Compression d'arbres de recherche

C. Bardoux, M. Hernouf, A. Rozovyk

Devoir de Programmation

26/11/2019

Plan de l'exposé

Introduction

Synthèse de données

- Extraction aléatoire.
- Algorithme de shuffle de *Fisher-Yates*
- Construction de l'ABR

FIGURE – Arbre binaire de recherche

Introduction

Synthèse de données

- Extraction aléatoire.
- Algorithme de shuffle de *Fisher-Yates*
- Construction de l'ABR

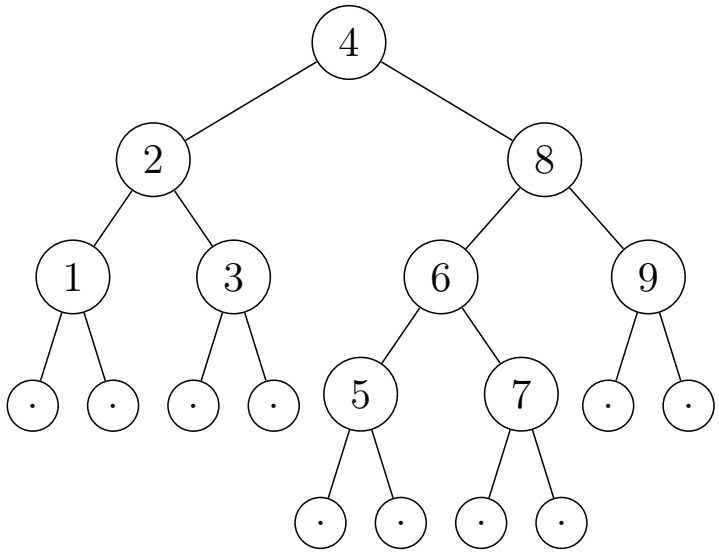
FIGURE – Arbre binaire de recherche

Introduction

Synthèse de données

- Extraction aléatoire.
- Algorithme de shuffle de *Fisher-Yates*
- Construction de l'ABR

FIGURE – Arbre binaire de recherche



Introduction

Synthèse de données

- Chemin le plus long en moyenne :

$$\frac{1}{\beta} \log n \approx 4.311 \log n,$$

où β est un nombre $0 < \beta < 1$ qui satisfait l'équation

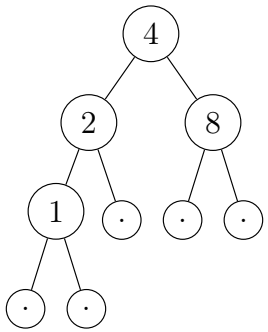
$$2\beta e^{1-\beta} = 1.$$

Compression d'ABR

"Carcasse"

- Les premiers noeuds uniques servent de carcasse :

FIGURE – Arbre initial



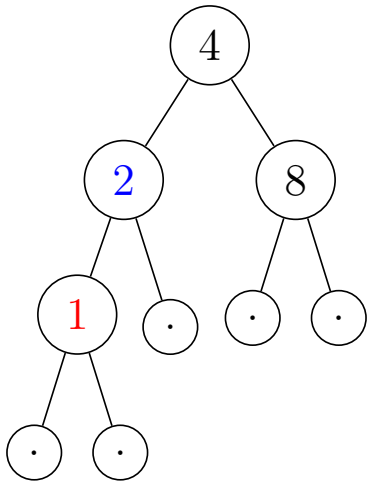

```
4 ,"(((())())(((())())())()"
2 ,"((())())()"
1 ,"()"
8 ,"(((())())())()"
```

Compression d'ABR

Les références

- Liste de (cle,référence) :

