SCIENCES SORBONNE UNIVERSITÉ

Master STL
MU4IN500 - ALGAV

Devoir de Programmation

Langage C

Alex XU & Stéphanie HUANG

2023-2024

Clef sur 128 bits

```c
typedef struct clef128 Clef128;
struct clef128{
    unsigned int b32_1; //poids faibles
    unsigned int b32_2;
    unsigned int b32_3;
    unsigned int b32_4; //poids forts
    char* clef_hexa; // cles en hexa
};
```

Tas Min Tableau - Structure

Tas Min Tableau - Ajouts Itératifs > Construction

Opération d'ajout :

- insertion dans un tableau O(1)
- remonté O(logn), n le nb élément du tas

=> O(logn)

```
// On remonte la clef à sa place
while (i > 0 && inf(&clef, &tas->tab[(i-1)/2])) { // Tant que la clef
    echanger(&tas->tab[i], &tas->tab[(i-1)/2]); // On remonte la clef
    i = (i-1)/2;
}
```

Ajouts Itératifs :

- n opération d'ajout

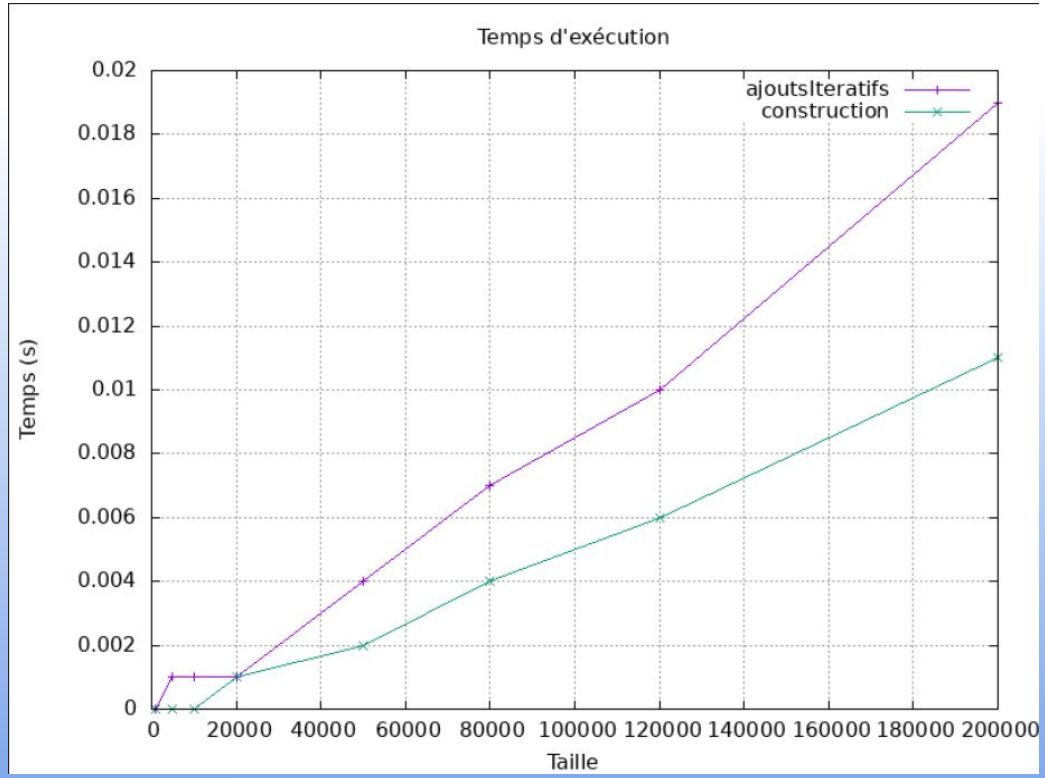
=>O(nlogn)

Construction :

- n insertion dans un tableau O(n)
- rééquilibrage du tas en O(logn)

=>O(n)

