

Diviser Pour Régner

Introduction aux algorithmes

Ce sont des techniques classiques pour résoudre un problème donné.

Par exemple : trier une liste d'éléments. On appelle alors une instance du problème une liste en particulier.

Un algorithme qui résout un problème P permet de donner une solution à chaque instance.

I - Principe

Pour résoudre un problème P sur une instance I :

- [0] - Cas de base : Les “petites” instances sont résolus immédiatement.
- [1] - Diviser : On “découpe” I en sous-instances I_1, \dots, I_k .
- [2] - Régner : On résout (récursivement) les sous-instances.
- [3] - Combiner : On combine les solutions des instances I_1, \dots, I_k pour donner une solution à l'instance I .

Remarque : on appelle les sous-instances des sous-problèmes.

II - Exemples

Tri fusion

Listes OCaml

```
1  let rec divide = function
2    | [] -> [], []
3    | [e] -> [e], []
4    | a::b::l ->
5      let (l1, l2) = divide l in
6      (a::l1, b::l2);;
7
8  let rec fusion l1 l2 = match l1, l2 with
9    | [], l2 -> l2
10   | l1, [] -> l1
11   | h1::t1, h2::t2 when h1 < h2 -> h1 :: (fusion t1 (h2::t2))
12   | h1::t1, h2::t2 -> h2 :: (fusion (h1::t1) t2);;
13
14
15 let rec fusion_sort l =
16   (*Cas de base*)
17   if List.length l < 2 then l
18   else
19     (*DIVISER*)
20     let (l1, l2) = divide l in
21     (*REGNER*)
22     let (l3, l4) = (fusion_sort l1, fusion_sort l2) in
23     (*COMBINER*)
```

ocaml

24

fusion l3 l4;;

Tableaux OCaml

```
1 let divide a =  
2   let len = Array.length a in  
3   let l1 = Array.init (len/2) (fun i -> a.(i)) in  
4   let l2 = Array.init (len-len/2) (fun i -> a.(len-1-i)) in  
5   l1,l2;;  
6  
7 let fusion_arr a1 a2 =
```

ocaml