INFO0947: Projet 1 -> Construction de Programme

Groupe 07: Julien Peffer, Alexandre Bourguignon

Table des matières

1 Introduction

L'objectif de ce projet est de créer une fonction qui prend en entrées un tableau, sa taille, un pointeur vers un entier qui correspond à la position du minimum dans le tableau et un autre pointeur vers un entier qui correspond à la position du maximum dans le tableau. Cette fonction va retourner 3 éléments : la position du minimum et du maximum via un passage par adresse et également la somme entre le minmum et le maximum inclus. Il est important de noter que le minimum n'est pas forcément avant le maximum et également que la fonction ne doit contenir qu'une seule boucle de type "while". Par conséquent, la complexité de la fonction doit être en O(N). Nous allons donc parcourir, point par point, chaque étape au moyen d'une approche constructive.

2 Formalisation du Problème

Voici 3 prédicats dont un prédicat intermédiaire. Rappel : un prédicat est une fonction produisant une valeur boolénne.

2.0.1 Prédicat A

La notation est MAXPOS

La formalisation est $MAXPOS(T, Inf, Sup) \equiv$
$(\exists x \ = max_{i \in Inf...Sup-1},\ (T[i]))\ \wedge\ (\exists j,\ Inf \leq j \leq Sup-1,\ T[j] = x)\ \wedge\ (\exists maxPos,\ Inf \leq maxPos \leq Sup-1,\ maxPos = j)$

2.0.2 Prédicat B

La notation est MINPOS

La formalisation est $MINPOS(T, Inf, Sup) \equiv$
$(\exists y \ = min_{i \in Inf...Sup-1},\ (T[i]))\ \wedge\ (\exists j,\ Inf \leq j \leq Sup-1,\ T[j] = y)\ \wedge\ (\exists minPos,\ inf \leq minPos \leq Sup-1,\ minPos = j)$

2.0.3 Prédicat C

La notation est SUM

La formalisation est $SUM(T, N, min\_pos, max\_pos) \equiv$
$(\forall i,\ 0 \leq i \leq N-1,\ sum = \sum_{i=min(min\_pos,max\_pos)}^{max(min\_pos,max\_pos)} T[i])$

2.0.4 Prédicat D

La notation est NEWSUM

La formalisation est $NEWSUM(T, N, min\_pos, max\_pos) \equiv$
$(\forall i, 0 \leq i \leq N-1, \exists new\_sum = \sum_{a=max(min\_pos,max\_pos)}^{i-1} T[a]$

3 Définition et Analyse du Problème

3.1 Input

-T, tableau d'entiers
-N, taille du tableau
-$*min\_pos$, un pointeur vers la position du minimum
-$*max\_pos$, un pointeur vers la position du maximum

3.2 Output

La somme des éléments entre la position minimale et la position maximale (inclus)

3.3 Objets Utilisés

-T, tableau d'entiers ($int * T$)
-N, valeur entière ($int\ N$)
-sum, valeur entière ($int\ sum$)
-new_sum, valeur entière ($int\ new\_sum$)

3.4 Découpe en sous problèmes

SP 1 : calcul de MAXPOS(T,0,i) et de MINPOS(T, 0, i)
SP 2 : Mettre à jour l'indice de max_pos et min_pos respectivement par MAXPOS(T,0,i) et MINPOS(T,0,i)
SP 3 : Calcul de SUM(T,N,min_pos,max_pos)

3.5 Emboitement

SP1 $\rightarrow$ (SP3 $\subset$ SP2)

4 Specifications

- Spécification du SP1 & SP2

```

/*
*@pre: T initialisé ∧ 0 ≤ inf ≤ sup ≤ N − 1
*@post: N = N0 ∧ T = T0  ∧ max_pos = MAXPOS(T, 0, N)
*/
int max_pos(int *T, int inf, int sup);

/*
*@pre: T initialisé ∧ 0 ≤ inf ≤ sup ≤ N − 1
*@post: N = N0 ∧ T = T0  ∧ min_pos = MINPOS(T, 0, N)
*/
int min_pos(int *T, int inf, int sup);
```

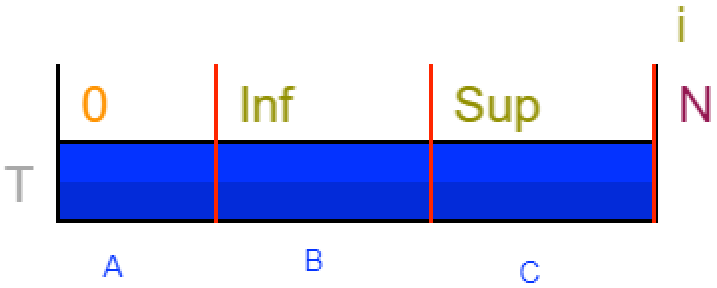
- Spécification du SP3

```
/*
*@pre: T initialisé ∧ N > 0 ∧ min_pos initialisé ∧ max_pos initialisé
*@post: T = T0 ∧ N = N0     ∧ sum = SUM(T, N, min_pos, max_pos)
*/
```

```
int sum(int *T, int N, int *min_pos, int *max_pos);
```

