PROJET ZDD

Zero-Suppressed Binary Decision Diagram

Ariane ZHANG Xue YANG

PLAN

1

STRUTURE DE DONNÉES ET PRIMITIVE

2

COMPRESSION DE L'ARBRE DE DÉCISION ET ZDD

3

GRAPHE EN DOT

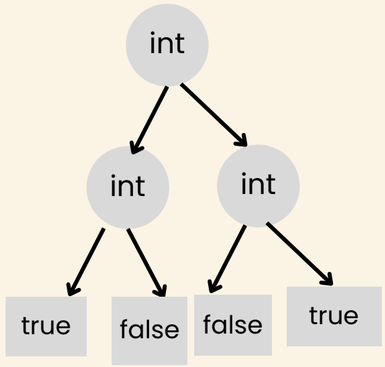
4

ANALYSE

STRUTURE DE DONNÉES ET PRIMITIVE

```
(* liste d'entiers a 64 bit *)
type grand_entier = int64 list;
```

$2^{100} \longrightarrow [0; 2^{36}]$




```
type arbre_decision =
  | Feuille of bool
  | Noeud of int * arbre_decision * arbre_decision
;;
```

```
type liste_deja_vus = (grand_entier * arbre_decision) list
```

```
type arbre_deja_vus =
  | Leaf
  | Node of  arbre_decision option * arbre_deja_vus  * arbre_deja_vus
;;
```

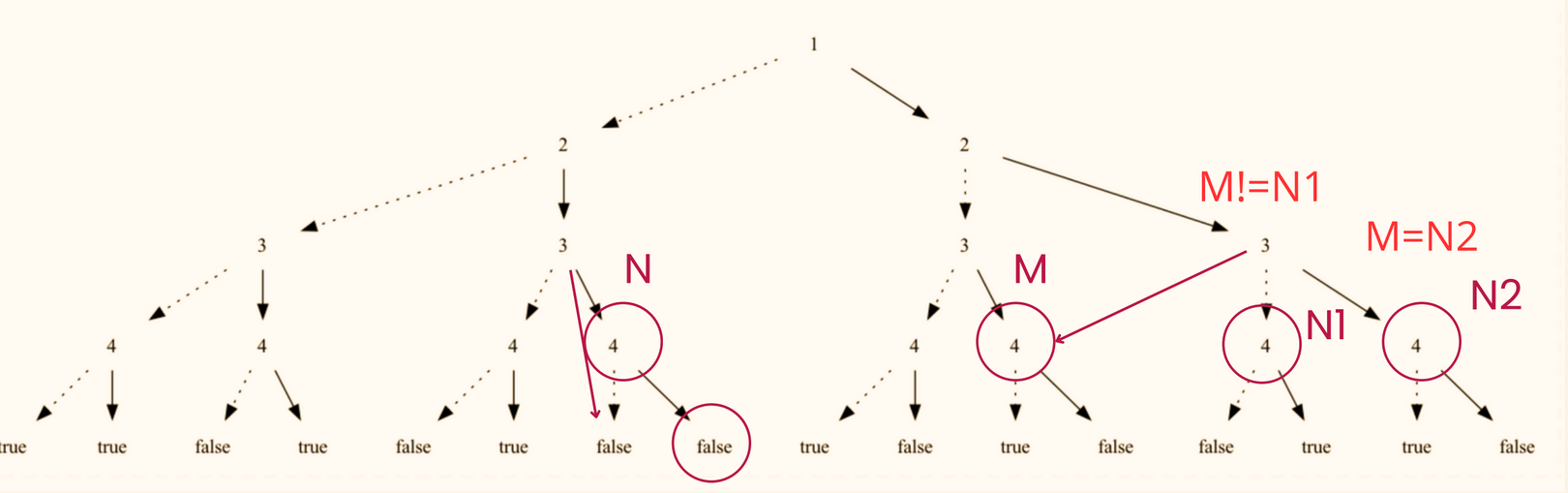
```
(* composition : bool list -> grand_entier *)
(* convertie bits en grand entier *)
let rec composition bits =
```

```
(* cons_arbre : bool list -> arbre *)
(* construit d'un arbre a partir d'une table de vérité *)
let cons_arbre table =
```