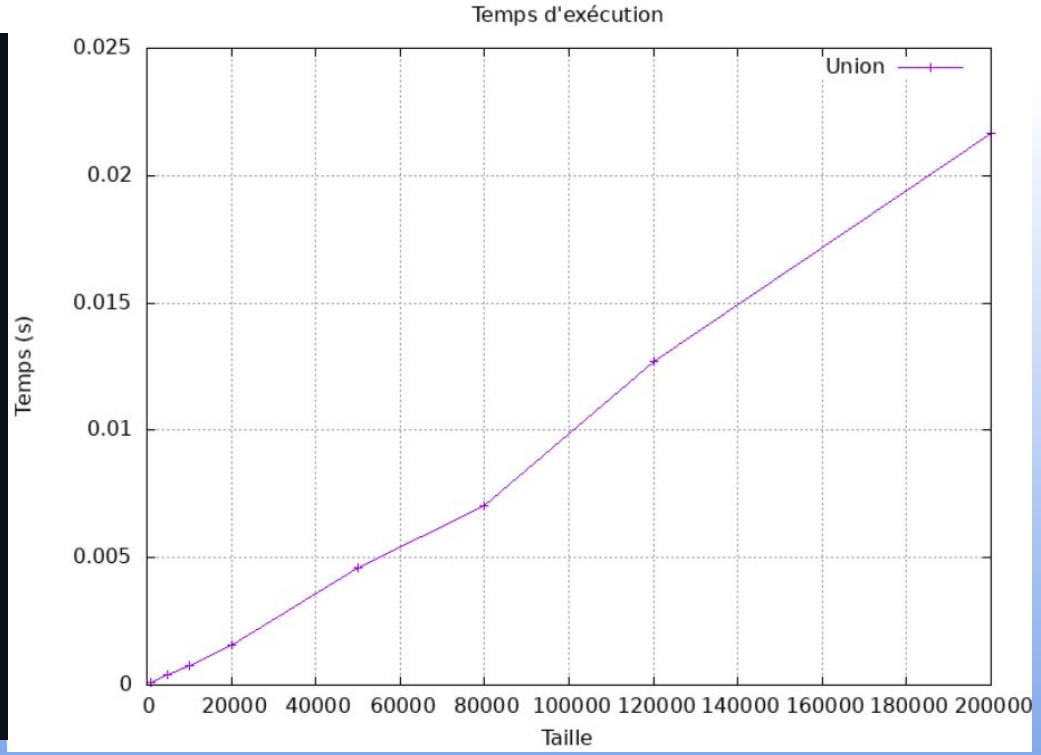
```
if (gauche < tas->taille && inf(&tas->tab[gauche], &tas->tab[min])) {
    min = gauche;
}
if (droite < tas->taille && inf(&tas->tab[droite], &tas->tab[min])) {
    min = droite;
}
// Si le minimum a changé, on échange et on rééquilibre le sous-arbre
if (min != i) {
    echanger(&tas->tab[i], &tas->tab[min]);
    reequilibrerTas(tas, min);
}
```



Temps d'exécution en fonction du nombre de clefs

Tas Min Tableau - Union de même taille

```c
TasTableau * UnionTasTableau(TasTableau *tas1, TasTableau *tas2) {
    if(tas1->taille == 0) return tas2;
    if(tas2->taille == 0) return tas1;

    int tailleTotale = tas1->taille + tas2->taille;
    TasTableau * tasUnion = initTas(tailleTotale);

    // Copie des éléments du tas 1 et 2 dans le nouveau tas
    for (int i = 0; i < tas1->taille; i++) {  // O(n)
        tasUnion->tab[i] = tas1->tab[i];
    }
    for (int i = 0; i < tas2->taille; i++) {  // O(m)
        tasUnion->tab[tas1->taille + i] = tas2->tab[i];
    }
    tasUnion->taille = tailleTotale;

    // rééquilibrage du nouveau tas
    for (int i = (tasUnion->taille / 2) - 1; i >= 0; i--) {
        reequilibrerTas(tasUnion, i);  // O(logn)
    }
    return tasUnion;
}
```



