Divide and Conquer
Stressen's Matrix Multiplication

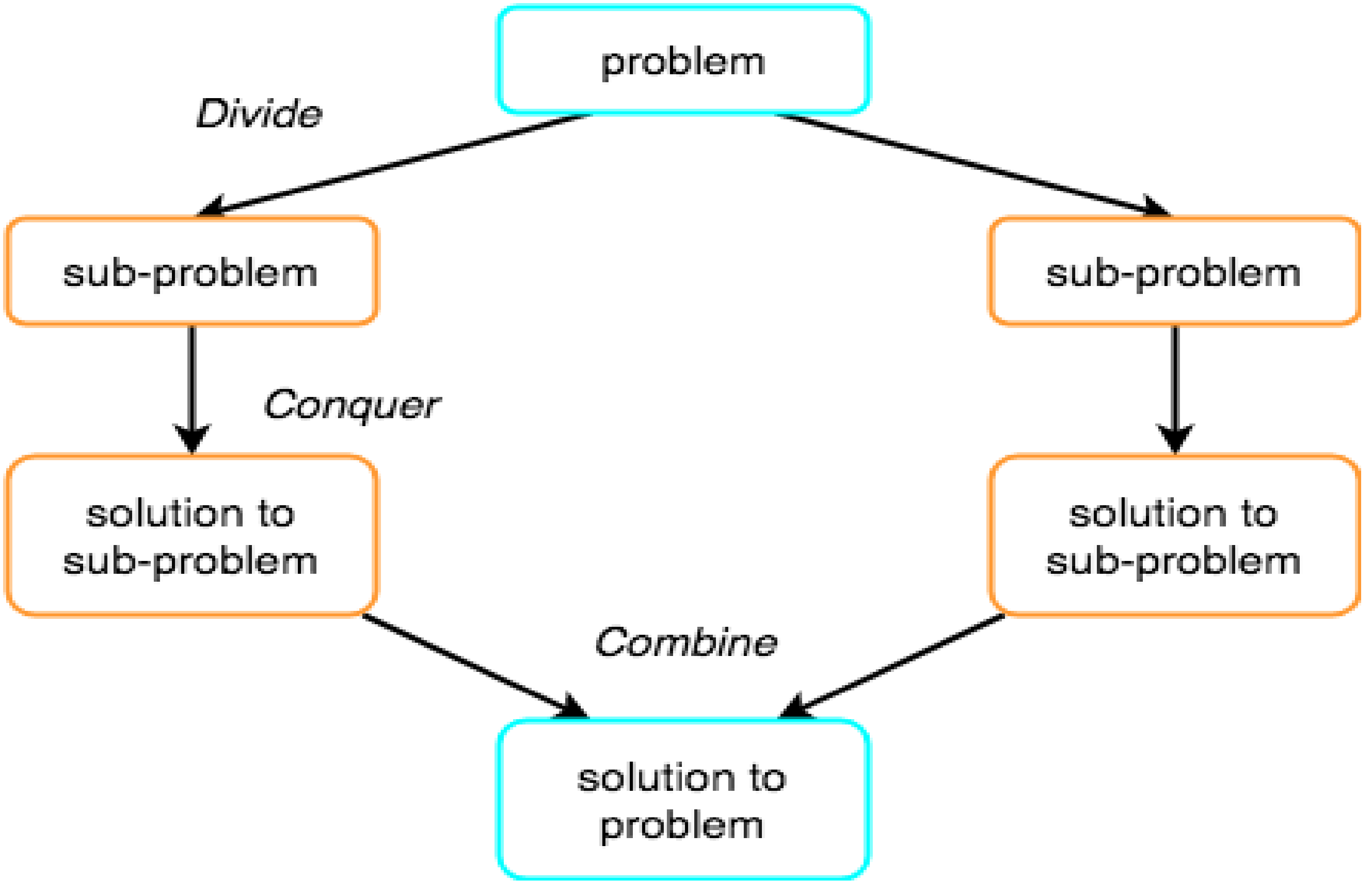
Divide-and-conquer approach

- Divide-and-conquer approach: they break the problem into several subproblems that are similar to the original problem but smaller in size, solve the subproblems recursively, and then combine these solutions to create a solution to the original problem

Steps of divide-and-conquer paradigm

- It involves three steps:
  - **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
  - **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in straightforward manner.
  - **Combine** the solutions to the subproblems into the solution for the original problem.

