Devoir de Programmation

Algorithmique Avancée

François Malenfer (28706664), Danaël Carbonneau (28709878)

Implémentation de structures de données de recherche (en OCaml)[2]

Représenter des entiers 128 bits en OCaml

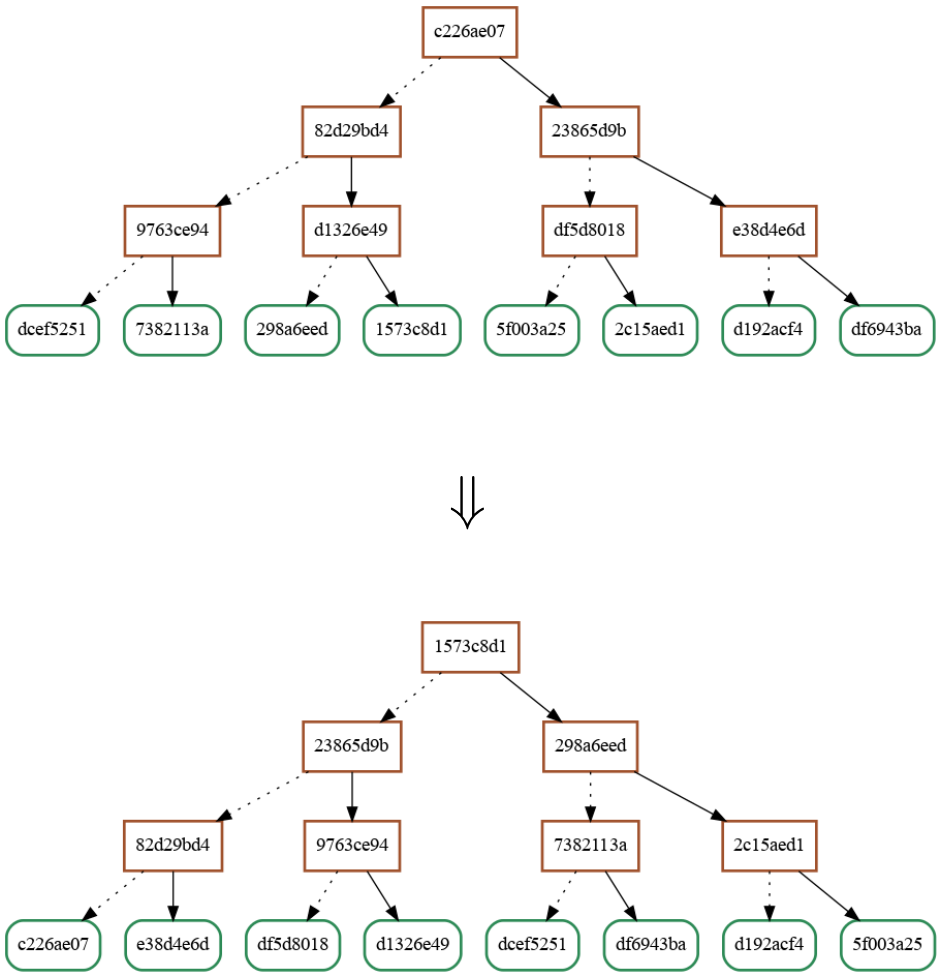
```
1  open Int32;;
2  type entier128 =
3  (Int32.t * Int32.t * Int32.t * Int32.t);;
```

```
1  val cmp : t -> t ->int
2
3  val inf : t -> t -> bool
4
5  val eg : t -> t -> bool
```

Structures[3]

```
1 (*indice dernier element * taille du tableau * tableau
      *)
2 type heapArray = int ref * int ref * (Int128.t option)
      Array.t;;
3
4 (* N of  rang * ndescendants * elt * fg * fd *)
5 type  heapTree = E | L of Int128.t | N of int * int *
      Int128.t *  heapTree *  heapTree;;
```