5 Invariants

5.1 Représentation de la Postcondition



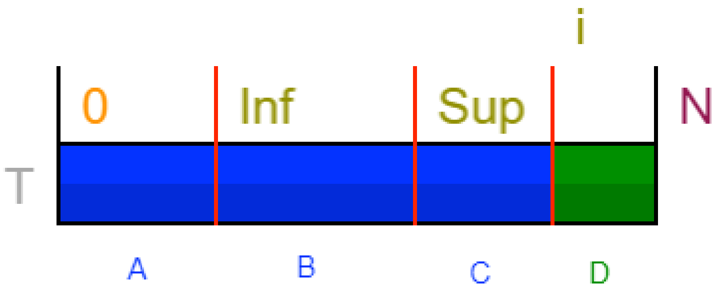
Avec :

A -> Déjà parcouru (sup $\neq MAXPOS(T, 0, inf) et\ inf \neq MINPOS(T, 0, inf)$)

$B->sum$ = SUM(T, Inf, Sup) , inf = MINPOS(T,0,N) et sup = MAXPOS(T,0,N)

$C->$Déjà parcouru (sup $\neq MAXPOS(T, 0, N) et\ inf \neq MINPOS(T, 0, N)$)

5.2 Représentation de l'invariant graphique



Avec :

A -> Déjà parcouru (sup $\neq MAXPOS(T, 0, inf) et\ inf \neq MINPOS(T, 0, inf)$)

$B->sum$ = SUM(T, Inf, Sup), inf = MINPOS(T, 0, i) et sup = MAXPOS(T, 0, i)

$C->new\_sum$ = NEWSUM(T,N,inf,sup)

$D->$Reste à parcourir

5.3 Invariant Formel

T = $T_0 \wedge N = N_0 \wedge 0 \leq inf \leq sup \leq N \wedge new\_sum = NEWSUM(T, N, min\_pos, max\_pos)$∧max_pos= $MAXPOS(T, 0, i)$∧min_pos= $MINPOS(T, 0, i) \wedge sum = SUM(T, N, min\_pos, max\_pos)$

6 Approche Constructive

- Spécification du problème général

```
/*
```

```
*@pre: T initialisé ∧ N > 0 ∧ min_pos initialisé ∧ max_pos initialisé
*@post: T = T0 ∧ N = N0    ∧ somme = SUM(T, N, min_pos, max_pos)
*/
int somme(T,N,*min_pos,*max_pos);
```