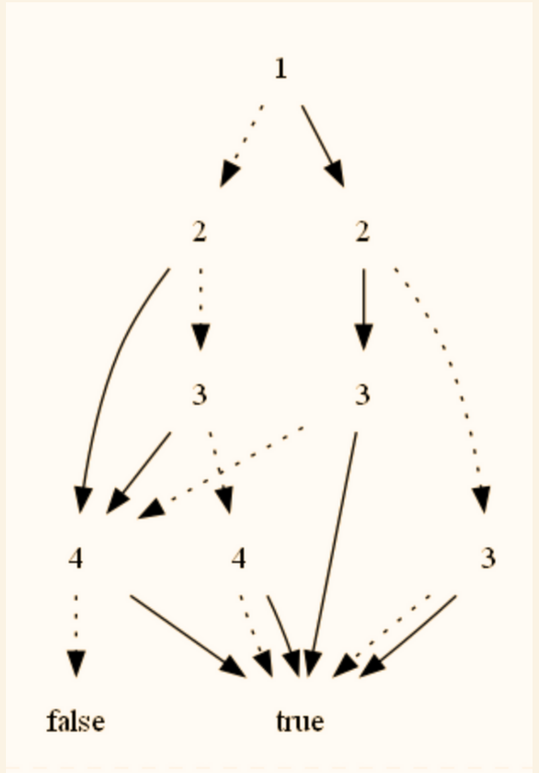
```
(* decomposition : grand_entier -> bool list *)
(*  convertie un grand entier en bits *)
let rec decomposition x =
```

```
(* liste_feuilles : arbre -> bool list *)
(* convertie un arbre en table de vérité *)
let rec liste_feuilles arbre =
```

COMPRESSION DE L'ARBRE DE DÉCISION ET ZDD



compression_par_liste

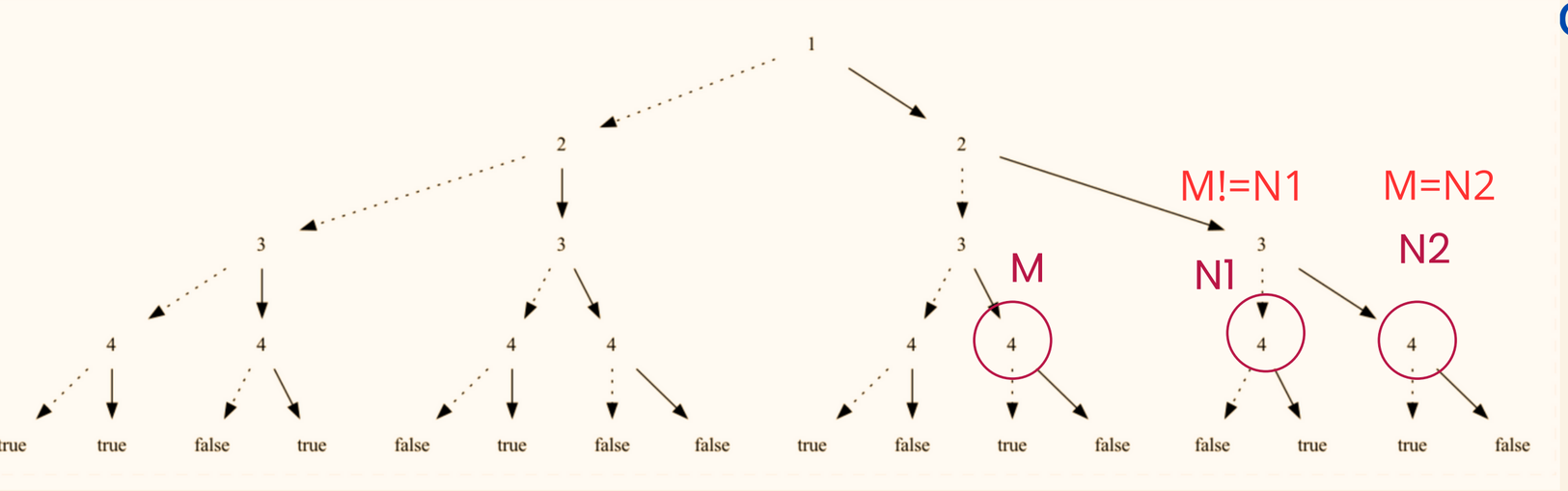



```
type liste_deja_vus = (grand_entier * arbre_decision) list
```