Temps d'exécution en fonction du nombre de clefs

Tas Min Arbre - Structure

```c
typedef struct tasArbre TasArbre;
struct tasArbre {
    Clef128* clef;
    struct tasArbre * fg;
    struct tasArbre * fd;
    struct tasArbre * parent; // parent du noeud
    int hauteur; // hauteur de l'arbre
    int noeud; // nombre de noeud present dans l'arbre
};

typedef struct Element Element;
struct Element {
    TasArbre* noeud;
    struct Element* suiv;
    struct Element* pre;
};

typedef struct Liste Liste;
struct Liste
{
    Element* tete;
    Element* queue;
};
```

Tas Min Arbre - Ajouts Itératifs > Construction

Opération d'ajout :

- insertion O(logn) :
  - position dans l'arbre en O(logn)
- remonté O(logn)

=> O(logn)

Ajouts Itératifs :

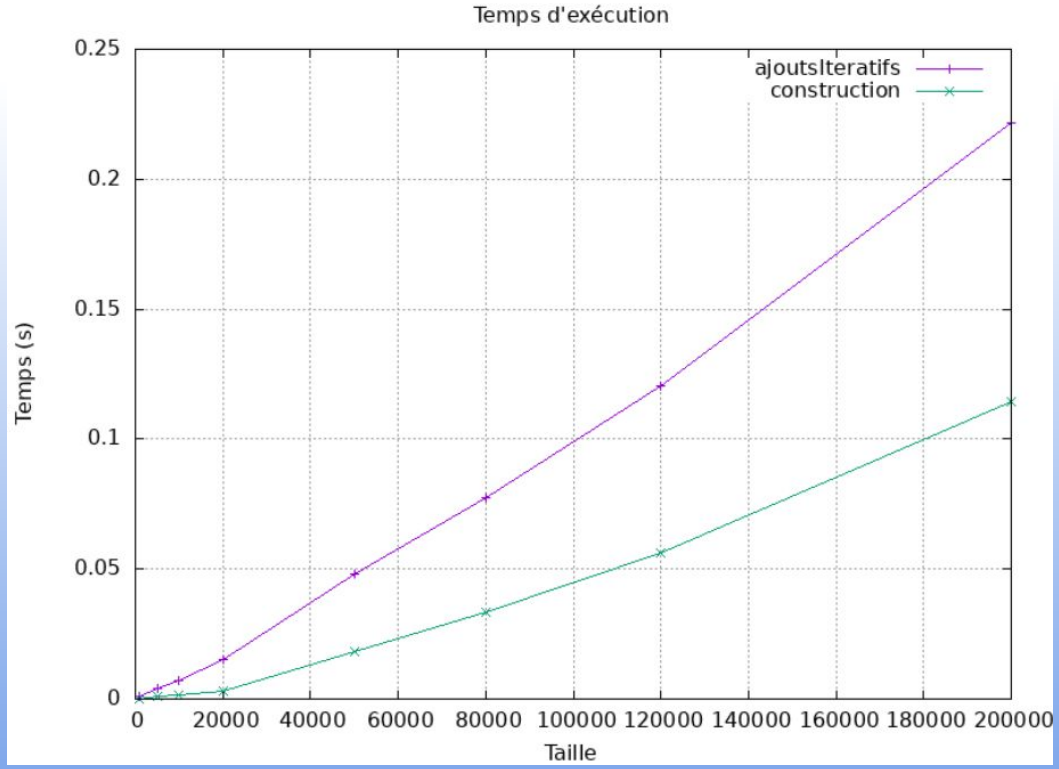
- n opérations d'ajouts => O(nlogn)

Construction :

- n insertions en 0(1) avec parcours en largeur O(n)
- rééquilibrage sur n/2 noeuds indépendants O(logn)

=> O(n)

```
for(int i = 1; i<len; i++)
{
    if(liste->tete->noeud->fg == NULL)
    {
        liste->tete->noeud->fg = creerNoeud(clefs[i]);
        liste->tete->noeud->fg->parent = liste->tete->noeud;
        liste->queue->suiv = ajoutListe(liste->tete->noeud->fg);
        liste->queue->suiv->pre = liste->queue;
        liste->queue = liste->queue->suiv;

    }else{
        if(liste->tete->noeud->fd == NULL)
        {
            liste->tete->noeud->fd = creerNoeud(clefs[i]);
            liste->tete->noeud->fd->parent = liste->tete->noeud;
            liste->queue->suiv = ajoutListe(liste->tete->noeud->fd);
            liste->queue->suiv->pre = liste->queue;
            liste->queue = liste->queue->suiv;
            liste->tete = liste->tete->suiv;
        }
    }
}
reequilibrageDescente(reequilibrage, len);
```