- Approche constructive

```
    int somme(T,N,*min_pos,*max_pos){
    assert(T != NULL && N > 0 && min_pos != NULL && max_pos != NULL);
    //PRE ≡ T initialisé ∧ N > 0 ∧ min_pos init ∧ max_pos init

    int min = T[0];
    // T = T0 ∧ N = N0 ∧ min = T[0]
    int max = T[0];
     // T = T0 ∧ N = N0 ∧ min = T[0] ∧ max = T[0]
    int sum = T[0];
    // T = T0 ∧ N = N0 ∧ min = T[0] ∧ max = T[0] ∧ sum = T[0]
    int new_sum = 0;
    // T = T0 ∧ N = N0 ∧ min = T[0] ∧ max = T[0] ∧ sum = T[0] ∧ new_sum = 0
    *min_pos = *max_pos = 0;
    /*
    T = T0 ∧ N = N0 ∧ min = T[0] ∧ max = T[0] ∧ sum = T[0]
    ∧ new_sum = 0 ∧ *min_pos = 0 ∧ *max_pos = 0
    */
    int i = 0;
    /*
    T = T0 ∧ N = N0 ∧ min = T[0] ∧ max = T[0] ∧ sum = T[0]
    ∧ new_sum = 0 ∧ *min_pos = 0 ∧ *max_pos = 0 ∧ i = 0
    => *min_pos = MINPOS(T,0,0) = MINPOS(T,inf,sup) ∧
    *max_pos = MAXPOS(T,0,0) = MAXPOS(T,inf,sup)
    ∧ sum = SUM(T,1,0,0) = SUM(T,1,inf,sup)
    ∧ new_sum = NEWSUM(T,1,min_pos,max_pos) = NEWSUM(T,1,inf,sup)
    */
    /*
    INV ≡ N = N0 ∧ T = T0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N ∧ *min_pos = MINPOS(T,0,i)
    ∧ *max_pos = MAXPOS(T,0,i) ∧ sum = SUM(T,i+1,*min_pos,*max_pos)
    ∧ new_sum = NEWSUM(T,i+1,*min_pos,*max_pos)
    */
    while(i < N){
    /*
    INV ∧ B ≡ N = N0 ∧ T = T0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N−1 ∧ *min_pos = MINPOS(T,0,i)
    ∧ *max_pos = MAXPOS(T,0,i) ∧ sum = SUM(T,i+1,*min_pos,*max_pos)
    ∧ new_sum = NEWSUM(T,i+1,*min_pos,*max_pos)
    */
    new_sum += T[i];
    /*
    N = N0 ∧ T = T0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N−1 ∧ *min_pos = MINPOS(T,0,i)
    ∧ *max_pos = MAXPOS(T,0,i) ∧ sum = SUM(T,i+1,*min_pos,*max_pos)
    ∧ new_sum = NEWSUM(T,i+2,*min_pos,*max_pos)
    */
     if(T[i] < min){
     /*
     N = N0 ∧ T = T0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N−1 ∧ *min_pos = MINPOS(T,0,i)
     ∧ *max_pos = MAXPOS(T,0,i) ∧ sum = SUM(T,i+1,*min_pos,*max_pos)
     ∧ new_sum = NEWSUM(T,i+2,*min_pos,*max_pos) ∧ T[i] < min
     */
           if(*max_pos <= *min_pos)
             /*
             N = N0 ∧ T = T0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N−1 ∧ *min_pos = MINPOS(T,0,i)
             ∧ *max_pos = MAXPOS(T,0,i) ∧ sum = SUM(T,i+1,*min_pos,*max_pos)
```

```
            ∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos) ∧ T[i] < min
            ∧ *max_pos ≤ *min_pos
            */
            sum += new_sum - T[*min_pos];
            /*
            N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
            ∧ *max_pos = MAXPOS(T, 0, i)
            ∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos) ∧ T[i] < min
            ∧ *max_pos ≤ *min_pos ∧ sum = SUM(T, i+1, *min_pos, *max_pos)+
            (new_sum - T[*min_pos])
            */

          else
            /*
            N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
            ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, *min_pos, *max_pos)
            ∧ new_sum = NEWSUM(T,i+2,*min_pos,*max_pos) ∧ T[i] < min ∧ *max_pos > *min_pos
            */

            sum = new_sum;
            /*
            N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
            ∧ *max_pos = MAXPOS(T, 0, i)∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos)
            ∧ T[i] < min ∧ *max_pos > *min_pos ∧ sum = new_sum
            */

      new_sum = T[i];
      /*
       N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
       ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, *min_pos, *max_pos)
       ∧ new_sum = T[i] ∧ T[i] < min
       */
      min = T[i];
      /*
       N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
       ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, *min_pos, *max_pos)
       ∧ new_sum = T[i] ∧ T[i] < min ∧ min = T[i]
       */
      *min_pos = i;
      /*
       N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1
       ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, i, *max_pos)
       ∧ new_sum = T[i] ∧ T[i] < min ∧ min = T[i] ∧ *min_pos = i
       */
      }
       if(T[i] > max){
       /*
       N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
       ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, *min_pos, *max_pos)
       ∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos) ∧ T[i] > max
       */
          if(*min_pos <= *max_pos)
            /*
            N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
            ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i+1, *min_pos, *max_pos)
            ∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos) ∧ T[i] > max
            ∧ *min_pos ≤ *max_pos
            */
            sum += new_sum - T[*max_pos];
            /*
```