regle_M et regle_Z : renvoie un nouveau arbre qui applique la regle

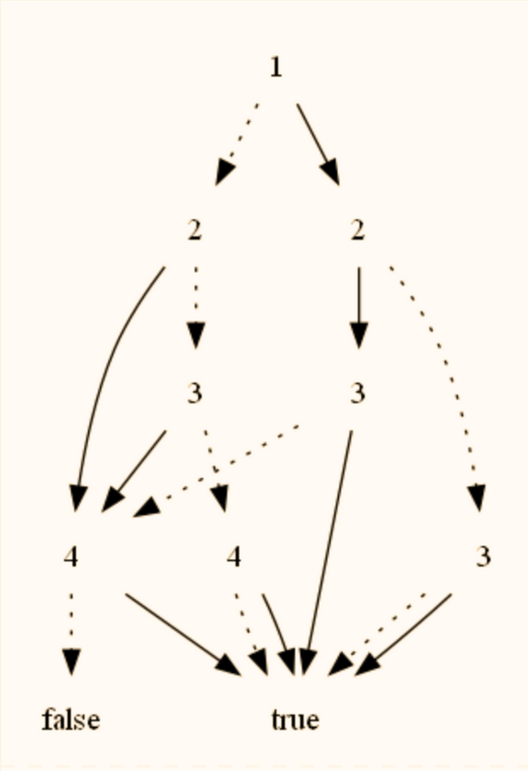
COMPRESSION DE L'ARBRE DE DÉCISION ET ZDD

compression_par_arbre

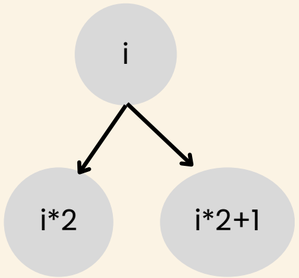


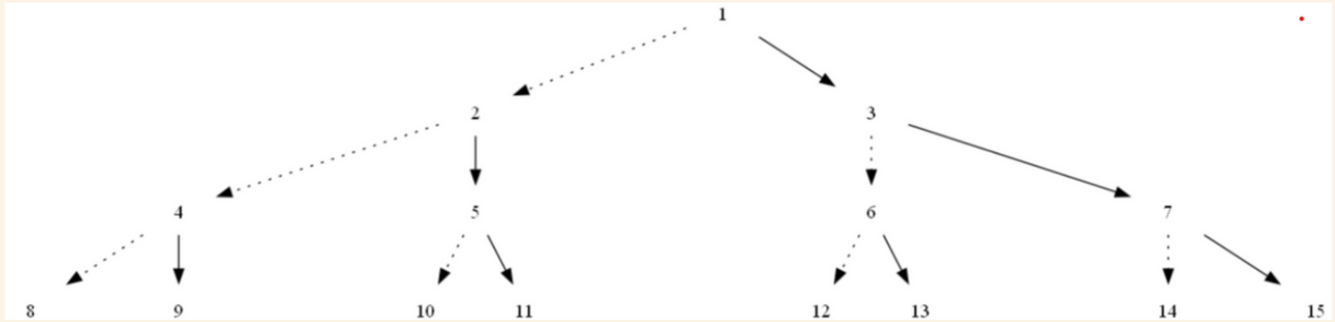

```
type arbre_deja_vus =
  | Leaf
  | Node of  arbre_decision option * arbre_deja_vus  * arbre_deja_vus
;;
```

regle_M_bis et regle_Z : renvoie un nouveau arbre qui applique la regle



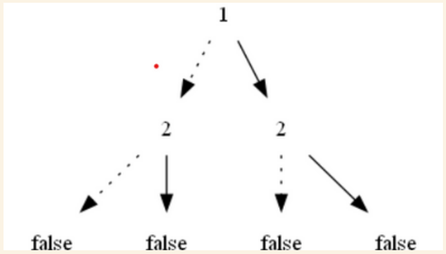
GRAPHE EN DOT



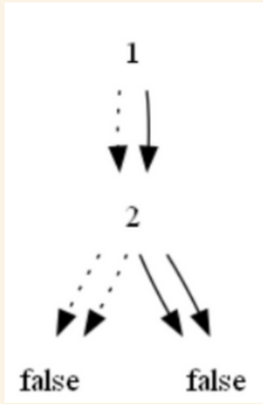



```
(* tableau : arbre_decision -> (arbre_decision * int) array *)
(* un tableau qui contient des couples (pointeur vers un arbre, id) de tout les noeuds de l'arbre*)
let tableau arbre =
```