Divide-and-conquer



Control Abstraction of Divide and Conquer

- The control abstraction for divide and conquer technique is DANDC(P), where P is the problem to be solved.

```
Algorithm DAndC(P)
{
    if Small(P) then return S(P);
    else
    {
        divide P into smaller instances P_1, P_2, ..., P_k, k ≥ 1;
        Apply DAndC to each of these subproblems;
        return Combine(DAndC(P_1),DAndC(P_2),...,DAndC(P_k));
    }
}
```

- SMALL (P) is a Boolean valued function which determines whether the input size is small enough so that the answer can be computed without splitting.
- If this is so function ‘S’ is invoked otherwise, the problem ‘p’ divided into smaller sub problems.
- These sub problems $P_1, P_2, . . . , P_k$ are solved by recursive application of DANDC.

Computing time of DANDC is

- If the sizes of the two sub problems are approximately equal

$$T(n) = \begin{cases} T(1) & n = 1 \\ aT(n/b) + f(n) & n > 1 \end{cases}$$

-
- Where, T (n) is the time for DANDC on ‘n’ inputs
- T(1) is the time to complete the answer directly for small inputs
- f (n) is the time for Divide and Combine

- Example
  - Merge Sort
  - Binary search
  - Strassen's Matrix Multiplication

Matrix Multiplication

- Let A and B be two n x n matrices .The product matrix C=AB is also an n x n matrix.
-
```
MATRIX-MULTIPLY(A, B)
if A.columns ≠ B.rows
    error "incompatible dimensions"
else let C be a new A.rows × B.columns matrix
    for i = 1 to A.rows
        for j = 1 to B.columns
            c_ij = 0
            for k = 1 to A.columns
                c_ij = c_ij + a_ik · b_kj
    return C
```
- The time for the resulting matrix multiplication is $O(n^3)$.

divide- and –conquer stratergy

•Assume that n is a power of 2,$n=2^k$.
• Matrix A and B are each partitioned into 4 square submatrices , each submatrix having dimensions n/2 X n/2.
•Then the product of AB can be computed for the product of 2 X2 matrices is

$$\begin{array}{c} a \qquad b \\ \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \\ c \qquad d \end{array} \begin{array}{c} e \qquad f \\ \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right] \\ g \qquad h \end{array} = \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

•To compute AB, we need to perform 8 multiplications of n/2 X n/2 matrices and 4 additions of n/2 X n/2 matrices.
The computing time T(n) of the resulting divide- and –conquer algorithm is given by the recurrence

Time complexity analysis

$$T(n) = \begin{cases} b & n \leq 2 \\ 8T(n/2) + cn^2 & n > 2 \end{cases}$$

- Solving the above recurrence relation $T(n)=8T(n/2)+cn^2$

  $a=8, b=2, f(n)=n^2$

$n^{\log_2 8} = n^3 > n^2$

- we obtained $T(n)=O(n^3)$

Strassen's method

- It is used to improve the time complexity of matrix multiplication.
- Strassen's method is based on divide and conquer method
- In the sense that this method also divide matrices to sub-matrices of size N/2 x N/2.
- Since matrix multiplication are more expensive than matrix additions ($O(n^3)$ versus $O(n^2)$),we can attempt to reformulate the equations of Cij so as to have fewer multiplications and possibly more additions.
- Volker strassen has discovered a way to compute the C_{ij} using only 7 multiplications and 18 additions or subtractions.
- In this method, first compute the seven n/2 X n/2 matrices P, Q, R, S, T, U and V. Then the C_{ij}'s are computed.