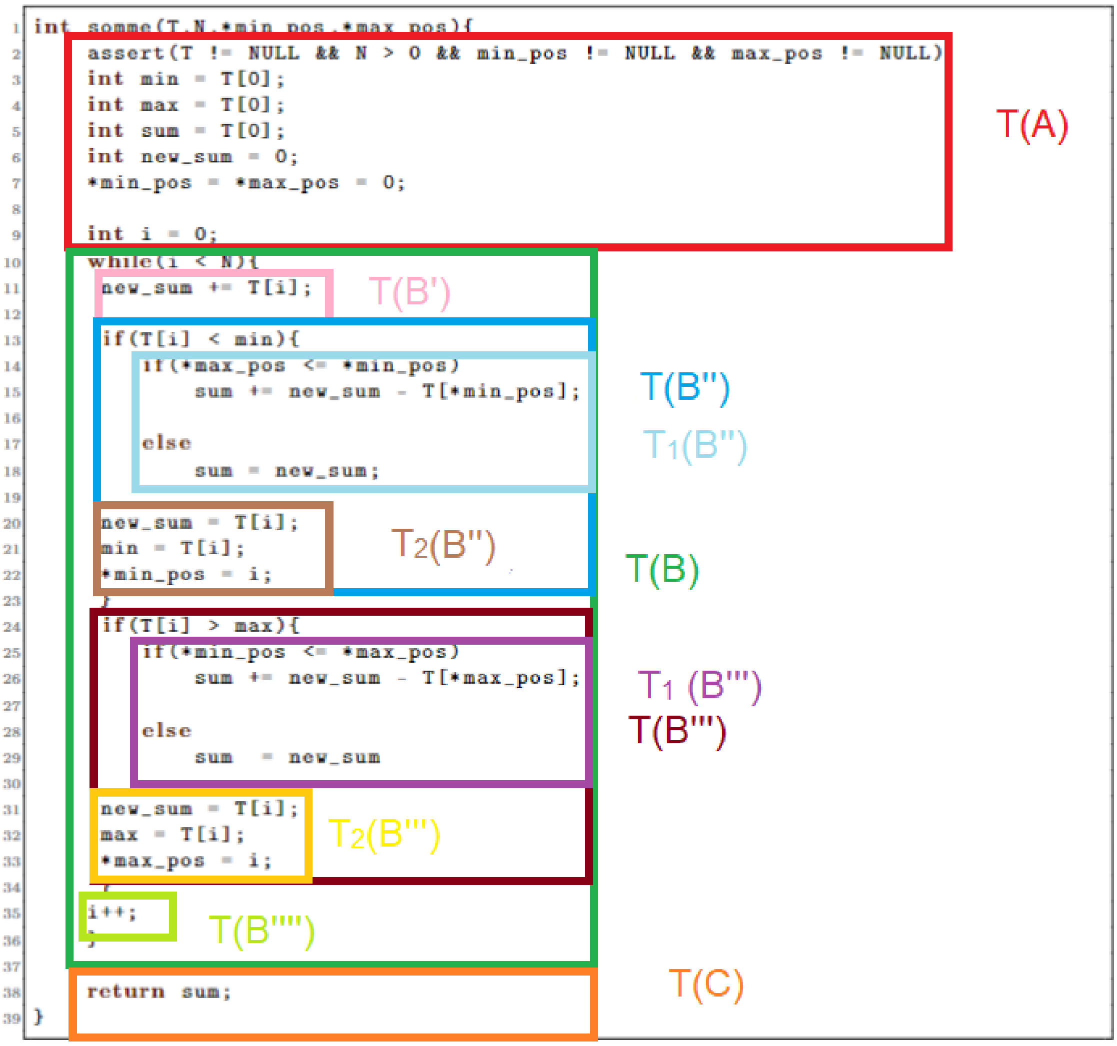
```
                N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
                ∧ *max_pos = MAXPOS(T, 0, i)
                ∧ new_sum = NEWSUM(T, i + 2, *min_pos, *max_pos) ∧ T[i] > max
                ∧ *min_pos ≤ *max_pos ∧ sum = SUM(T, i + 1,*min_pos,*max_pos) +
                (new_sum - T[*max_pos])
                */

            else
            /*
                N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
                ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i + 1, *min_pos, *max_pos)
                ∧ new_sum = NEWSUM(T, i+2, *min_pos, *max_pos) ∧ T[i] > max ∧ *min_pos > *max_pos
                */
                sum   = new_sum
                /*
                N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
                ∧ *max_pos = MAXPOS(T, 0, i) ∧ new_sum = NEWSUM(T, i + 2, *min_pos, *max_pos)
                ∧ T[i] > max ∧ *min_pos > *max_pos ∧ sum = new_sum
                */

        new_sum = T[i];
        /*
         N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
         ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i + 1, *min_pos, *max_pos)
         ∧ new_sum = T[i] ∧ T[i] > max
         */
        max = T[i];
        /*
         N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
         ∧ *max_pos = MAXPOS(T, 0, i) ∧ sum = SUM(T, i + 1, *min_pos, *max_pos)
         ∧ new_sum = T[i] ∧ T[i] > max ∧ max = T[i]
        */
        *max_pos = i;
        /*
         N = N_0 ∧ T = T_0 ∧ 0 ≤ inf ≤ sup ≤ i ≤ N − 1 ∧ *min_pos = MINPOS(T, 0, i)
         ∧ sum = SUM(T, i + 1, *min_pos, i)
         ∧ new_sum = T[i] ∧ T[i] > max ∧ max = T[i] ∧ *max_pos = i
        */
        }
        i++;
        /*
        INV ≡ 0 ≤ i ≤ N ∧ T = T_0 ∧ N = N_0
        ∧ *min_pos = MINPOS(T, 0, i + 1)
        ∧ *max_pos = MAXPOS(T, 0, i + 1) ∧ sum = SUM(T, i + 2, *min_pos, *max_pos)
        ∧ new_sum = NEWSUM(T, i + 3, *min_pos, *max_pos)
        */
        }
        /*
        INV ∧ ¬ B ≡ i == N ∧ T = T_0 ∧ N = N_0
        ∧ *min_pos = MINPOS(T, 0, N)
        ∧ *max_pos = MAXPOS(T, 0, N) ∧ sum = SUM(T, N, *min_pos, *max_pos)
        ∧ new_sum = NEWSUM(T, N, *min_pos, *max_pos)
        */
        return sum;
        /*
        POST ≡ T = T_0 ∧ N = N_0 ∧ sum = SUM(T, inf, sup)
        ∧ *max_pos = MAXPOS(T, 0, N) ∧ *min = MINPOS(T, 0, N)
}       */
```