FIGURE – Arbre initial

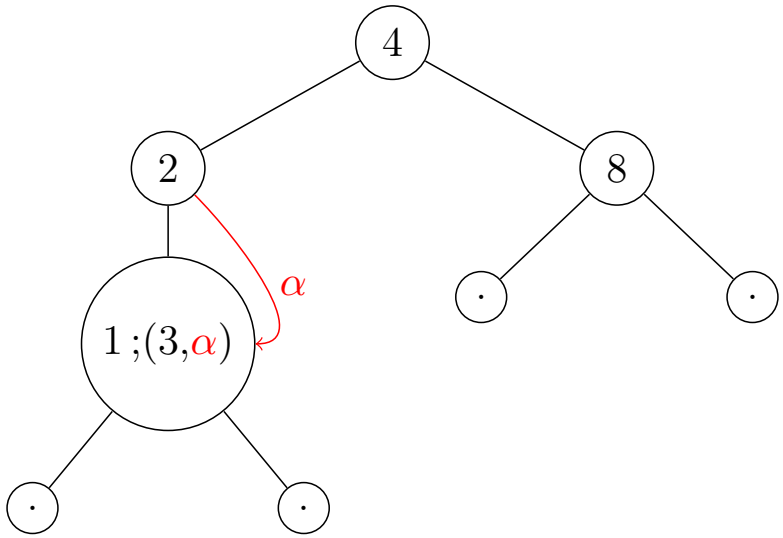


[(3, 1); (6, 2); (5, 1); (7, 1); (9, 1)]

Compression d'ABR

Les références

- Liste de (cle,référence) :

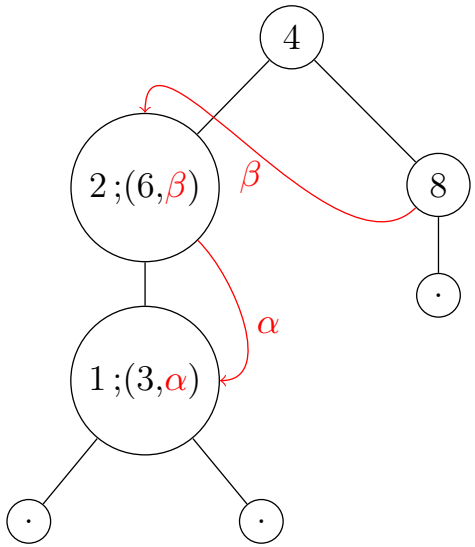


$$[(6\,,\ 2)\,;\ \ (5\,,\ 1)\,;\ \ (7\,,\ 1)\,;\ \ (9\,,\ 1)]$$

Compression d'ABR

Les références

- Liste de (cle,référence) :

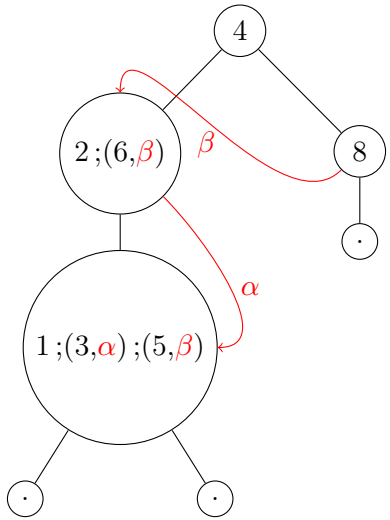


[(5 , 1) ; (7 , 1) ; (9 , 1)]

Compression d'ABR

Les références

- Liste de (cle,référence) :

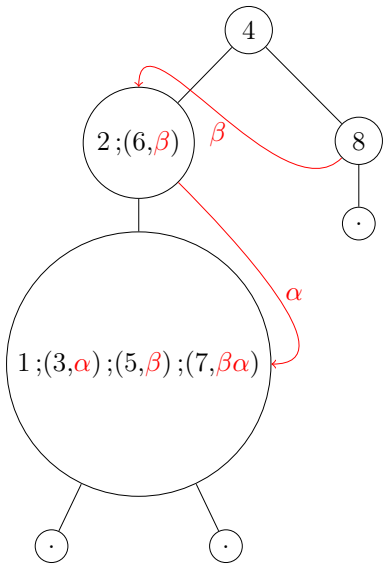


[(7 , 1) ; (9 , 1)]

Compression d'ABR

Les références

- Liste de (cle,référence) :