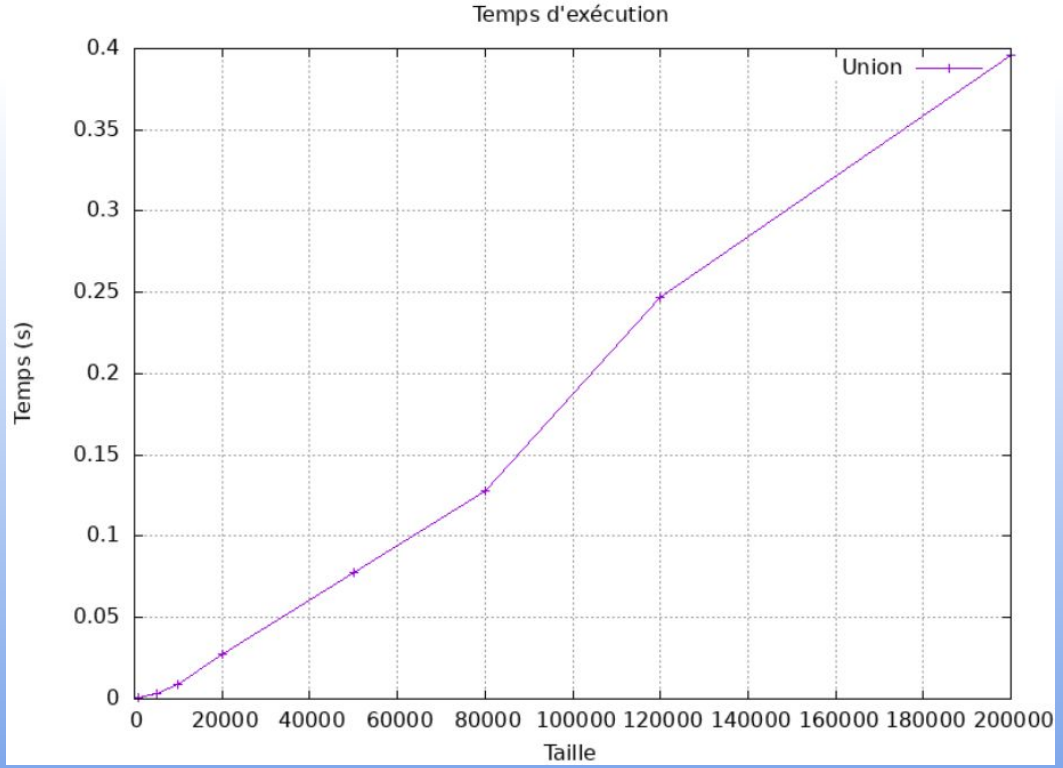


Temps d'exécution en fonction du nombre de clefs

Tas Min Arbre - Union de même taille

```c
while(tasAcopie->tete != NULL)
{
    if(tasAcopie->tete->noeud->fg)
    {
        tasAcopie->queue->suiv = ajoutListe(tasAcopie->tete->noeud->fg);
        tasAcopie->queue = tasAcopie->queue->suiv;
    }
    if(tasAcopie->tete->noeud->fd)
    {
        tasAcopie->queue->suiv = ajoutListe(tasAcopie->tete->noeud->fd);
        tasAcopie->queue = tasAcopie->queue->suiv;
    }
```

```c
if(newliste->tete->noeud->fg == NULL)
{
    len++;
    newliste->tete->noeud->fg = creerNoeud(tasAcopie->tete->noeud->clef);
    newliste->queue->suiv = ajoutListe(newliste->tete->noeud->fg);
    newliste->queue->suiv->pre = newliste->queue;
    newliste->queue = newliste->queue->suiv;
}else{
    if(newliste->tete->noeud->fd == NULL)
    {
        len++;
        newliste->tete->noeud->fd = creerNoeud(tasAcopie->tete->noeud->clef);
        newliste->queue->suiv = ajoutListe(newliste->tete->noeud->fd);
        newliste->queue->suiv->pre = newliste->queue;
        newliste->queue = newliste->queue->suiv;
        newliste->tete = newliste->tete->suiv;
    }
}
```