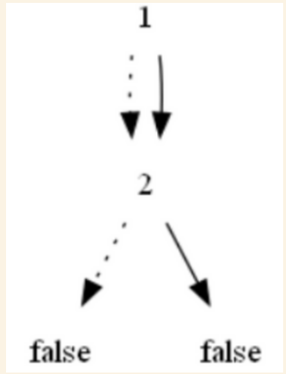
```
(* mise_a_jour_tableau : (arbre_decision * int) array -> arbre_decision -> int -> unit *)
(* t est un array composé de de couple de arbre et un id *)
(* index est l'index du tableau ou l'on souhaite mettre a jour *)
let mise_a_jour_tableau t nouveau_arbre index =
```



utilisation de tableau



remove_duplicate_lines



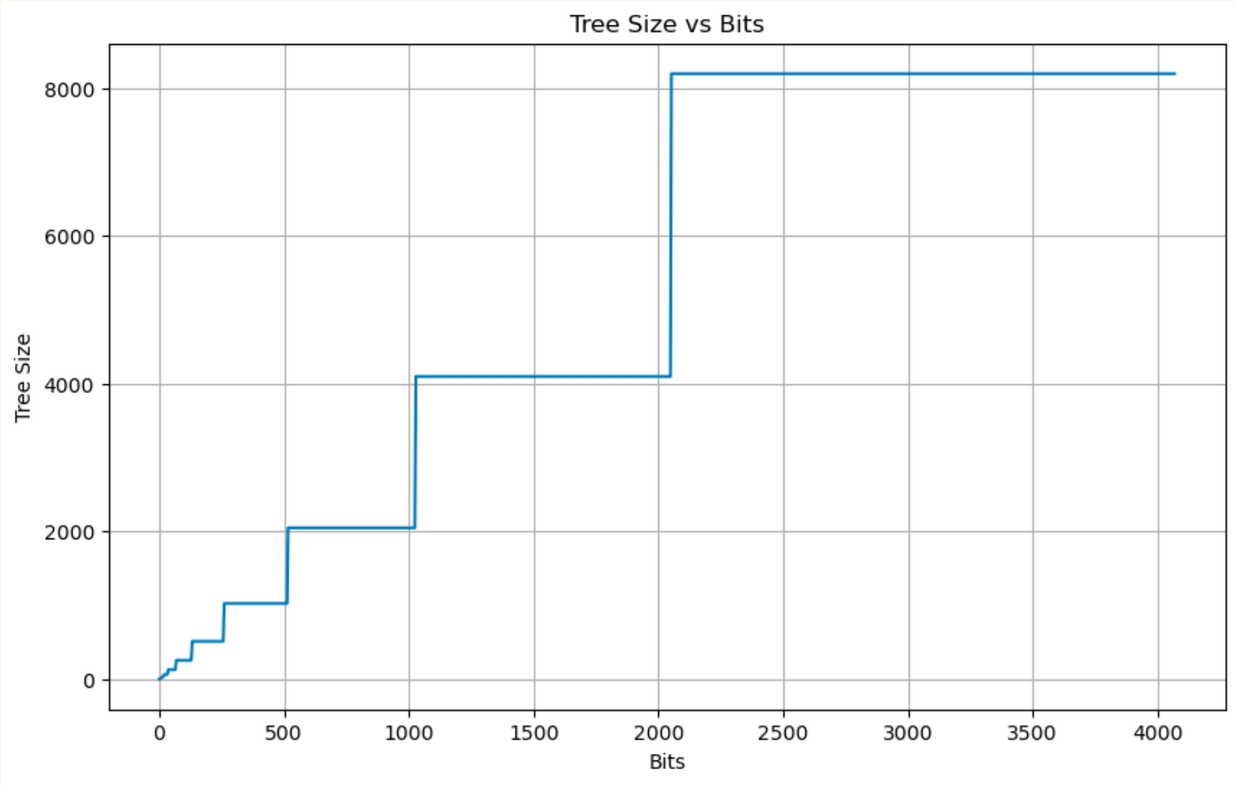
pointeur non pris en compte

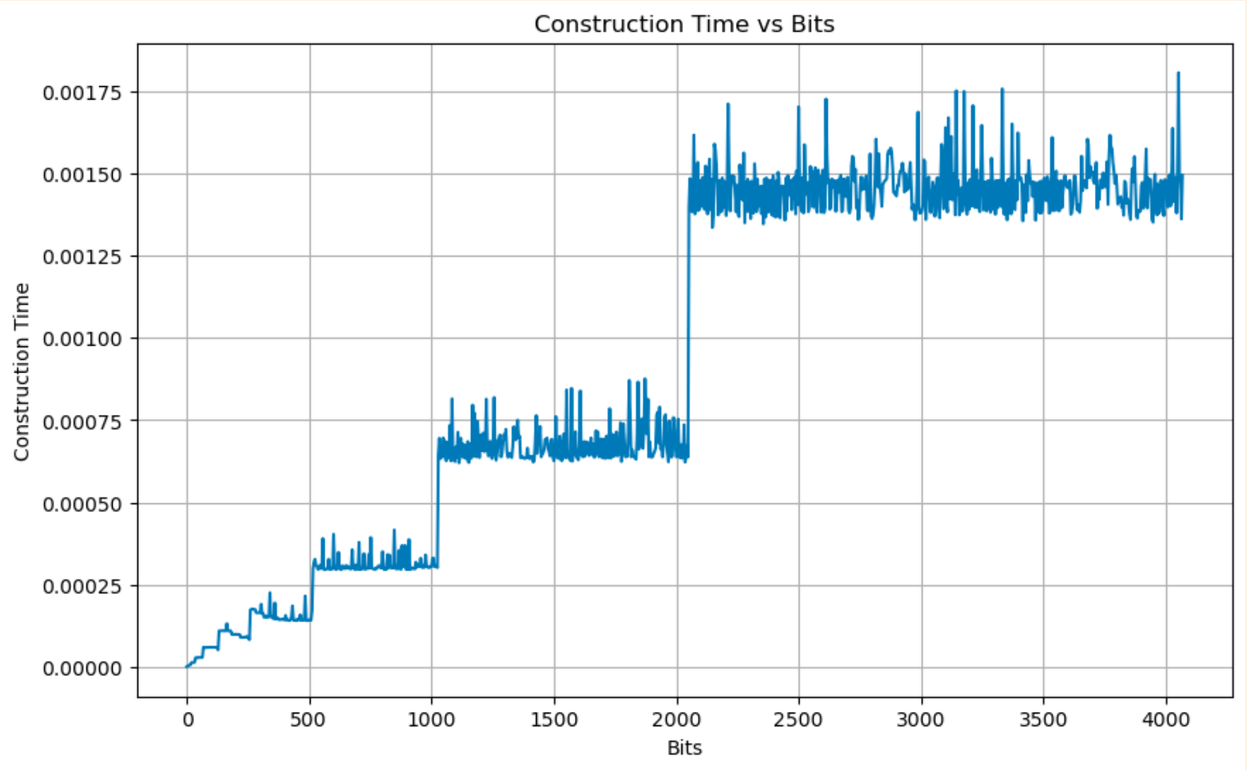
