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 $I_1, ...I_k$.
- [2] - Régner : On résout (récursivement) les sous-instances.
- [3] - Combiner : On combine les solutions des instances $I_1, ...I_k$ pour donner une solution à l'instance I.

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

II - Exemples

Tri fusion

Listes OCaml

```
let rec divide = function
    | [] -> [],[]
    | [e] -> [e],[]
    | a::b::l ->
    let (l1,l2) = divide l in
    (a::l1, b::l2);;

let rec fusion l1 l2 = match l1,l2 with
    | [],l2 -> l2
    | l1,[] -> l1
    | h1::t1, h2::t2 when h1 < h2 -> h1 :: (fusion t1 (h2::t2))
    | h1::t1, h2::t2              -> h2 :: (fusion (h1::t1) t2);;


let rec fusion_sort l =
    (*Cas de base*)
    if List.length l < 2 then l
    else
        (*DIVISER*)
        let (l1,l2) = divide l in
        (*REGNER*)
        let (l3,l4) = (fusion_sort l1, fusion_sort l2) in
        (*COMBINER*)
        fusion l3 l4;;
```

Tableaux OCaml

```
let divide a =
    let len = Array.length a in
    let l1 = Array.init (len/2) (fun i -> a.(i)) in
```

```
    let l2 = Array.init (len-len/2) (fun i -> a.(len-1-i) in
    l1,l2;;

let fusion_arr a1 a2 =
```