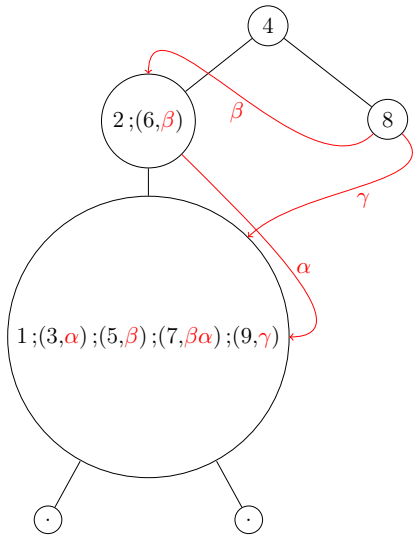


[(9 , 1)]

Comression d'ABR

Les références

- Liste de (cle,référence) :



[ ]

Compression d'ABR

Amélioration de stockage de noeuds internes

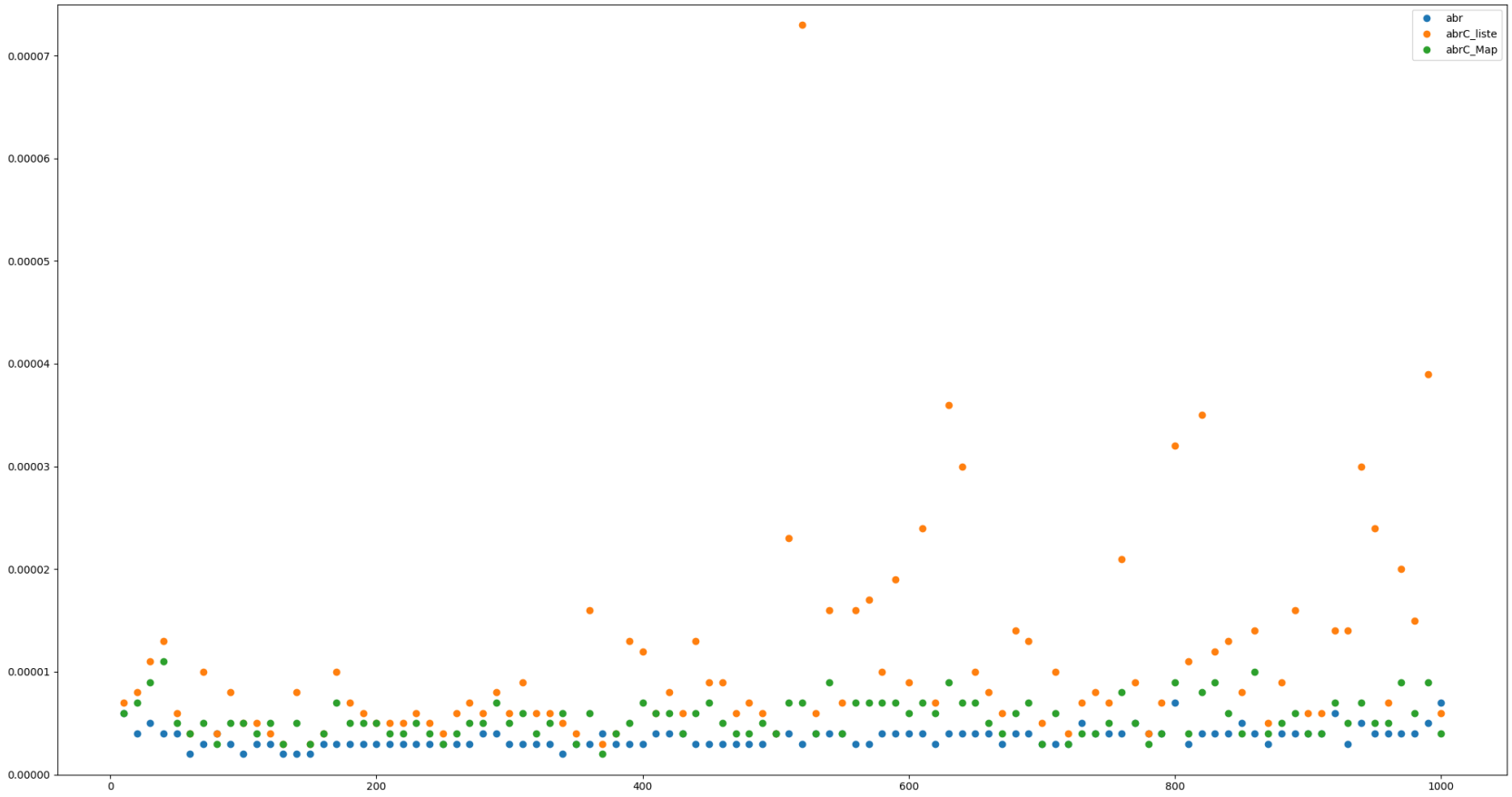
- Implémention interne

```
NodeABRC ([(3, [])],
   ({contents =
      NodeABRC ([(2, [])],
       ({contents =
          NodeABRC ([(1, []); (4, [1]); (7, [2]); (10, [3])],
           ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
        0),
       ({contents = EmptyABRC}, 0))},
    0)
```