redondance de flèche

résultat attendu

ANALYSE

Taille et Temps de Construction

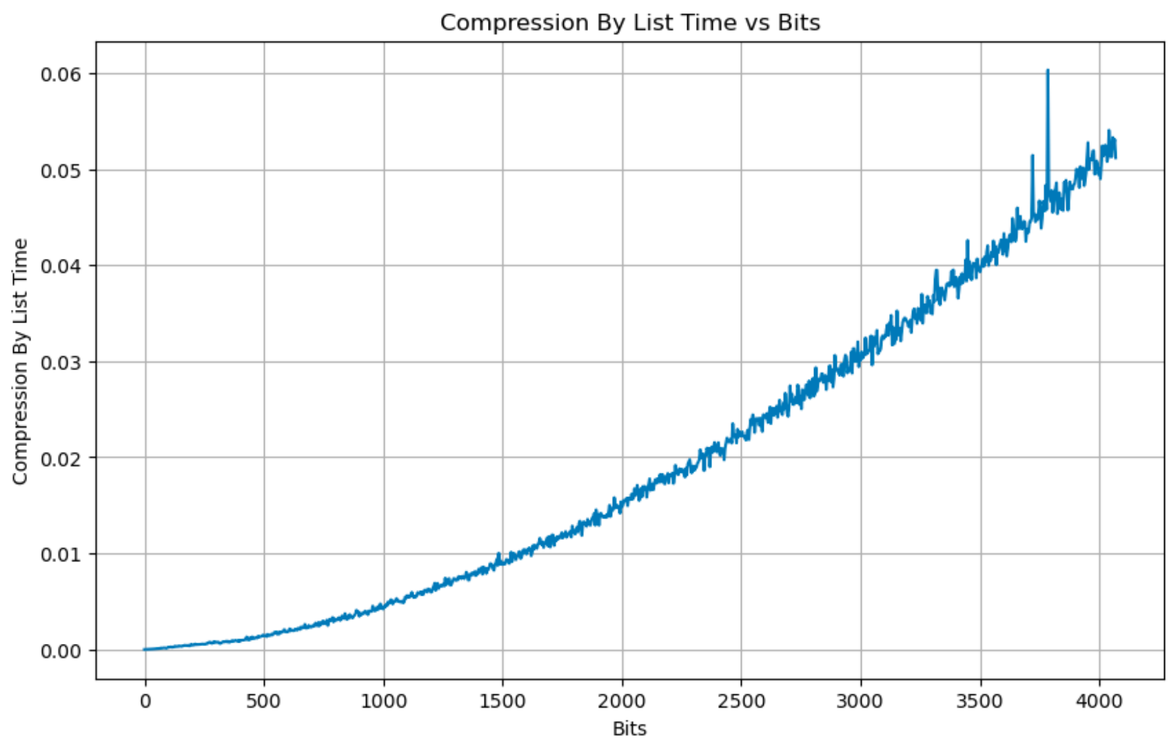




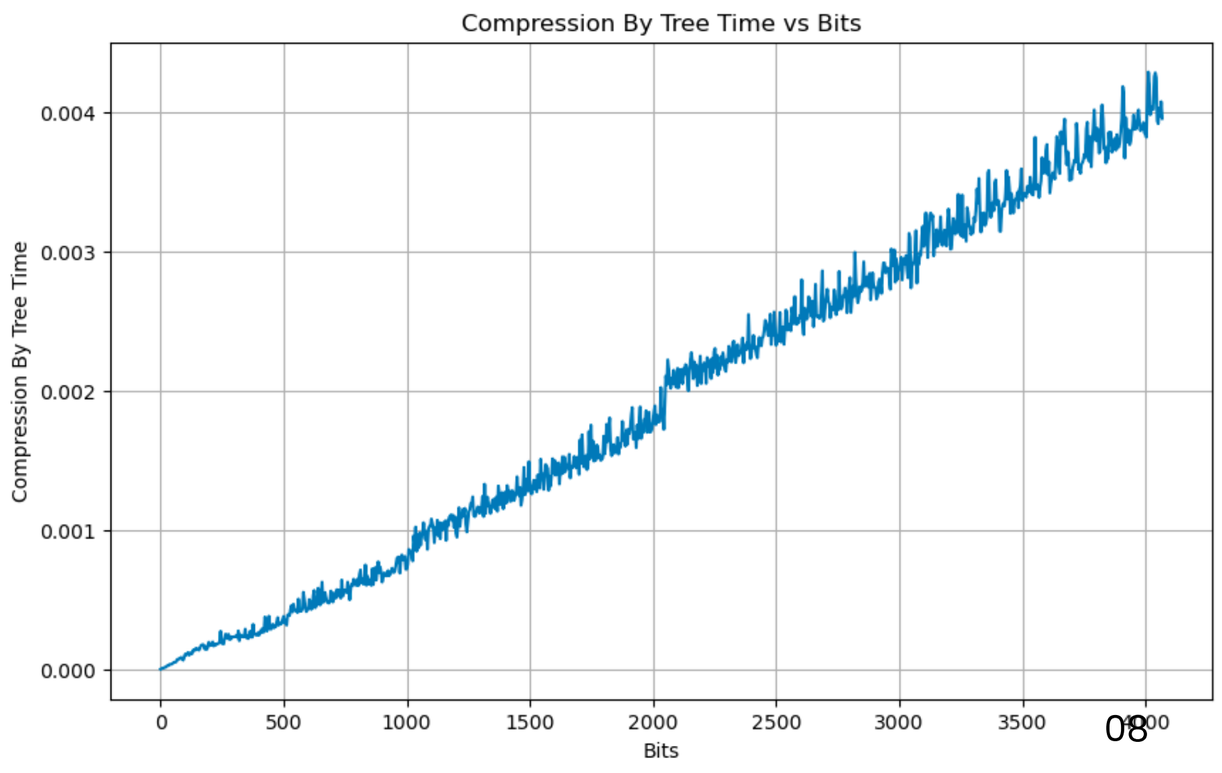
ANALYSE

Temps de compression

Complexité O(n2+nm)