Temps d'exécution en fonction du nombre de clefs

File Binomiale - Structure

```c
typedef struct tournoiBinomial
{
    Clef128 clef;
    int degre;
    struct tournoiBinomial * fils;
    struct tournoiBinomial * frere; // frere droit
} TournoiBinomial;

typedef struct elementFile
{
    TournoiBinomial * tournoi;
    struct elementFile * suivant;
} ElementFile;

typedef struct fileBinomiale
{
    ElementFile * tete;
    ElementFile * queue;
} FileBinomiale;
```

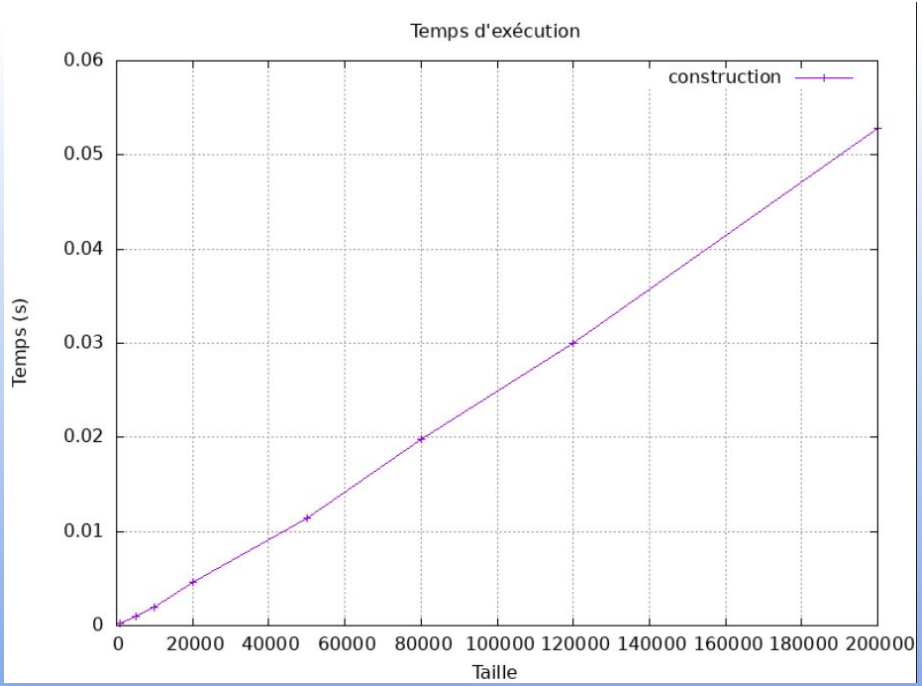
File Binomiale - Construction jusqu'à 200k clefs

Opération d'ajout :

- Insertion en tête O(1)
- Opération de rééquilibrage
  - parcours de la file O(k), k tournois dans la file
  - Union2Tid O(1)

=> O(k)