- Recherche : complexité au pire cas O(n)
- Amélioration grâce au module Map

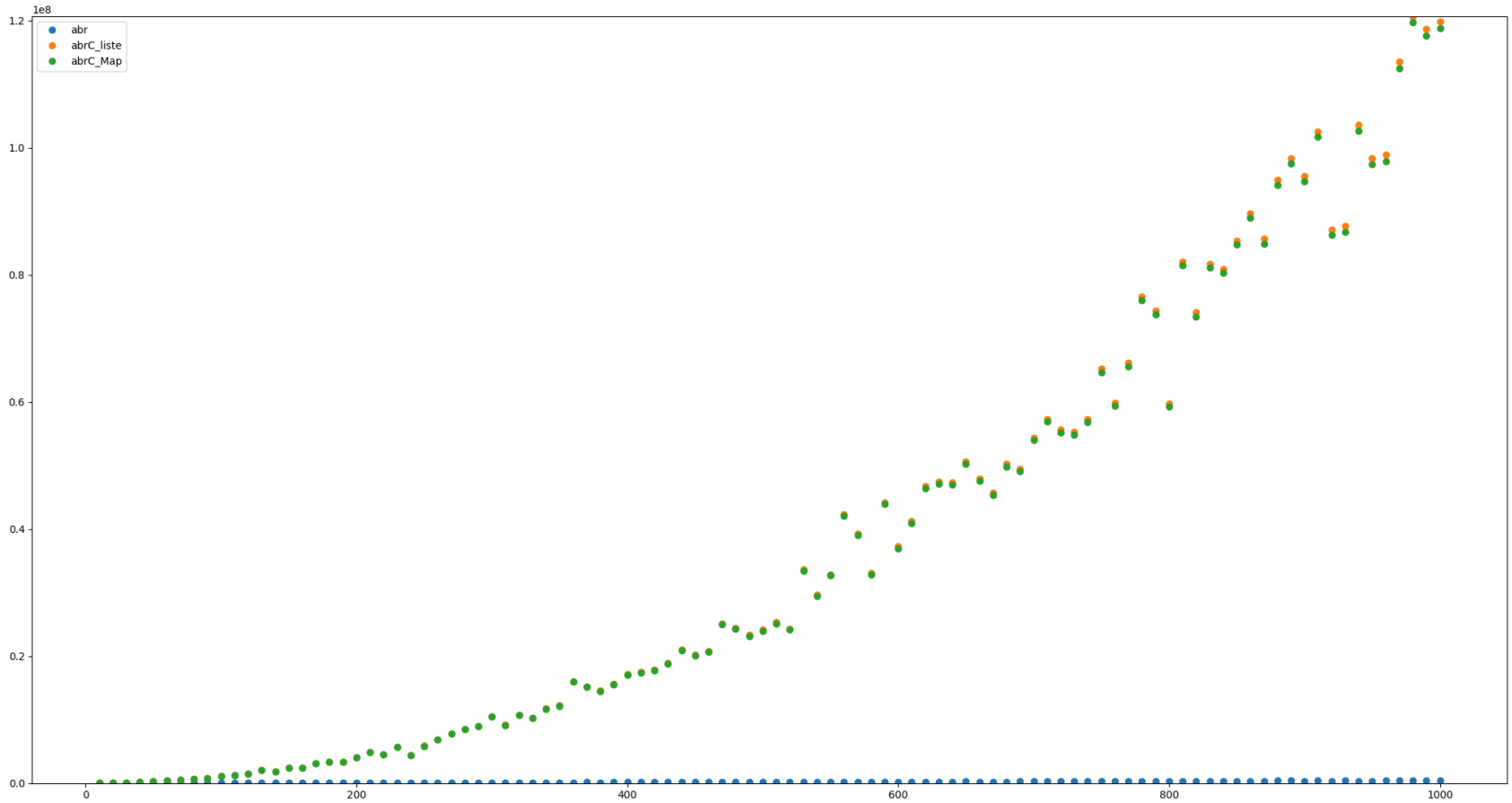
Expérimentations

Étude de complexité en temps



Expérimentations

Étude de complexité en espace



Expérimentations

Investigations

```
                    NodeABRC(x, (refL, etL), (refR, etR))))
              else
                (NodeABRC (insert_ordered_list (v,etiquettesVisitees) x, (refL,etL),(refR,etR)))
    in aux abrc [];;

(*Insertion d'un ensemble des valeurs dans ABRC (avec la creation des etiquettes)*)
let liste_to_abrc_etiq l abrc =
  let rec aux l abrc=
    match l with
      | [] -> abrc
      | x::q ->aux q (insert_abrc_etiq (fst x) (snd x) abrc)
  in aux l abrc

(*Compression d'un ABR*)
let compresse_abr_listes abr =
  let s = ref (Gc.allocated_bytes()) in
    relancer_gen();
    (*Avec l'ABR du ennoncé:
      listeConstr1 = [(4, "((())(())((()))())()"); (2, "(())()"); (1, "()"); (8, "((()))(())()")]
      listeConstr2 = [(3, 1); (6, 2); (5, 1); (7, 1); (9, 1)]*)
    let (listeConstr1, listeConstr2) = listes_construction abr
    in
      (*Avec l'ABR du ennoncé:       (x [etiquetes]) - noeud
        abrc_init=
        (4 [])
        (2 [])        (8 [])
        (1 [])                          *)