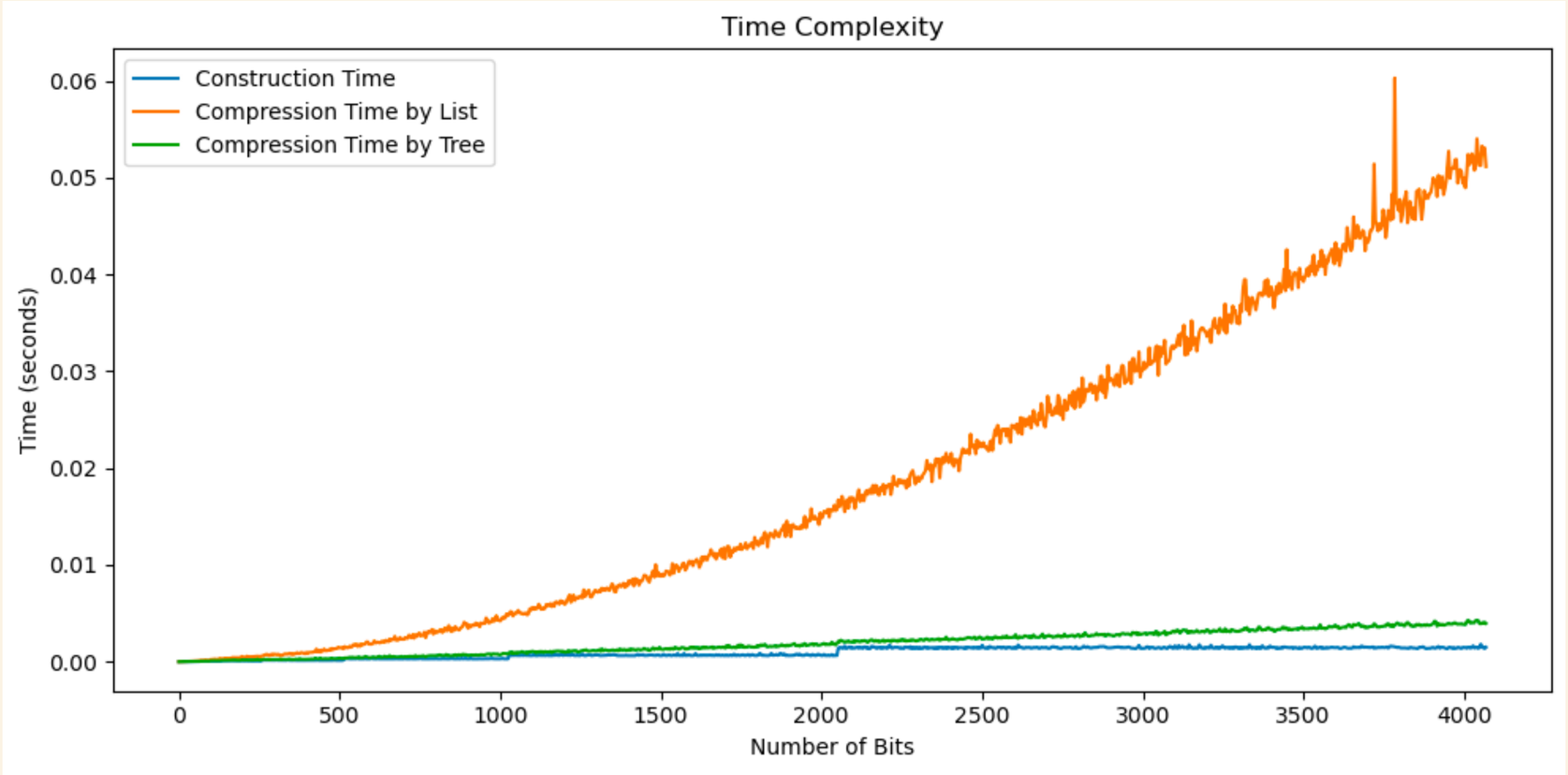
Complexité O(n log n)



