DECREASE AND CONQUER TECHNIQUE

As divide-and-conquer approach is already discussed, which include following steps: Divide the problem into a number of sub problems that are smaller instances of the same problem. Conquer the sub problems by solving them recursively.

Decrease and conquer is a technique used to solve problems by reducing the size of the input data at each step of the solution process. This technique is similar to divide-and-conquer, in that it breaks down a problem into smaller sub problems, but the difference is that in decrease-and-conquer, the size of the input data is reduced at each step. The technique is used when it's easier to solve a smaller version of the problem, and the solution to the smaller problem can be used to find the solution to the original problem.

IMPLEMENTATION OF DECREASE AND CONQUER

This approach can be either implemented as top-down or bottom-up. Top-down approach: It always leads to the recursive implementation of the problem. Bottom-up approach: It is usually implemented in iterative way, starting with a solution to the smallest instance of the problem.

Variations of Decrease and Conquer:

There are three major variations of decrease-and-conquer:

1. Decrease by a constant
2. Decrease by a constant factor
3. Variable size decrease

Decrease by a Constant: In this variation, the size of an instance is reduced by the same constant on each iteration of the algorithm. Typically, this constant is equal to one, although other constant size reductions do happen occasionally. Below are example problems:

- Insertion sort
- Graph search algorithms: DFS, BFS
- Topological sorting
- Algorithms for generating permutations, subsets

Decrease by a Constant factor: This technique suggests reducing a problem instance by the same constant factor on each iteration of the algorithm. In most applications, this constant factor is equal to two. A reduction by a factor other than two is especially rare. Decrease by a constant factor algorithms are very efficient especially when the factor is greater than 2 as in the fake-coin problem. Below are example problems:

- Binary search
- Fake-coin problems
- Russian peasant multiplication

Variable-Size-Decrease: In this variation, the size-reduction pattern varies from one iteration of an algorithm to another. As, in problem of finding god of two number though the value of the second argument is always smaller on the right-hand side, it decreases neither by a constant nor by a constant factor. Below are example problems:

Computing median and selection problem.

- Interpolation Search
- Euclid's algorithm

There may be a case that problem can be solved by decrease-by-constant as well as decrease-by-factor variations, but the implementations can be either recursive or iterative. The iterative implementations may require more coding effort; however, they avoid the overload that accompanies recursion