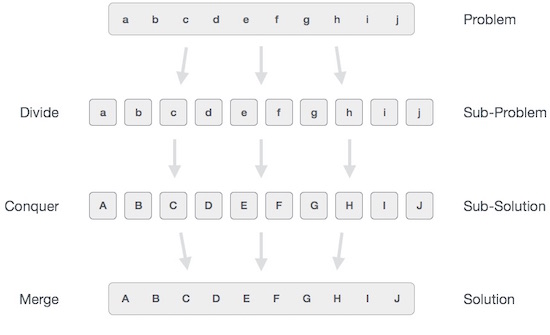
**DIVIDE AND CONQUER**

In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.



Broadly, we can understand **divide-and-conquer** approach in a three-step process.

**Divide/Break**

This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, sub-problems become atomic in nature but still represent some part of the actual problem.

**Conquer/Solve**

This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

**Merge/Combine**

When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer & merge steps works so close that they appear as one.

**Examples**

The following computer algorithms are based on **divide-and-conquer** programming approach −

* Merge Sort
* Quick Sort
* Binary Search
* Strassen's Matrix Multiplication
* Closest pair (points)

There are various ways available to solve any computer problem, but the mentioned are a good example of divide and conquer approach.

***PROGRAM FOR QUICK SORT***

***// quick sort.***

***#include<stdio.h>***

***#include<conio.h>***

***int arr[40];***

***void quicksort(int a[],int p,int r);***

***int partition(int a[],int p,int r);***

***void exchange(int i,int j);***

***void quicksort(int a[],int p,int r)***

***{***

***int q;***

***if(p<r)***

***{***

***q=partition(a,p,r);***

***quicksort(a,p,q-1);***

***quicksort(a,q+1,r);***

***}***

***}***

***int partition(int a[],int p,int r)***

***{***

***int x,j,i;***

***x=a[r];***

***i=p-1;***

***for(j=p;j<=(r-1);j++)***

***if(a[j]<x)***

***{***

***i=i+1;***

***exchange(i,j);***

***}***

***exchange(i+1,r);***

***return(i+1);***

***}***

***void exchange(int i,int j)***

***{***

***int temp;***

***temp=arr[i];***

***arr[i]=arr[j];***

***arr[j]=temp;***

***}***

***void main()***

***{***

***int n,i;***

***clrscr();***

***printf("\n Enter no. of elements");***

***scanf("%d",&n);***

***printf("\nEnter elements:");***

***for(i=1;i<=n;i++)***

***scanf("%d",&arr[i]);***

***quicksort(arr,1,n);***

***printf("\nSorted arrray is:");***

***for(i=1;i<=n;i++)***

***printf("%d\n",arr[i]);***

***getch();***

***}***