      s:=Gc.allocated_bytes() -. !s; print_endline ("listes_constructions prend " ^(string_of_float
 !s)); s:=Gc.allocated_bytes();
      let abrc_init = liste_to_abrc (List.map fst listeConstr1)
      in
        (*Avec l'ABR du ennoncé:
          liste_refernces = [(3, reference de (1 [])); (6, reference de (2 [])); (5, reference de (
1 []));
          (7, reference de (1 [])); (9, reference de (1 []))]*)
        s:=Gc.allocated_bytes() -. !s; print_endline ("liste_to_abrc prend " ^(string_of_float !s))
; s:=Gc.allocated_bytes();
        let list_references = List.combine (List.map fst listeConstr2) (liste_refs (List.map snd li
steConstr2) abrc_init) in
          (*Avec l'ABR du ennoncé:
            abrc_fin=
            (4 [])
            (2 []; 6 [2])                       (8 [])
            (1 []; 3 [1]; 5 [2]; 7 [2,1]; 9 [3])          etiq1           etiq2        etiq3
          *)
          s:=Gc.allocated_bytes() -. !s; print_endline ("List.combine prend " ^(string_of_float !s)
); s:=Gc.allocated_bytes();
          let abrc_fin =  liste_to_abrc_etiq list_references abrc_init in
            s:=Gc.allocated_bytes() -. !s; print_endline ("liste_to_abrc_etiq prend " ^(string_of_f
loat !s));
```