Construction[1]



$$C = \sum_{i=0}^{h-1} 2^i (h-1-i) \quad (1)$$

$$= \sum_{j=0}^{h-1} 2^{h-1-j} j \quad (2)$$

$$= 2^{h-1} \sum_{j=0}^{h-1} 2^{-j} j \quad (3)$$

$$= 2^{h-1} \sum_{j=0}^{h-1} j \frac{1}{2^j} \quad (4)$$

$$= O(2^{h-1}) \quad (5)$$

$$= O(n) \quad (6)$$

Ajouts Itératifs

Formule pour la régression : $(ax + b)(log(mx + c))$

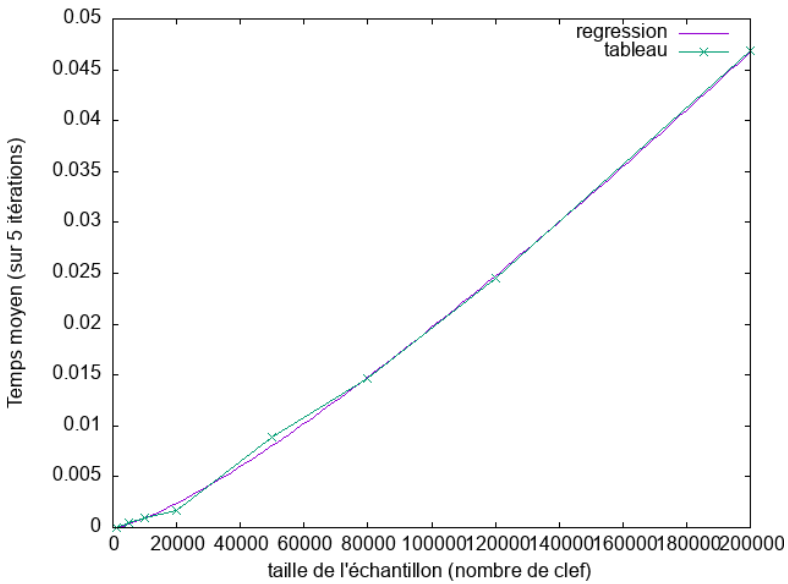


Figure – Tas sous forme de tableau

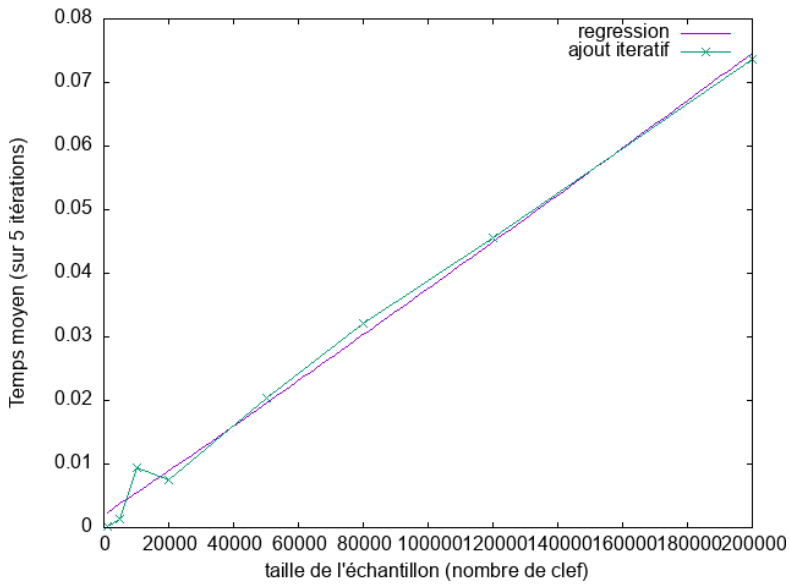


Figure – Tas sous forme d'arbre

Construction

Formule pour la régression : $(ax + b)$

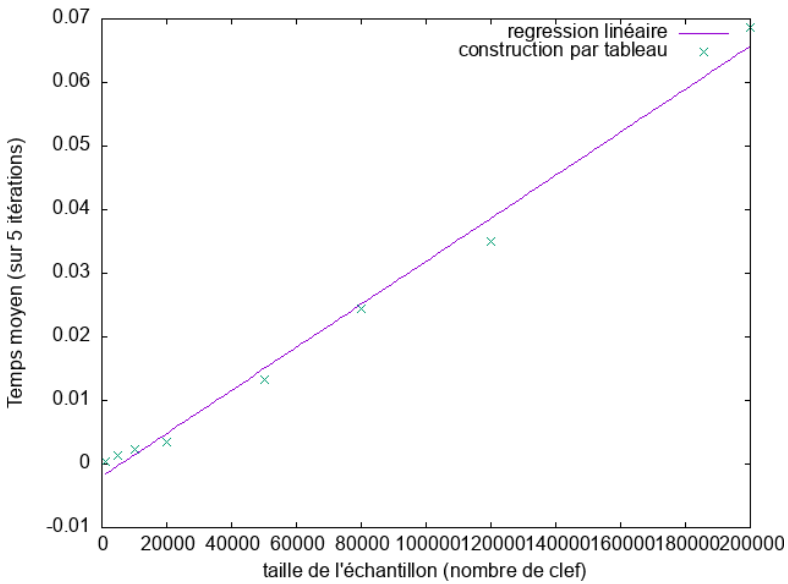


Figure – Tas sous forme de tableau