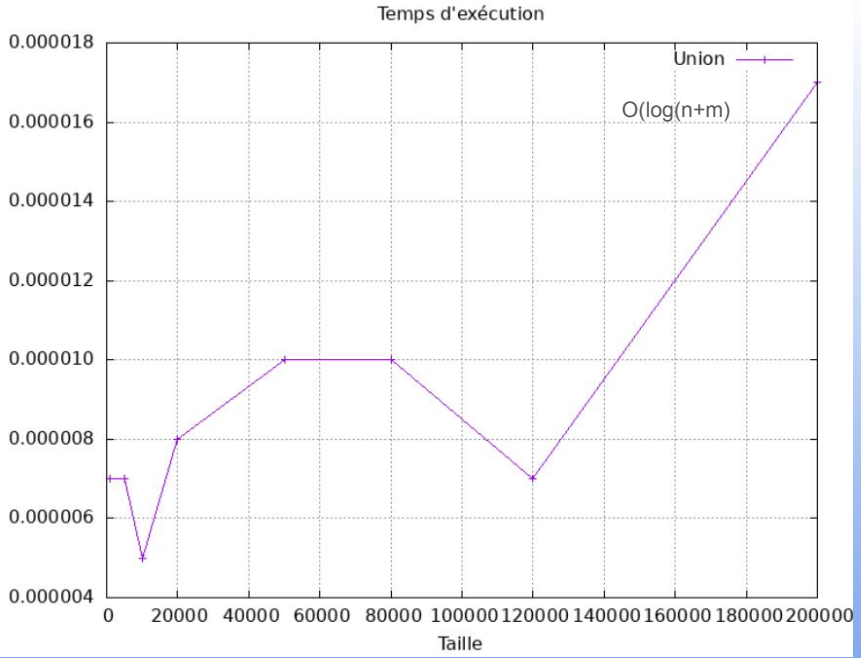
```
FileBinomiale * Construction (Clef128 ** clefs, int debut, int fin)
{
    FileBinomiale * F = NULL;
    for (int i = debut; i < fin; i++)
    {
        TournoiBinomial * T = Tournoi(*clefs[i]);
        F = AjoutMin(F, T);
    }
    return F;
}
```



File Binomiale - Union de même taille

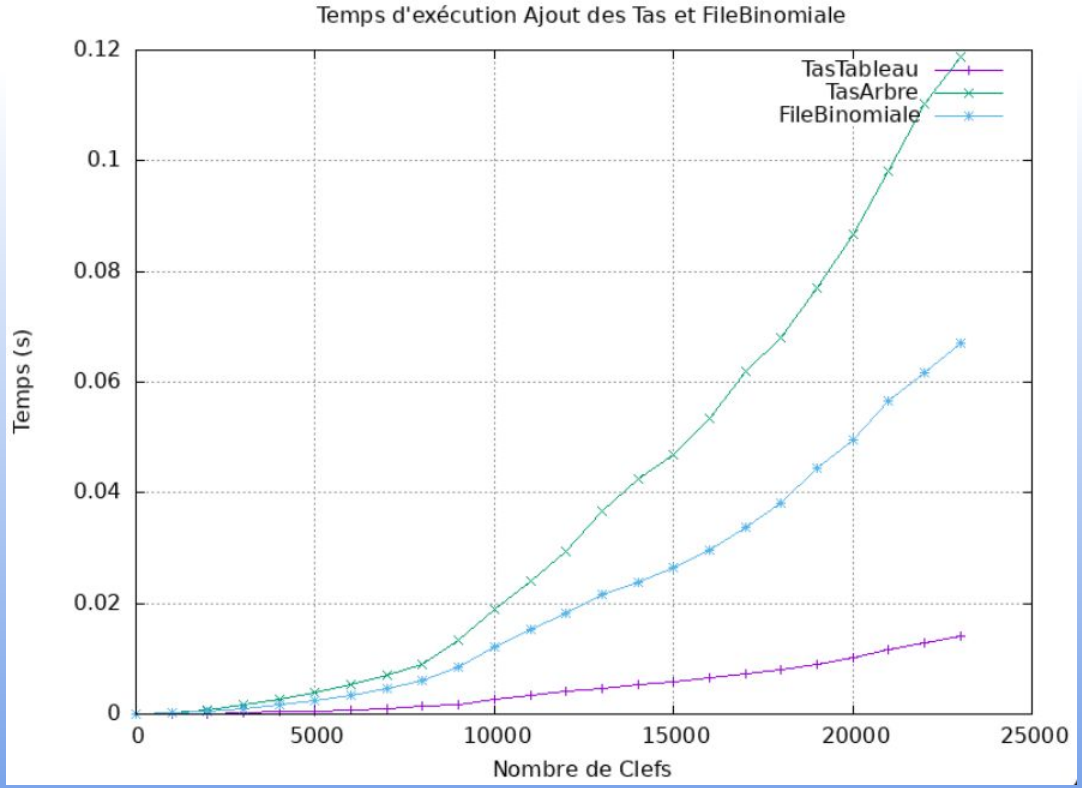
```
def UnionFile(F1, F2):
        """FileB * FileB -> FileB
           Renvoie la file binomiale union des deux files F1 et F2."""

        return UFret(F1, F2, vide)
```

```
def UFret(F1, F2, T):
        """FileB * FileB * TournoiB-> FileB
           Renvoie la file binomiale union de deux files et d'un tournoi."""

        if EstVide(T):  #pas de tournoi en retenue
                if EstVide(F1):
                        return F2
                if EstVide(F2):
                        return F1

                T1 = MinDeg(F1)
                T2 = MinDeg(F2)
                if Degre(T1) < Degre(T2):
                        return AjoutMin(T1, UnionFile(Reste(F1), F2))
                if Degre(T2) < Degre(T1):
                        return AjoutMin(T2, UnionFile(Reste(F2), F1))
                if Degre(T1) == Degre(T2):
                        return UFret(Reste(F1), Reste(F2), Union2Tid(T1,T2))

      ...
      else:   #T tournoi en retenue
                if EstVide(F1):
                        return UnionFile(File(T), F2)
                if EstVide(F2):
                        return UnionFile(File(T), F1)

                T1 = MinDeg(F1)
                T2 = MinDeg(F2)
                if Degre(T) < Degre(T1) and Degre(T) < Degre(T2):
                        return AjoutMin(T, UnionFile(F1, F2))
                if Degre(T) == Degre(T1) and Degre(T) == Degre(T2):
                        return AjoutMin(T, UFret(Reste(F1), Reste(F2), Union2Tid(T1, T2)))
                if Degre(T) == Degre(T1) and Degre(T) <  Degre(T2):
                        return UFret(Reste(F1), F2, Union2Tid(T1, T))
                if Degre(T) == Degre(T2) and Degre(T) <  Degre(T1):
                        return UFret(Reste(F2), F1, Union2Tid(T2, T))
```



