Divide and Conquer

General Method:-

Given a function to compute on n inputs. The divide and conquer strategy suggests splitting the inputs into k distinct subsets $1 < k \leq n$, yielding k subproblems. These subproblems must be solved, and then a method must be found to combine subsolutions into a solution of the whole.

If the subproblems are still large, then the divide-and-conquer strategy can possibly reapplied. Smaller and Smaller subproblems of the same kind are generated until eventually subproblems that are small enough to be solved without splitting are produced.

The reapplication of the divide and conquer principle is naturally expressed by a recurrence algorithm.

Algorithm DAndC(P)

```
{
    if small(P) then return S(P);
    else
    {
        Divide P into smaller (instances)
        subproblems P1, P2, ...., PK , K ≥ 1
        Apply DAndC to each of these subproblems;
        return Combine (DAndC(P1), DAndC(P2),...
                        DAndC(PK));
    }
}
```

Control Abstraction for divide-and-conquer

→ P is the problem to be solved.

→ Small(P) is a Boolean-valued function that determines whether the input size is small enough that the answer can be computed without splitting.

→ If that is so, the function S is invoked.

S(P) — solution of problem P.

Combine is a function that determines the solution to P by using the solutions of the K subproblems.

If the size of P is n and the sizes of the k subproblems are $n_1, n_2 \ldots n_k$ respectively,

Then the computing time of the DAndC is described by the recurrence relation.

$$\boxed{T(n) = \begin{cases} g(n) & \text{if } n \text{ is small} \\ T(n_1) + T(n_2) + \ldots T(n_k) + f(n), & \text{otherwise} \end{cases}}$$

→ $g(n)$ is the time to compute the answer directly for small input problems.

→ The function $f(n)$ is the time for dividing P and combining the solutions of subproblems.

→ The complexity of many divide-and-conquer algorithms is given by recurrence relations of the form.

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

where a and b are known constants.

→ We assume that $T(1)$ is known and n is a power of b (i.e., $n = b^k$)

→ substitution method is used to solve recurrence relations.

Applications of Divide and Conquer

1) Binary Search
2) Merge sort
3) Quick Sort
4) Strassen's matrix multiplication.