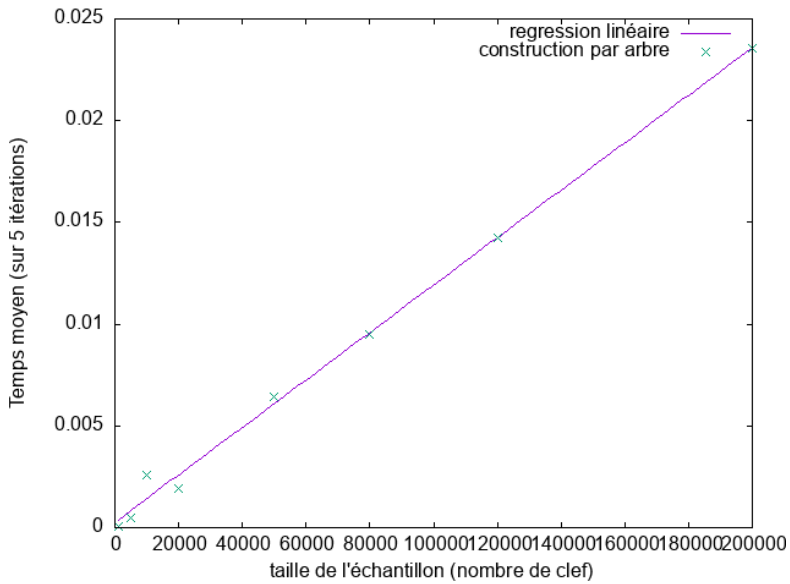


Figure – Tas sous forme d'arbre

Union

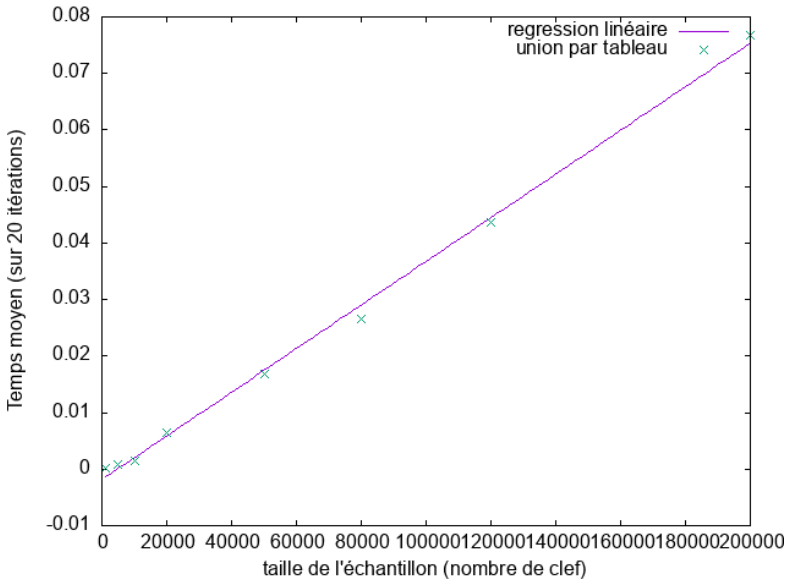


Figure – Tas sous forme de tableau

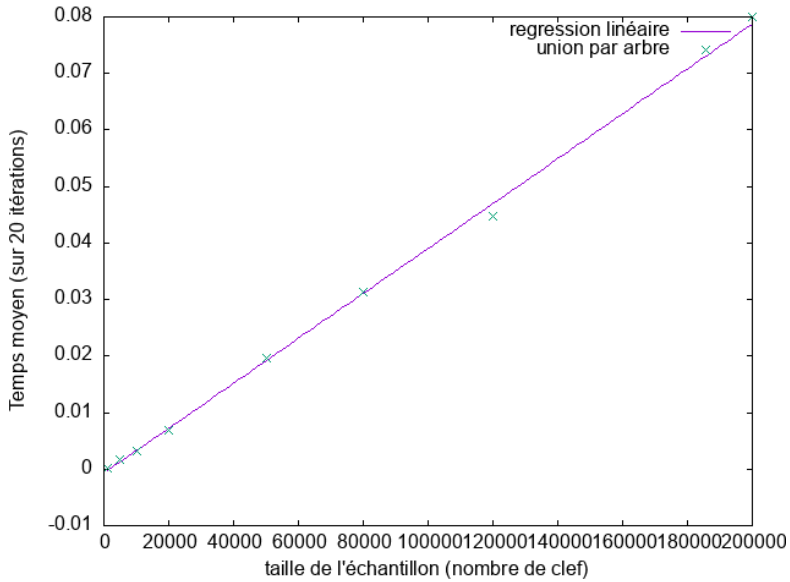


Figure – Tas sous forme d'arbre

Implémentation

```
1  (*Racine(degre,cle,fils)*)
2  type tournois_b = Racine of int * Int128.t * (
       tournois_b list) | Empty
3  (*File(indice,tournois) tournois le plus petit a droite
       de la liste *)
4  type file_b = File of int * (tournois_b list) | Empty
```

Construction et union

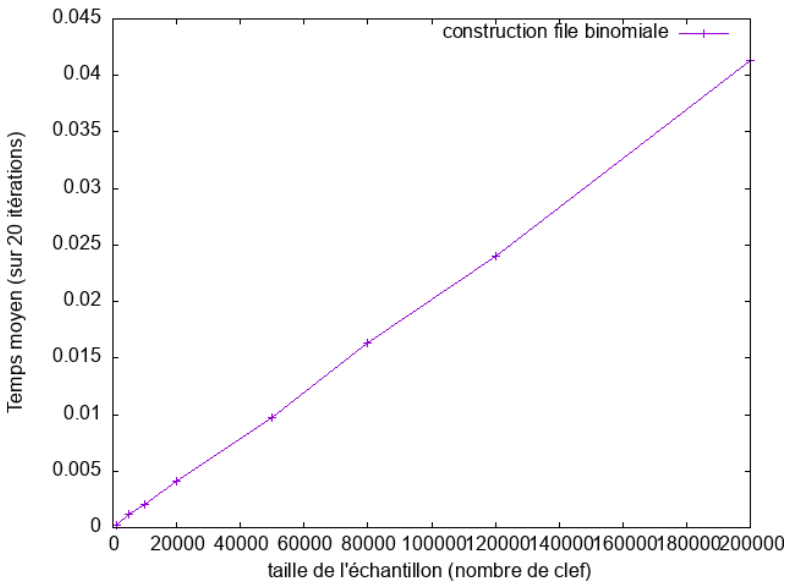


Figure – Complexité de la construction