ANALYSE

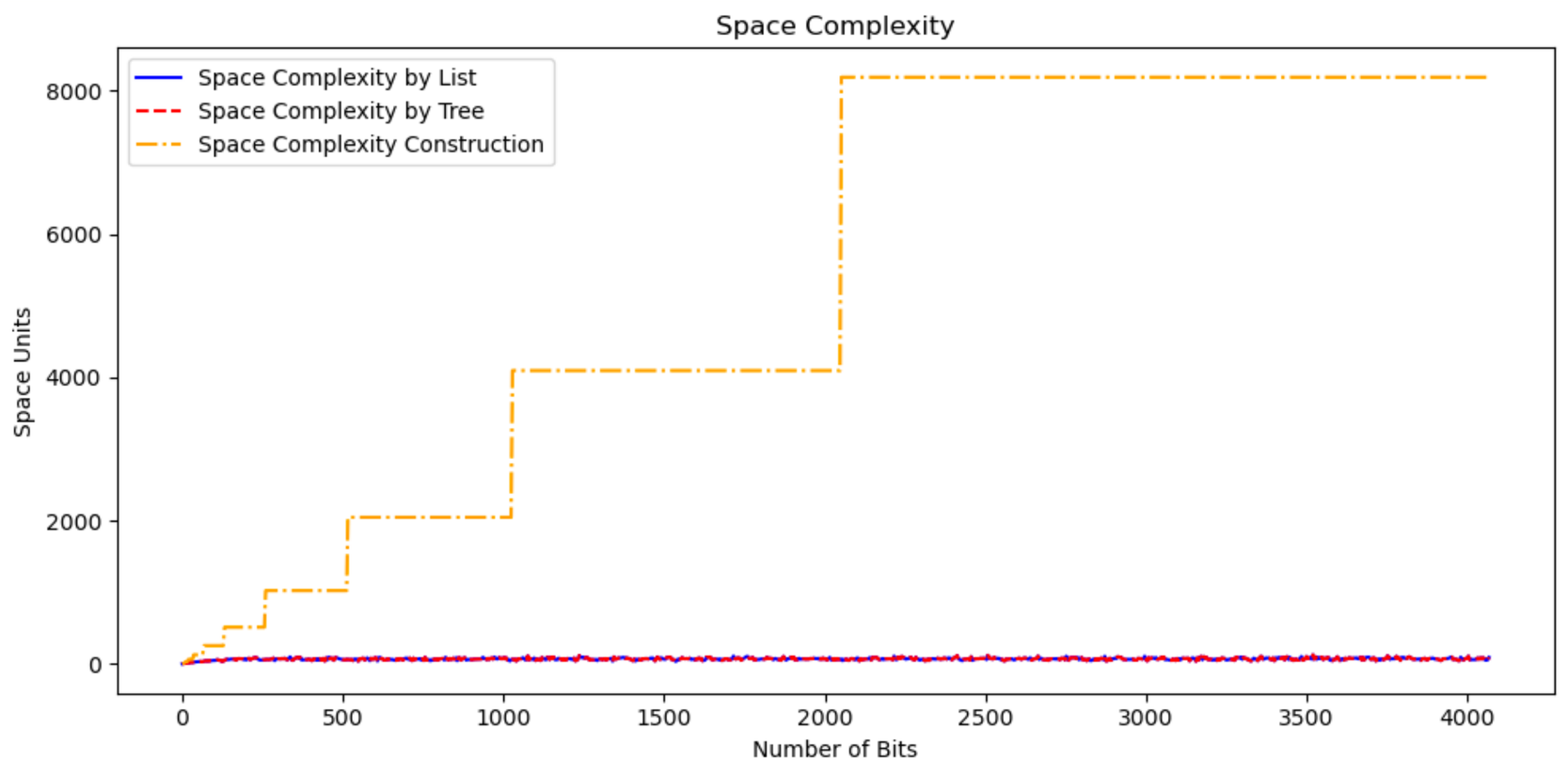
Temps de compression et Construction

Efficacité: par arbre > par liste



ANALYSE

Espace mémoire de compression



MERCI DE VOTRE ATTENTION

Ariane ZHANG Xue YANG