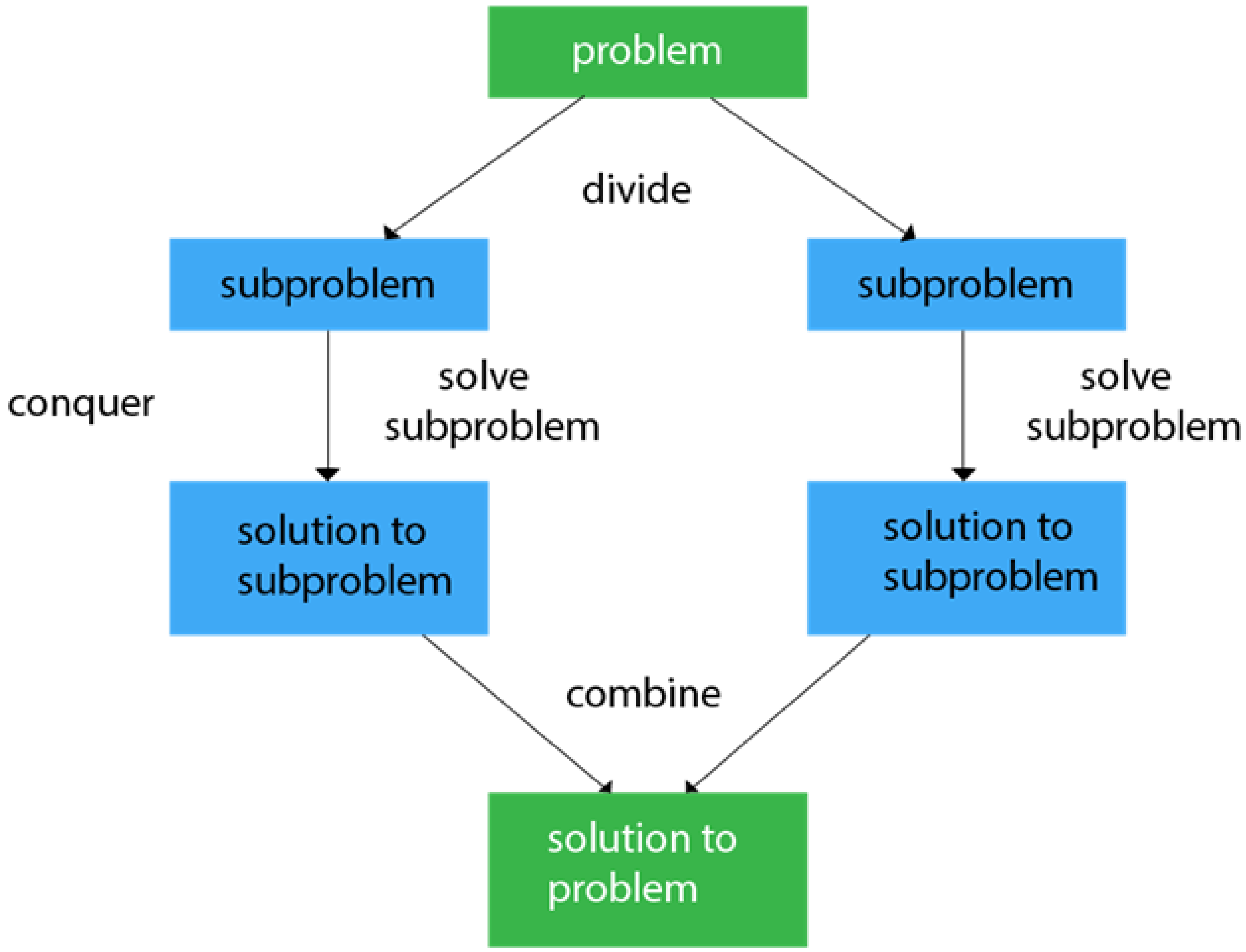
Divide and Conquer approach

- Divide and conquer approach is a top down technique that consists of dividing the problem into two or more sub-problems and recursively solving each sub-problem.
- Divide and conquer algorithm has three steps
  - ✓ **Divide** the problem into two or more sub problems.
  - ✓ **Conquer** the Sub problem by solving them recursively.
  - ✓ **Combine** the solutions of each sub-problem to get the solution of original problem.