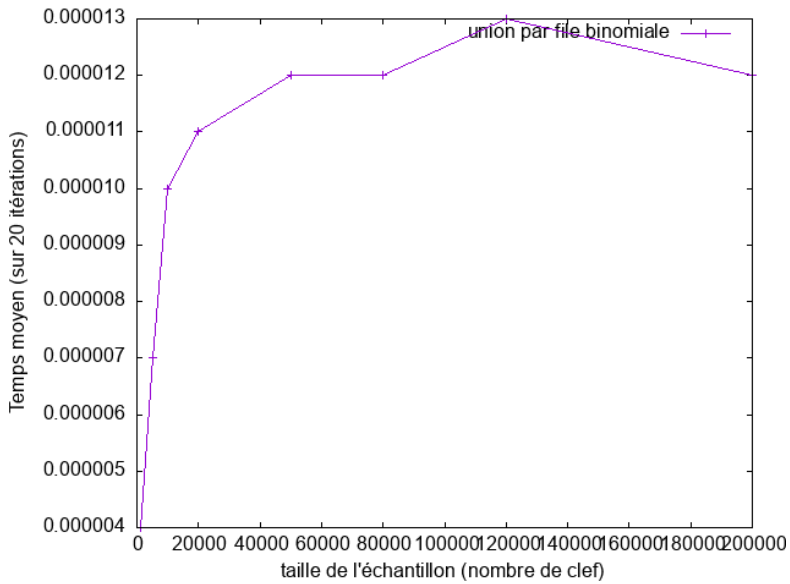


Figure – Complexité de l'union

Implémentation[4]

```
1       val get_int32_le : string -> int -> int32
```

```
1  let finish (entre : int32) : int32 =
2    let res = 0l in
3    let res = Int32.logor res (Int32.shift_left (Int32.
       logand entre 0x000000FFl) 24) in
4    let res = Int32.logor res (Int32.shift_left (Int32.
       logand entre 0x0000FF00l) 8) in
5    let res = Int32.logor res (Int32.shift_right_logical
       (Int32.logand entre 0x00FF0000l) 8) in
6    let res = Int32.logor res (Int32.shift_right_logical
       (Int32.logand entre 0xFF000000l) 24) in
7    res
8  ;;
```