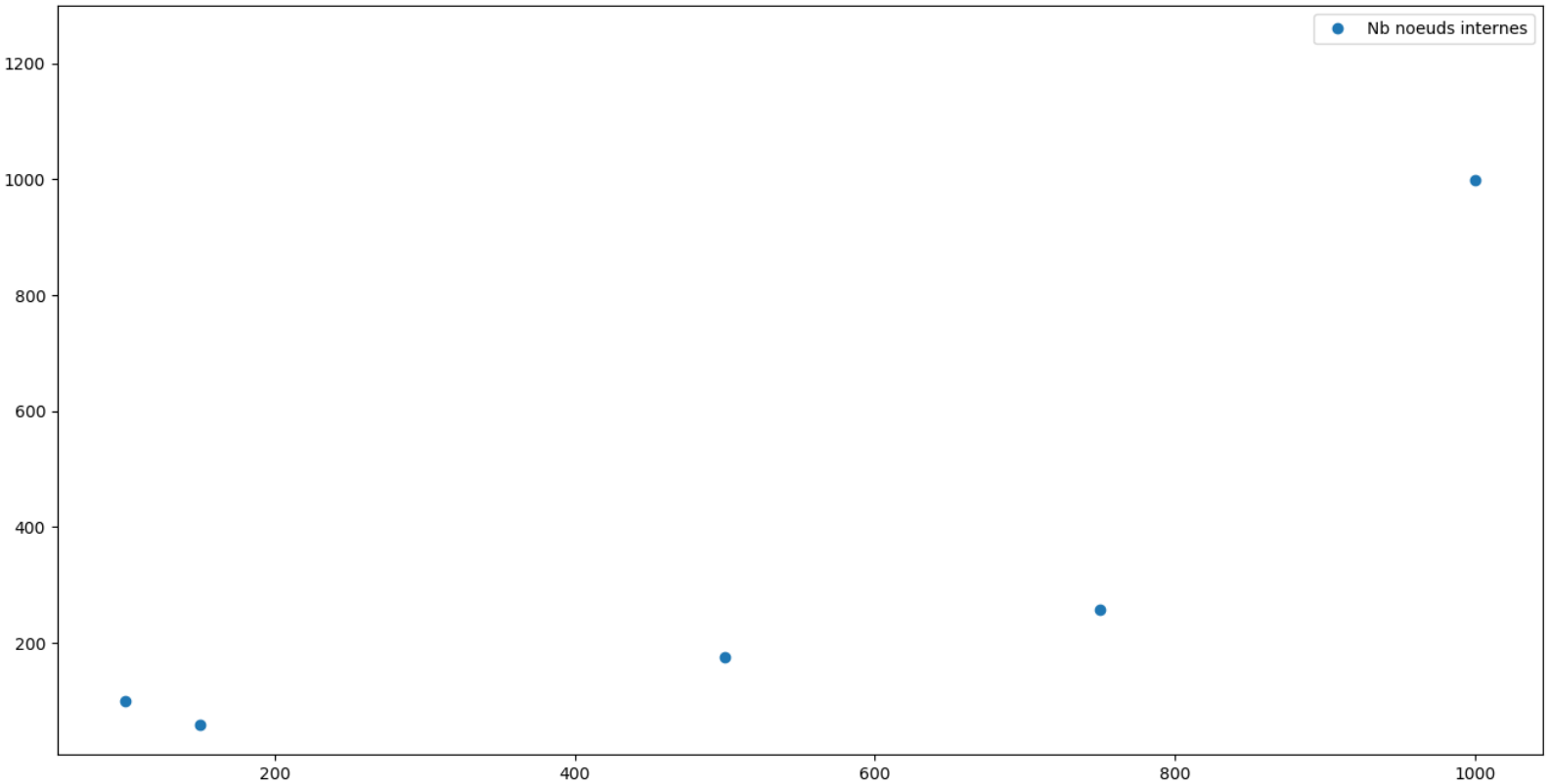
```
          NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
           ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
        1))},
     2),
    ({contents =
       NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
        ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
     3))},
  0))

# (space (fun () -> compresse_abr_listes arbclassic))
Execution space: 12536.000000 bytes
listes_constructions prend 6168.
liste_to_abrc prend 1304.
List.combine prend 808.
liste_to_abrc_etiq prend 3024.

- : abrc_listes =
NodeABRC ([(4, [])],
 ({contents =
    NodeABRC ([(2, []); (6, [2])],
     ({contents =
        NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
         ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
      0),
     ({contents =
        NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
         ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
      1))},
  0),
 ({contents =
    NodeABRC ([(8, [])],
     ({contents =
        NodeABRC ([(2, []); (6, [2])],
         ({contents =
            NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
             ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
          0),
         ({contents =
            NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
             ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
          1))},
      2),
     ({contents =
        NodeABRC ([(1, []); (3, [1]); (5, [2]); (7, [2; 1]); (9, [3])],
         ({contents = EmptyABRC}, 0), ({contents = EmptyABRC}, 0))},
      3))},
  0))
#
```