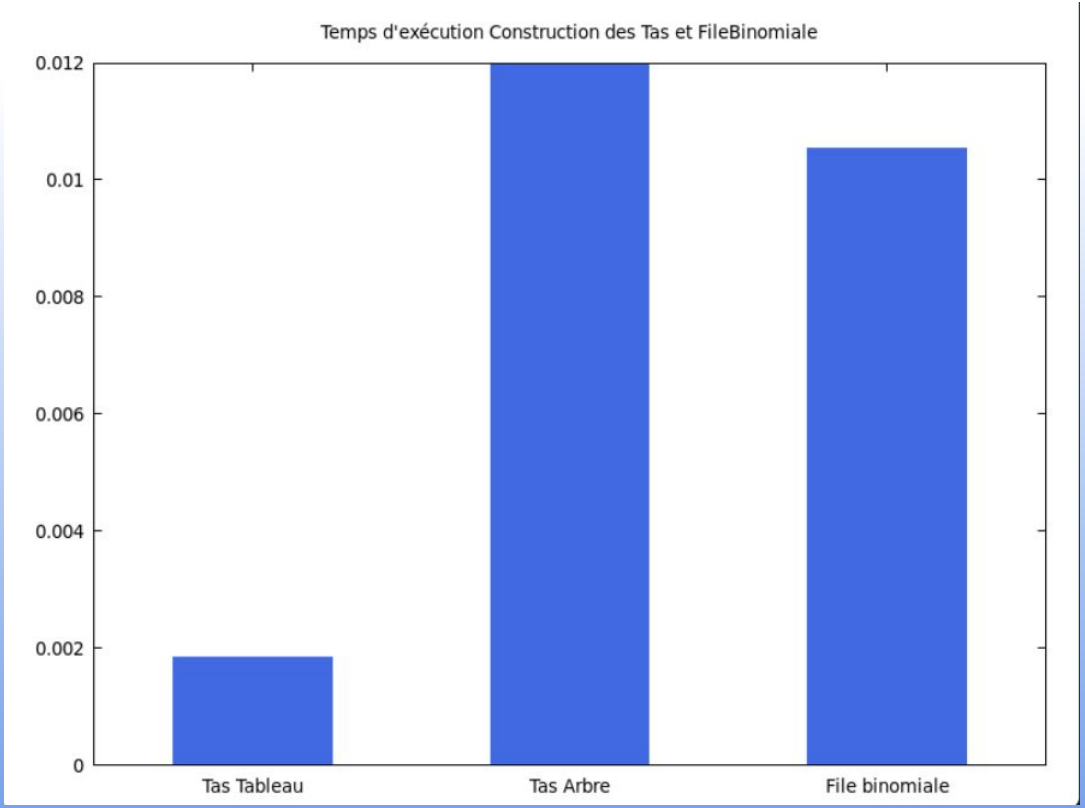
Etude Expérimentale - Ajout

1er Tas Min Tableau
2e File Binomiale
3e Tas Min Arbre



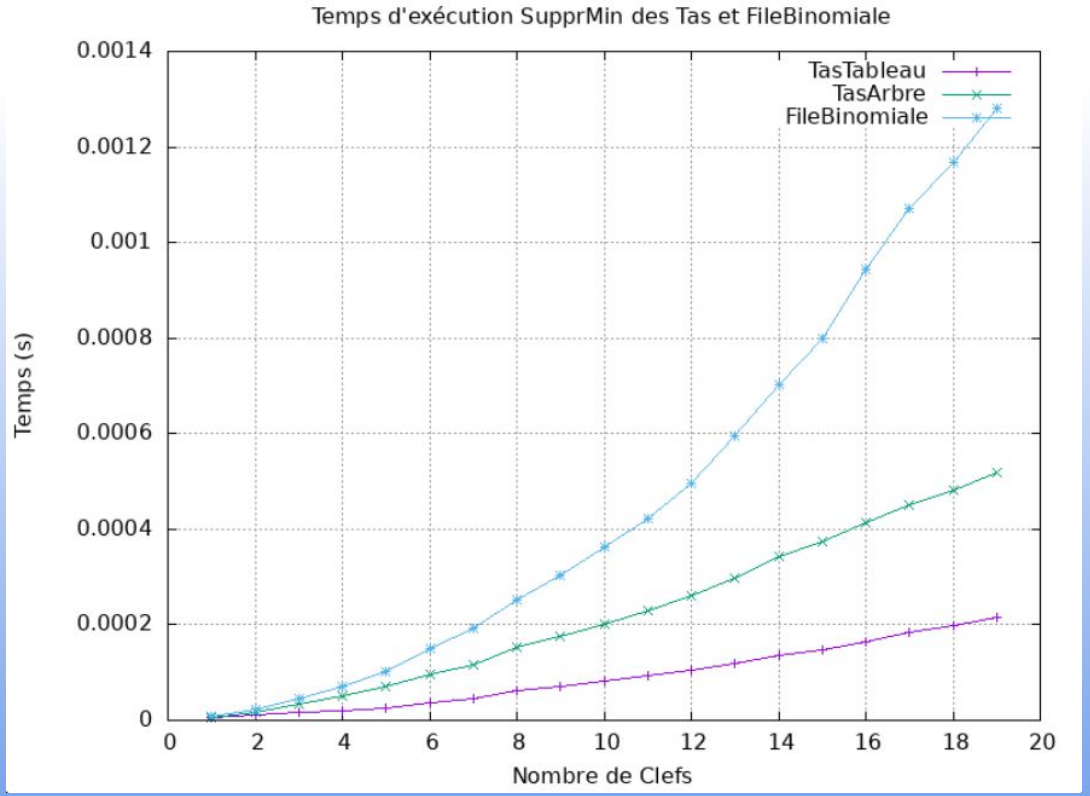
Etude Expérimentale - Construction 23 086 clefs

1er Tas Min Tableau
2e File Binomiale
3e Tas Min Arbre



Etude Expérimentale - Suppression

1er Tas Min Tableau
2e Tas Min Arbre
3e File Binomiale



Etude Expérimentale - Union de taille différente

1er File Binomiale
2e Tas Min Tableau
3e Tas Min Arbre