Collisions

$$\mathbb{P} = 1 - \prod_{i=0}^{n-1}(1 - \frac{i}{m}) \tag{7}$$

$$= 1 - exp \prod_{i=0}^{n-1} ln(1 - \frac{i}{m}) \tag{8}$$

$$\approx 1 - e^{-\frac{n^2}{2m}} \tag{9}$$

- $m = 2^{128}$ (nombre de clés possibles codées sur 128 bits)
- $n \approx 271000$ (nombre de mots de la langue anglaise)

$$\mathbb{P} \approx 1 - e^{-\frac{7,3441 \times 10^{10}}{2^{129}}} \approx 1,0791 \times 10^{-28}$$

Choix et et implémentation

```
1  type arbre234 =
2     | Empty
3     | Feuille1 of Int128.t
4     | Feuille2 of Int128.t * Int128.t
5     | Noeud2 of Int128.t * arbre234 * arbre234
6     | Noeud3 of Int128.t * Int128.t * arbre234 * arbre234
       * arbre234
7     | Noeud4 of Int128.t * Int128.t * Int128.t * arbre234
       * arbre234 * arbre234 * arbre234
```

```
1  exception Eclatement of
2  (Int128.t * arbre234 * arbre234) ;;
```

Récupérer la liste de mots

```
1  let filtredroite (ligne : string ) : string = (*...*)
2
3  let filtregauche (ligne : string ) : string = (*...*)
4
5  let extraire_liste (nom : string)
6      (liste : Int128.t list)
7       (arbre : Arbre_234.arbre234) :
8        Int128.t list* Arbre_234.arbre234 = (*...*)
9
10 let extraire_liste_rep (path_rep : string) : Int128.t
       list = (*...*)
```