✓ Eg: quick sort, binary search, merge sort


problem
divide
subproblem
subproblem
conquer
solve subproblem
solve subproblem
solution to subproblem
solution to subproblem
combine
solution to problem

```
DAndC(P)
{
if(small(P))//if P is small
    return S(P);//solution of P
else
{
Divide P into k sub-problems P1,P2,P3...P_K where k>=1;
Apply DAndC to each sub-problems;
Return(Combine(DAndC(P1),DAndC(P2),..,DAndC(P_K)));
}
}
```

The divide and conquer problem can be represented by using recurrence relation of the form

$$T(n)= \quad aT(n/b)+f(n);$$

$$T(1)=1;$$

The recurrence relation can be solved by substitution method, iteration method, recursion tree or by using master's theorem

Merge Sort

- Merge sort is one for sorting algorithm based on divide and conquer approach.
- Given an array, the merge sort algorithm divides the array into two sub-arrays and recursively sort each sub-arrays and merge two sorted sub-arrays.
- The major operation associated with merge sort is merging which combines two sorted arrays to form a new sorted array.

INPUT: An integer array A[low…high] with low and high be the index of the first and last element.

OUTPUT: The sorted array A[low..high];

Merge-sort(A,low,high)

{if(low>=high) return;

else

{

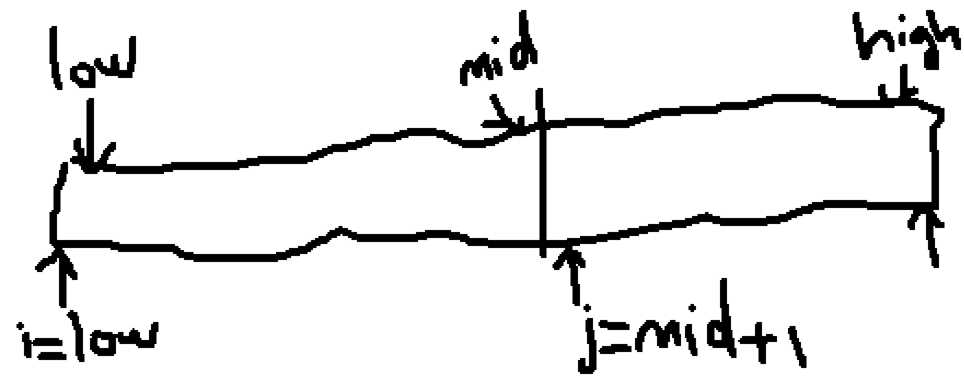
Mid=(low+high)/2;

Merge-sort(A,low,mid);

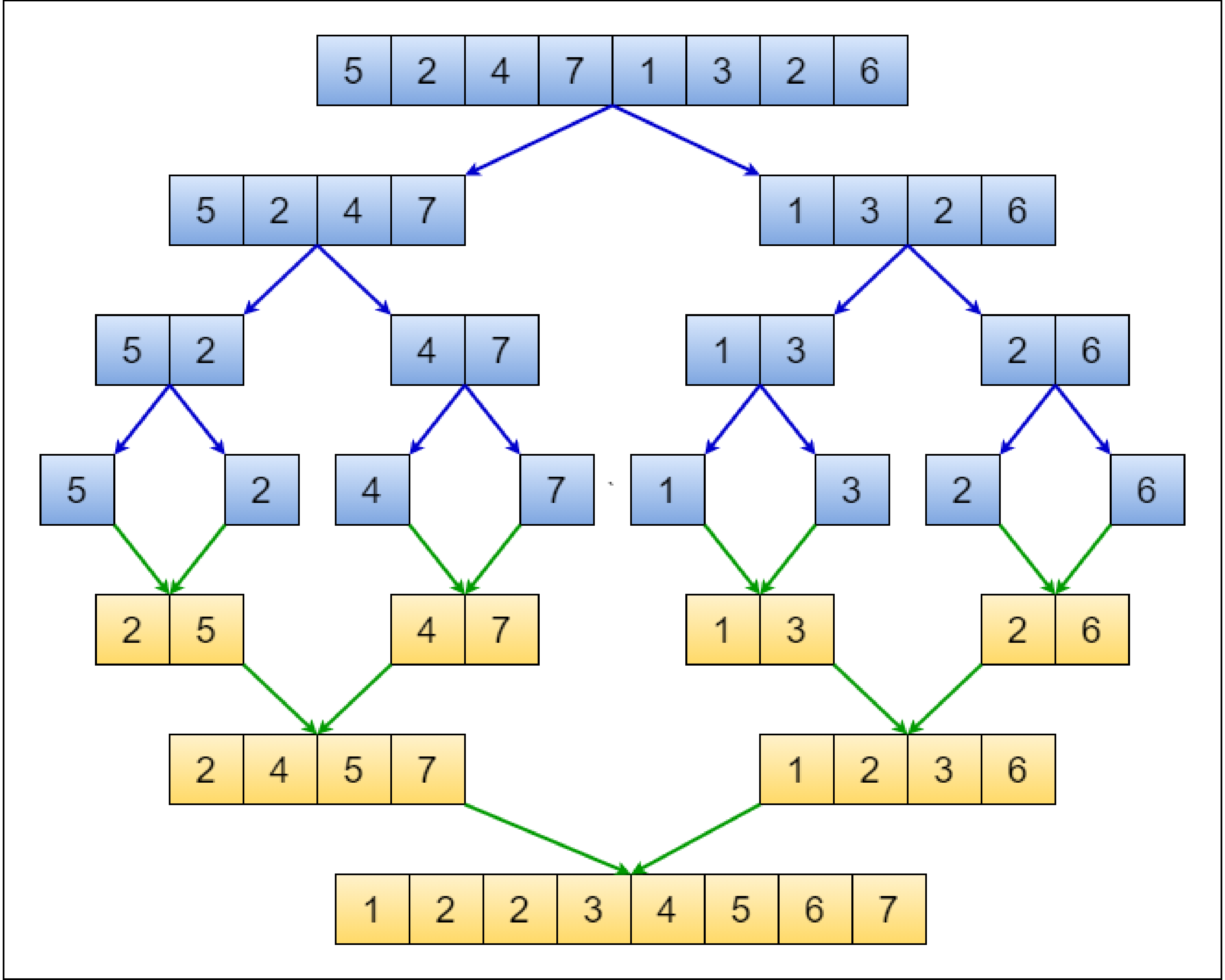
Merge-sort(A,mid+1;high);