$$
\begin{aligned}
P &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
Q &= (A_{21} + A_{22})B_{11} \\
R &= A_{11}(B_{12} - B_{22}) \\
S &= A_{22}(B_{21} - B_{11}) \\
T &= (A_{11} + A_{12})B_{22} \\
U &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
V &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{aligned}
$$

$$
\begin{aligned}
C_{11} &= P + S - T + V \\
C_{12} &= R + T \\
C_{21} &= Q + S \\
C_{22} &= P + R - Q + U
\end{aligned}
$$

The resulting recurrence relation for $T(n)$ is

$$T(n) = \begin{cases} b & n \leq 2 \\ 7T(n/2) + an^2 & n > 2 \end{cases}$$

where a and b are constants. Working with this formula, we get

$$\begin{aligned} T(n) &= an^2[1 + 7/4 + (7/4)^2 + \cdots + (7/4)^{k-1}] + 7^k T(1) \\ &\leq cn^2(7/4)^{\log_2 n} + 7^{\log_2 n},\ c \text{ a constant} \\ &= cn^{\log_2 4 + \log_2 7 - \log_2 4} + n^{\log_2 7} \\ &= O(n^{\log_2 7}) \approx O(n^{2.81}) \end{aligned}$$

Q3.Multiply the following two matrices using Strassen's Matrix Multiplication Algorithm. (May 2019-5 marks)

$$A = \begin{bmatrix} 6 & 8 \\ 9 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Apply stressen's formula,we will get

P=(A11+A22)(B11+B22)=104 Q=(A21+A22)B11 = 32

R= A11(B12-B22) = -6 S= A22(B21-B11)=7

T=(A11+A12) B22=84 U=(A21-A11)(B11+B12)=21

V= (A12-A22)(B21+B22) =9

C11=P+S-T+V =36 C12 = R+T =78

C21= Q+S = 39 C22= P+R-Q+U =87

Is Strassen’s method of matrix multiplication suitable for practical applications? Justify your answer. (December 2018-3marks)

- Generally Strassen’s Method is not preferred for practical applications for following reasons.
    1. The constants used in Strassen’s method are high and for a typical application Naive method works better.
    2. For Sparse matrices, there are better methods especially designed for them.
    3. The submatrices in recursion take extra space.
    4. Because of the limited precision of computer arithmetic on noninteger values, larger errors accumulate in Strassen’s [algorithm](#) than in Naive Method.

Example 1. *Use Strassen's algorithm to compute the product to two given square matri*

(UI

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \textit{ and } \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Solution. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ and } B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

So

$$\begin{matrix} A_{11} = 1 & A_{12} = 2 \\ A_{21} = 3 & A_{22} = 4 \end{matrix} \quad \text{and} \quad \begin{matrix} B_{11} = 5 & B_{12} = 6 \\ B_{21} = 7 & B_{22} = 8 \end{matrix}$$

Now compute P, Q, R, S, T, U, V as follows :

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$= (1+4)(5+8) = 5 \times 13 = 65$$

$$Q = (A_{21} + A_{22}) B_{11}$$
$$= (3+4)5 = 35$$

$$R = A_{11}(B_{12} - B_{22})$$
$$= 1(6-8) = -2$$

$$S = A_{22}(B_{21} - B_{11})$$
$$= 4(7-5) = 8$$

$$T = (A_{11} + A_{12}) B_{22}$$
$$= (1+2)8 = 24$$

$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$= (3-1)(5+6) = 22$$

$$V = (A_{12} - A_{22})(B_{21} + B_{22})$$
$$= (2-4)(7+8) = -30$$

Now computer c_{ij}'s as follows :

$$c_{11} = P+S-T+V$$
$$=65+8-24-30$$
$$=19$$

$$c_{12} = R+T$$
$$=-2+24$$
$$=22$$

$$c_{21} = Q+S$$
$$=35+8$$
$$=43$$

$$c_{22} = P+R-Q+U$$
$$=65-2-35+22$$
$$=50$$

$$\therefore \quad c=\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Thus the required matrix product is $\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$