implémn naïve

```
let rec walk tree = match tree with
  | Leaf a              -> [a]
  | Node (a, left, right) -> [a] @ walk left @ walk right
```

complexité pour un peigne à droite

Notons c_n le nombre de $::$.

$$\begin{cases} c_1 = 0 \\ \forall n \in \mathbb{N}^*,\ c_n = 1 + c_{n-2} \end{cases}$$

$\Rightarrow c_n = \Theta(n)$

pour un arbre (quasi-)complet

$$c_n = 1 + \left\lceil \frac{n}{2} \right\rceil + 2c_{\left\lfloor \frac{n}{2} \right\rfloor} = \Theta(n) + 2c_{\frac{n}{2}} = \Theta(n \log_2 n)$$

rem

Un parcours en profondeur préfixe permet de reconstituer l'arbre de départ à condition qu'on y distingue les feuilles des autres noeuds.

3. Algorithme linéaire

pb La fonction est récursive enveloppée

solution une fonction récursive terminale

idée Il est plus simple de parcourir une liste d'arbres

```
let walk_prefix_linear tree =
    let rec aux tosee seen =
      match tosee with
        | [] -> List.rev seen
        | Leaf x :: rest -> aux rest ((L x) :: seen)
        | Node (x, fg, fd) :: rest -> aux (fg :: fd :: rest)
                                          ((N x) :: seen)
    in
    aux [tree] [];;
```