# Chapitre 1. Arbres binaires de recherche. Tas.

**Exercice 2** On définit le type d'arbres binaires :

```
type 'a arbre =
    |Vide
    |N of 'a * 'a arbre * 'a arbre;;
```

a) Écrire une fonction `arbre_de_tas : int array -> int arbre` qui transforme,un tas codé par un tableau en un arbre codé par le type `int arbre`.

On utilise une fonction auxiliaire récursive : `aux i` retourne le sous-arbre de racine `t.(i)`

```
let arbre_de_tas t =
    let rec aux i =
        if i > t.(0) then Vide
        else N(t.(i), aux (2 * i), aux(2 * i + 1)) in
    aux 1;;
```

b) Écrire la fonction réciproque `tas_d_arbre` de la fonction précédente.

On aura besoin d'une fonction donnant la taille d'un arbre du type `'a arbre`

```
let rec taille a = match a with
    |Vide -> 0
    |N(_,g,d) -> 1 + taille g + taille d;;
```

Pour écrire la fonction demandée, on crée un tableau `t` de taille adéquate qu'on remplit ensuite à l'aide d'une fonction auxiliaire : `aux i b` complète la partie de `t` concernant `t.(i)` et ses fils, `b` étant le sous-arbre de `a` de racine `t.(i)`

```
let tas_d_arbre a =
    let n = taille a in
    let t = Array.make (n + 1) n in
    let rec aux i a =match a with
        |Vide -> ()
        |N(x,g,d) -> t.(i) <- x; aux (2 * i) g; aux (2 * i + 1) d
    in aux 1 a;
    t;;
```