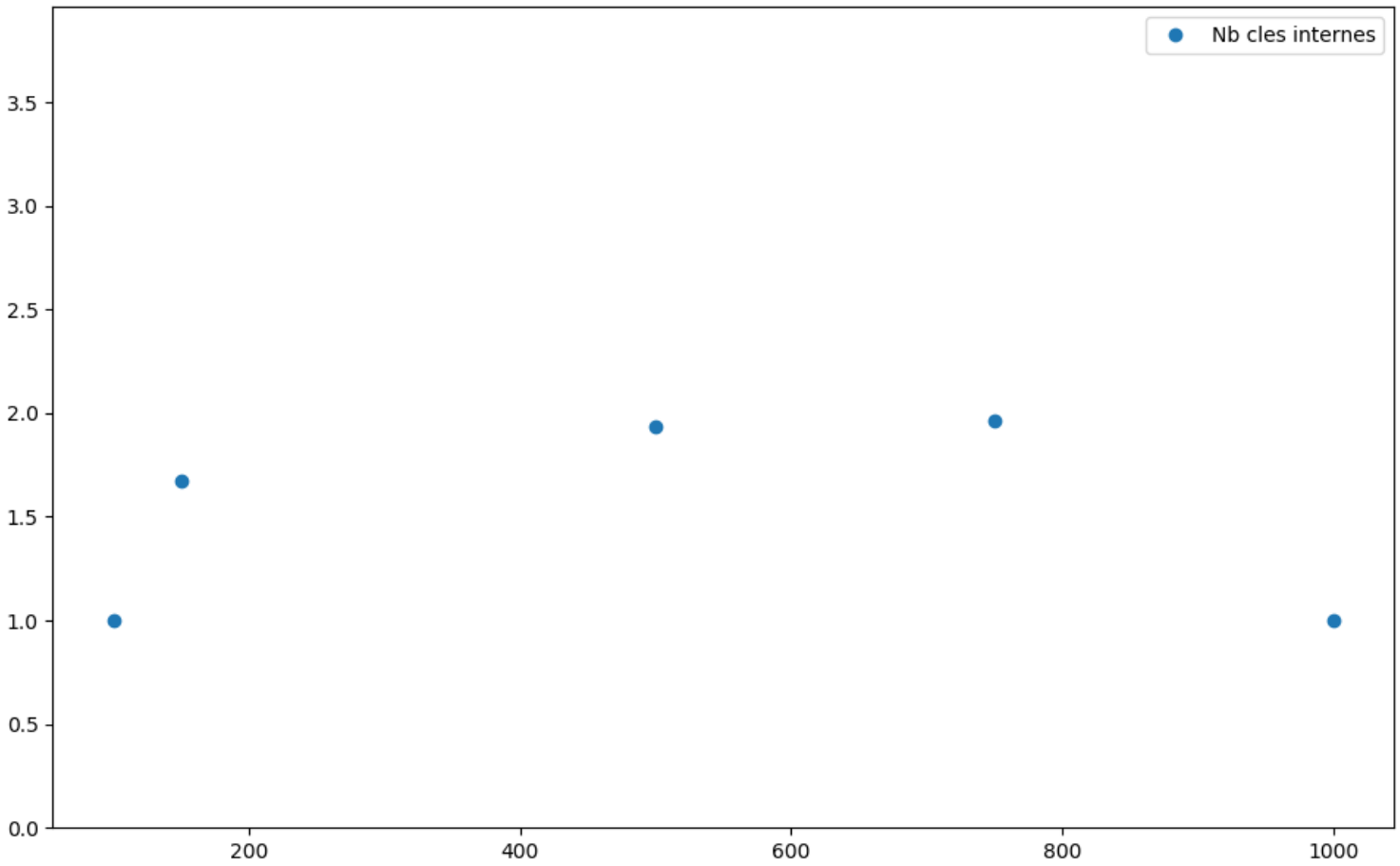
Expérimentations

Nombre de noeuds internes



Expérimentations

Nombre moyen de clés dans les noeuds internes



Conclusion

Merci !