Comparaison de nos structures

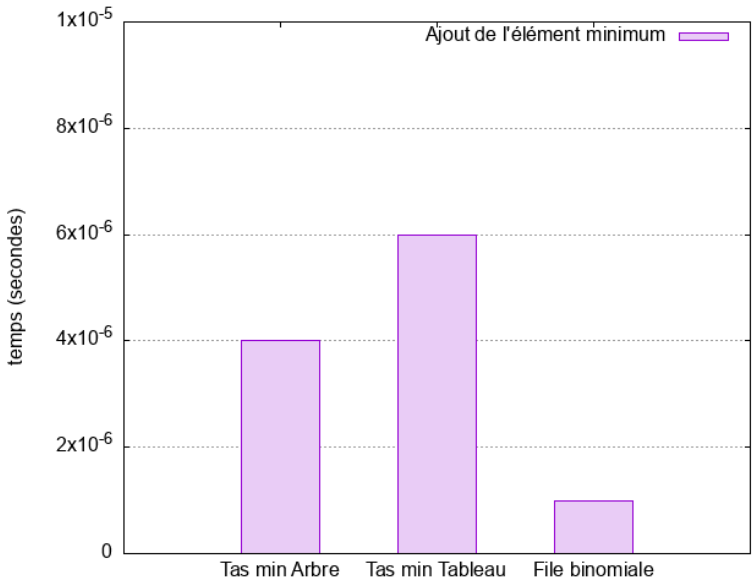


Figure – Ajout

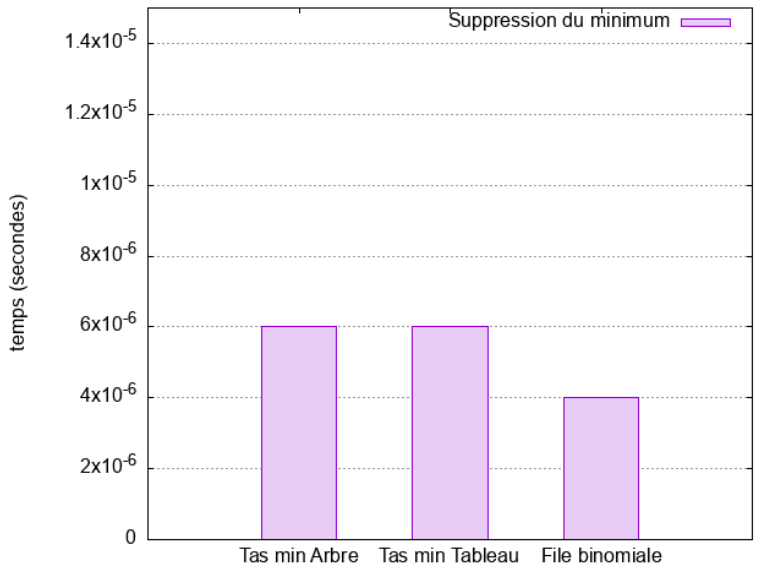


Figure – Suppression du minimum

Comparaison de nos structures (suite)

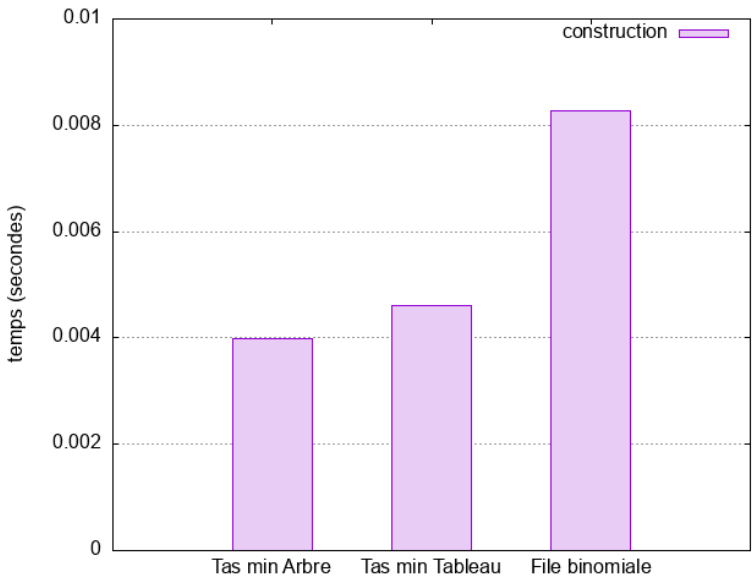


Figure – Construction

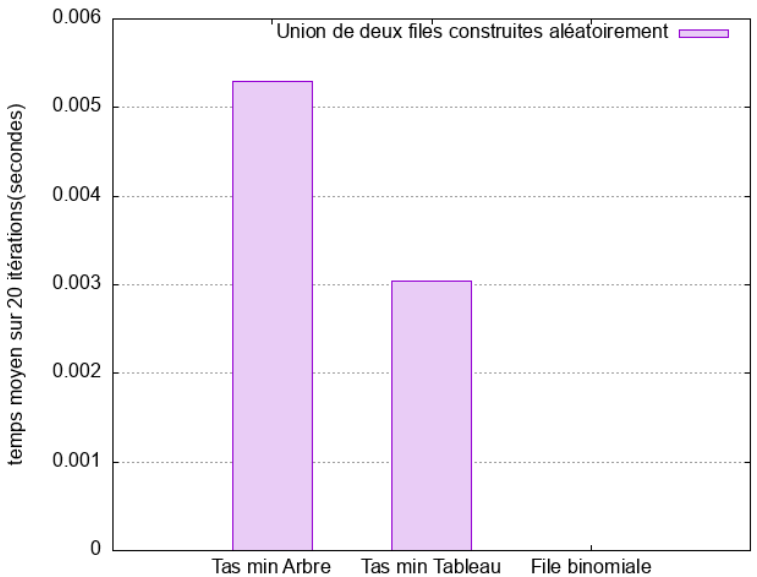


Figure – Union


Robert W. Floyd.
Algorithm 245 : Treesort.
Commun. ACM, 7(12) :701, dec 1964.


Xavier Leroy, Damien Doligez, Alain Frisch, Jacques Garrigue, Didier Rémy, and Jérôme Vouillon.
The ocaml system : Documentation and user's manual.
INRIA, 3 :42.


Chris Okasaki.
Purely Functionnal Data Structures.
Cambridge University Press, 1998.
page 17.


Ronald Rivest.
The md5 message-digest algorithm.
Technical report, 1992.