Merge(A,low,mid,high);

}}



```
Merge(A,low,mid,high)
{
i=low;j=mid+1;k=low;
While(i<=mid && j<=high)
    if(A[i]>A[j])
          B[k++]=A[j++];
    else
           B[k++]=A[i++];
While(i<=mid)
    B[k++]=A[i++];
While(j<=high)
    B[k++]=A[j++];
For(i=low;i<=high;i++)
A[i]=B[i];
}
```


5 2 4 7 1 3 2 6
5 2 4 7
1 3 2 6
5 2
4 7
1 3
2 6
5
2
4
7
1
3
2
6
2 5
4 7
1 3
2 6
2 4 5 7
1 2 3 6
1 2 2 3 4 5 6 7

Complexity

- ✓ The merge sort algorithm can be represented by using recurrence relation

$$T(n)=2T(n/2)+n, T(1)=1$$

By solving this $T(n)=O(n\lg n)$.

- ✓ The major drawback of merge sort is that it requires an auxiliary array of size n.

Strassen's Matrix Multiplication

- Let A and B be two nxn matrices. The product matrix C=A*B is also an nxn matrix whose element can be

$$C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$$

- To compute C(i,j), we need n multiplication. As matrix C has n^2 elements, the total number of multiplication required is n^3
- This method is called naïve method. Time complexity is $O(n^3)$

- The divide and conquer is another method to multiply two nxn matrices.
- Let A and B be the two matrices of order nxn and n be the power of 2. A and B are partitioned into four sub-matrices each having order n/2xn/2.
- Recursively partition each sub-matrix, until its order becomes 2x2.

- Let A and B be the two matrices of order 2x2, C=A*B

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

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

- The above method for matrix multiplication requires 8 scalar multiplication and four addition.
- The complexity of above method can be expressed as $T(n)=8T(n/2)+n^2 =O(n^3)$.

- Strassen’s Matrix multiplication can be performed only on **square matrices** where **n** is a **power of 2**.If matrices are not the power of 2, then matrix is padded with 0
- Strassen algorithm is a recursive method for matrix multiplication where we divide the matrix into 4 sub-matrices of dimensions n/2 x n/2 in each recursive step.
- Let A and B are two 2x2 matrices, C=A*B

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

•To multiply C=A*B, we compute the values of seven variables by using elements of A and B as follows.

P = (A11 + A22)(B11 + B22)

Q = (A21 + A22) B11

R = A11 (B12 – B22)

S = A22 (B21 – B11)

T = (A11 + A12) B22

U = (A21 – A11) (B11 + B12)

V = (A12 – A22) (B21 + B22)

C11 = P + S – T + V

C12 = R+T

C21 = Q + S

C22 = P+R -Q + U

The values of matrix C is obtained from the above seven constants

- The strassen matrix multiplication requires 7 multiplication and 18 addition.
- The complexity can be represented as $T(n)=7T(n/2)+n^2=O(n^{2.81})$
- The drawback stressen’s matrix multiplication is that it requires too many constants