Divide and Conquer

Algoritma divide and conquer merupakan algoritma yang digunakan untuk memecah sebuah permasalahan yang kompleks menjadi beberapa masalah-masalah kecil sehingga akan didapat penyelesaian masalah yang lebih mudah. Menurut beberapa sumber algoritma ini cukup populer dan banyak digunakan dalam penyelesaian masalah” yang cukup kompleks

Cara kerja divide and conquer adalah

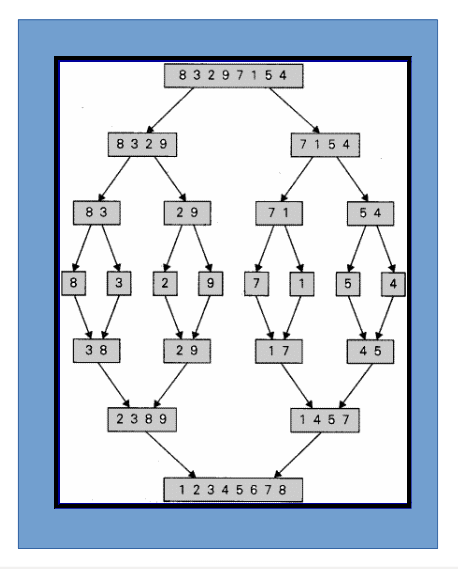
Melakukan divide yaitu proses yang memecah kekompleksan sebuah problem dengan membagi problem tersebut menjadi beberapa proses yang memiliki kemiripan namun dapat diselesaikan hanya dalam beberapa langkah saja

Dan conquer yaitu proses untuk memecahkan masalah” yang sudah di divide tadi umumnya secara rekursif

Dan langkah terakhir yaitu menggabungkan masalah” hasil proses divide and conquer dan menyatukan kembali untuk membentuk solusi penyelesaian masalah tersebut.

Menurut sumber [ensiklopedia.kurikulum.org](http://ensiklopedia.kurikulum.org/id1/1-1489-1388/Divide-And-Conquer_27771_m-independen.html), [id.wikipedia.org](http://id.wikipedia.org), [ensiklopedia.web.id](http://ensiklopedia.web.id) proses devide and conquer memiliki langkah” **Divide,** **Conquer,** dan **Combine.**

Contoh simple adalah merge sort yang mennggunakan divine and conquer algotythm



Dengan menggunakan potongan program sebagai berikut

void merge\_sort (arr[],L, R)

{

if(L >= R)

{

print(“gagal”)

}  
/\*arr akan dipisah menjadi 2 part  
arr 1 adalah dari awal menuju ke mid dan yang kedua adalah arr dari index mid+1 ke R/ end dari arr\*/

int mid = L+(R-L)/2;  
merge\_sort(arr,L,mid);  
merge\_sort(arr,mid+1,R);  
merge(ara,L,mid,R); //function untuk menggabungkan kembali array yang sudah dipisah

}

void Merge(arr[], L, mid,R)

{

for(int i = L; i <= mid; i++ )

{ArrayKiri[i - L] = arr[i];}

for (int i = mid + 1; i <= R; i++ )

{ArrayKanan[i - mid - 1] = arr[i];}

int indexkiri = 0, indexkanan = 0;

for (int k = L; k <= R; k++){

{

if(leftArray[indexkiri] <= rightArray[indexkanan])

{arr[k] = leftArray[indexkiri++];}

else

{arr[k] = rightArray[indexkanan++];}  
 }

}

}