HASH CONSING = PARTAGE MAXIMAL

- application de dictionnaire
- idée "compresser" un arbre en évitant d'avoir plusieurs copies d'un "m̂" sous arbre.

  ↳ en bonus on met l'arbre sous une forme optimisant les calculs par induc° sur la structure de l'arbre. En effet comme on "retient" que deux occurrences d'un m̂ sous arbre sont la m̂ chose, on s'évite de le recalculer.

- hypothèse On suppose avoir une fonction de hachage sur les arbres qui est indépendante de l'adressage.

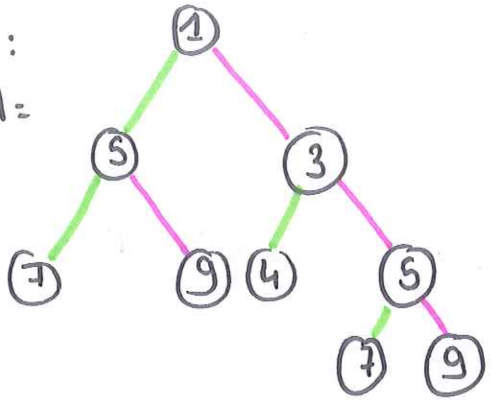
  Ainsi du pt de vue du dictionnaire deux occurrences d'un m̂ sous-arbre sont identifiées.

```
HASH_CONS (A)
  let D = DICO_VIDE in
  let c = 0 in
  let rec f (B) = match B with
    | feuille (x) -> si ! D.EST_CLÉ (B)
                     alors D.(B) <- [x|c] ; c++;
                     retourner D.(B)

    | nd(D,x,G) -> si ! D.EST_CLÉ (B)
                   alors D.(B) <- [x|c] ; c++
                                 /   \
                              f(D)   f(G)
                   retourner D.(B)
  in
  f(A);
  D.(A) ;;
```

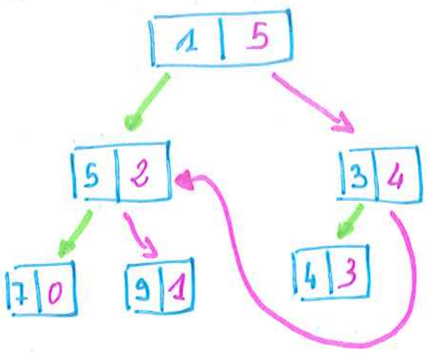
ex :

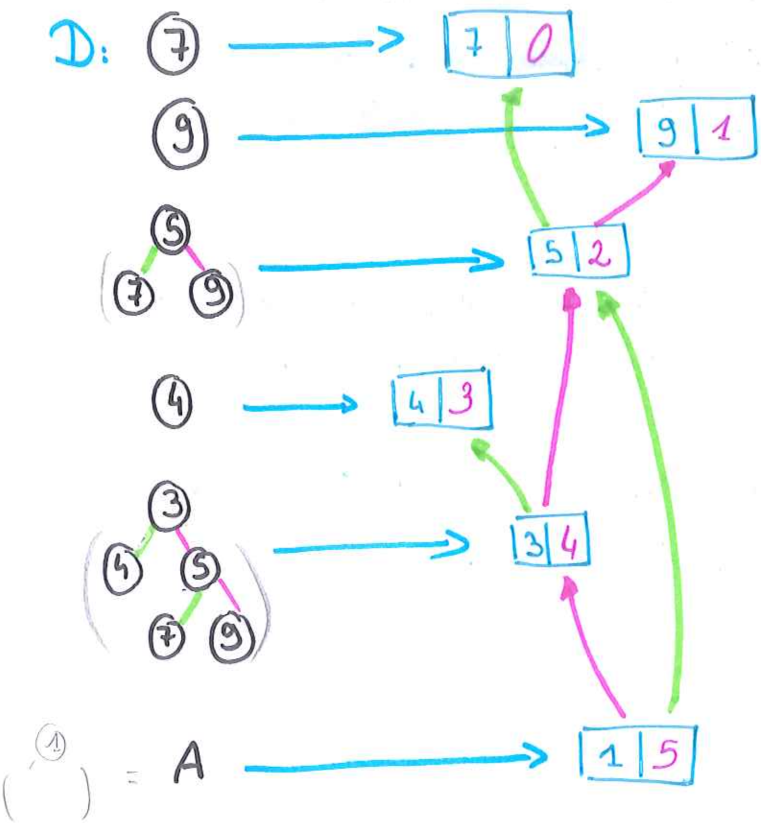
A =



HASH_CONS

« arbre compressé » A'





ex Calcul de la somme des nd sur A.

S :

0	1	2	3	4	5
7	9	21	4	28	50

⚠ Ne pas croire qu'on a voulu calculer d'abord la somme pour le sous arbre "0" et qu'on a pu ! Il n'y a pas de pointeur i ⟶ ss arbre d'indice i.

On a appelé S sur A'. On sait qu'il y a 5+1 ss-arbres dans A. On prépare le tableau résultat. Puis on lance le calcul récursif sur A', en faisant à chaque étape le test "est-ce que j'ai déjà calculé ça ?" en consultant la case du tableau résultat correspondant à l'indice du ss-arbre courant dans A'.