

*DEPARTMENT:*

**SOFTWARE ENGINEERING LEVEL ONE**

***COURSE***: FUNDAMENTALS OF ALGORITHM

***COURSE FACILITATOR***:

Mr. TEKOH PALMA

**EXPOSE EXERCISE**

***WORK REQUIRED:*** Expose on divide and conquer algorithms how to solve a problem using a divide and conquer algorithms, sample problems and their divide and conquer solutions, implementation in any programming language of your choice.

GROUP MEMBERS:

1. AMUNGWA KINGSLY
2. NJU PAUL NTUM
3. NJOMBIKIRI BEVANIE FOMONYUY
4. NSANGO BLESS

**Divide And Conquer Algorithms**

***Introduction***

They are also often referred to as "recursive algorithms," divide and conquer algorithms start with a big problem and work their way down to smaller subproblems. They then combine the solutions from each subproblem to solve the main problem.

***Advantages (uses) of divide and conquer algorithms***

**\*Efficiency**

Divide and conquer algorithms can benefit from a feature known as "divide and conquer efficiency," which makes them frequently more efficient than other algorithms. This condition indicates that the execution time of a divide and conquer method will be at most O (N log N) when applied to a problem with N sub-problems. This indicates that divide and conquer algorithms are highly effective in resolving issues that may be divided into numerous smaller issues.

**\*Flexibility**

As for flexibility, the divide and conquer approach can be applied to a wide range of problems, making it a very useful technique. Yes, divide and conquer algorithms are very flexible because they can be applied to a wide range of problems. One way to think about this flexibility is to consider the different levels of divide and conquer algorithms. At the lowest level, divide and conquer algorithms can be applied to basic mathematical operations like addition, subtraction, and multiplication.

At a higher level, divide and conquer algorithms can be used to solve more complex problems like sorting, searching, and graph traversal. And at the highest level, divide and conquer algorithms can be used to solve problems that involve optimization, such as the traveling salesman problem.

***Disadvantages (limitations) of divide and conquer algorithms***

There are some restrictions on divide and conquer algorithms that need to be considered.

One drawback is that they can be challenging to apply to issues that are hard to break down. This implies that some issues might not be suitable to divide and conquer algorithms because they cannot be effectively divided into smaller subproblems.

Divide and conquer algorithms may use more memory than other kinds of algorithms, which is another drawback. This is because, in order to solve the main problem, the smaller subproblems must first be stored independently.

***How To Solve A Problem Using Divide And Conquer Algorithms***

Finding the subproblems into which the main problem can be divided is the first step in applying divide and conquer to solve a problem. After determining which subproblems exist, you must figure out how to address each one recursively. After resolving the subproblems, you can solve the main problem by combining the solutions. To demonstrate this procedure, let's go over an example problem together. Let's say you are asked to identify the second-highest number in a list of numbers. You could split the list into two sub-lists and identify the second one to solve this problem.

***Sample Problems Of A Divide And Conquer Algorithm***

Okay, let's examine an illustration of a divide and conquer algorithm in operation.

***Here are a few sample problems and their divide and conquer solutions:***  
1. Binary Search

Given a sorted array and a target value, find the index of the target value in the array. The divide and conquer solution for this problem involves dividing the array into two halves, comparing the target value with the middle element, and recursively searching in the left or right half of the array based on the comparison result.  
  
2. Merge Sort

The merge sort algorithm, which sorts a list of numbers, is a well-known example. The list is initially split into two smaller lists in order for the algorithm to function. It then sorts each of the smaller lists in a recursive manner. The two sorted lists are finally combined into a single sorted list.

Given an unsorted array, sort it in ascending order. The divide and conquer solution for this problem involves dividing the array into two halves, recursively sorting each half, and then merging the sorted halves to obtain the final sorted array.  
  
3. Maximum Sub-array Sum

Given an array of integers, find the contiguous sub-array with the largest sum. The divide and conquer solution for this problem involves dividing the array into two halves, finding the maximum sub-array sum in the left and right halves, and then combining the solutions to handle the case where the maximum sub-array crosses the midpoint.