7 Code Complet

```
int somme(T,N,*min_pos,*max_pos){
    assert(T != NULL && N > 0 && min_pos != NULL && max_pos != NULL)
    int min = T[0];
    int max = T[0];
    int sum = T[0];
    int new_sum = 0;
    *min_pos = *max_pos = 0;

    int i = 0;
    while(i < N){
     new_sum += T[i];

     if(T[i] < min){
        if(*max_pos <= *min_pos)
            sum += new_sum - T[*min_pos];

        else
            sum = new_sum;

         new_sum = T[i];
         min = T[i];
         *min_pos = i;
     }
     if(T[i] > max){
        if(*min_pos <= *max_pos)
            sum += new_sum - T[*max_pos];

        else
            sum  = new_sum

        new_sum = T[i];
        max = T[i];
       *max_pos = i;
     }
    i++;
    }

    return sum;
}
```

Extrait de Code 1 – somme.c

8 Complexité



Extrait de Code 1 – somme.c

- Par application des règles 2 et 6 :

$$T(N) = T(A) + T(B) + T(C)$$

- Par application des règles 1 et 5 :

$$T(N) = 8 + \sum_{i=0}^{N-1} (T(B') + T(B'') + T(B''') + T(B'''')) + 1$$

$$T(N) = 8 + \sum_{i=0}^{N-1} (1 + T(B'') + T(B''') + 1) + 1$$

- Quid de T(B”) ?

$$T(B'') = T_1(B'') + T_2(B'')$$

- Quid de T_1 (B”) ? Par application de la règle 3 :

$$T_1(B'') = max(T_{if}(), T_{else}())$$

$$T_{if}() = 1$$

et

$$T_{else}() = 1$$

donc

$$T_1(B'') = 1$$

- Quid de T_2(B") ? Par application de la règle 2 :

$$T_2(B'') = 3$$

- Ainsi T(B") devient :

$$T(B'') = 1 + 3$$

$$T(B'') = 4$$

- Quid de T(B"') ?

$$T(B''') = T_1(B''') + T_2(B''')$$

- Quid de T_1(B"') ? Par application de la règle 3 :

$$T_1(B''') = max(T_{if}(), T_{else}())$$

$$T_{if}() = 1$$

et

$$T_{else}() = 1$$

donc

$$T_1(B''') = 1$$

- Quid de T_2(B"') ? Par application de la règle 2 :

$$T_2(B''') = 3$$

- Ainsi T(B"') devient :

$$T(B''') = 1 + 3$$

$$T(B''') = 4$$

- Ainsi T(B) devient :

$$T(B) = \sum_{i=0}^{N-1} 10 = 10N$$

Donc :

$$T(N) = 8 + 10N + 1 = 9 + 10N$$

Par quoi borner T(N) ?

$$O(n)$$

complexité linéaire.
Preuve ?
But : Trouver une constante $c \in R^+$ et un seuil n_0 à partir duquel $T(n_0) \leq c$ x n_0.
On remarque que pour $c = 11$ et $n_0 = 9$ on a $\forall n \geq n_0$: c x n $\geq$ 9 + 10N
<u>exemple :</u>

- si n = 9 : 99 $\geq$ 99
- si n = 10 : 110 $\geq$ 109
- si n = 11 : ...

9 Conclusion

Il est donc possible de résoudre ce problème en O(N) (i.e en une seule boucle)