrix mtxSR
13. Create matrix mtxS // Recursive call strassenMultiplicatoin(a22, mtxSR)

//Make T
14. Create sub matrix mtxTL
15. Create matrix mtxT // Recursive call strassenMultiplicatoin(mtxTL, b22)

//Make U
16. Create sub matrix mtxUL
17. Create sub matrix mtxUR
18. Create matrix mtxU // Recursive call strassenMultiplicatoin(mtxUL, mtxUR)

//Make V
```

```
19.Create sub matrix mtxVL
20.Create sub matrix mtxVR
21.Create matrix mtxV // Recursive call strassenMultiplicatoin(mtxVL, mtxVR)

//Make C11
22.Calculate mtxP + mtxS – mtxT + mtxV as c11
        // Call CalcC11_C22(mtxP, mtxS, mtxT, mtxV)

//Make C12
23.Calcluate R+T as c12// Call AddMtx(mtxR, mtxT)

//Make C21
24.Calculate Q + S as c21// Call AddMtx(mtxQ, mtxS)

//Make C22
25.Calculate mtxP + mtxR – mtxQ + mtxU as c22
        // Call CalcC11_C22(mtxP, mtxR, mtxQ, mtxU)

//Matrix C
26.Aggregate sub matrix c11,  c12, c21 and c22 //Call AggregateMtx(c11, c12, c21,c22)
27.Return mtxC
28.Stop
```