**VARIABLES**

*tab*, *T* et *n* sont initialisés pour tous les algorithmes avec

***LIRE*** *T,* ENTIER  
*tab* ← [ **POUR** ENTIER aléatoire (entre 0 et T) jusqu’à la borne T ]

**LIRE** *n, ENTIER*

*………………………………………………………………………………………………………………*

*debut* ← 0 ENTIER

*fin* ← *T* – 1 ENTIER

*trouve* ← FAUX BOULÉEN

*start3* ← enregistre\_temps FLOAT

**DEBUT**

**TANT QUE** *debut* ≤ *fin* && n ≥ *tab*[*debut*] && *n* ≤ *tab*[*fin*], **FAIRE**

*probplace* ← ‘FLOAT’ *debut* + (((*fin* - *debut*) \* (*n* - *tab*[*debut*])) / (*tab*[*fin*] - *tab*[*debut*]))

*probplace* ← arrondi\_vers\_l’entier\_inférieur(*probplace*) ENTIER

**SI** *debut* = *fin*, **ALORS**

**SI** *tab*[*debut*] = *n*, **ALORS**

*trouve* ← VRAI

**quitter la boucle** (« break »)

**FIN SI**

**FIN SI**

**SI** *tab*[*probplace*] = *n*, **ALORS**

*trouve* ← VRAI

**quitter la boucle**

**SINON SI** *tab*[*probplace*] < *n,* **ALORS**

*debut* ← *probplace* + 1

**SINON**

*fin* ← probplace - 1

**FIN SI**

**FIN TANT QUE**

*end3* ← enregistre\_temps FLOAT

**SI** *trouve* = VRAI, **ALORS**

**AFFICHER** "La valeur", *n*, "est à l'indice", *probplace*, "dans le tableau."

**SINON**

**AFFICHER** "Le nombre", *n*, "n'est pas dans le tableau."

**FIN SI**

**AFFICHER** "Temps de recherche par interpolation", end3 - start3

**FIN**

**Complexité :**

Le meilleur des cas se produit lorsque l’élément ‘n’ correspond à l’exacte milieu du tableau ‘tab’. On trouve alors l’élément à la première itération de la boucle TANT QUE. Aussi, la complexité temporelle (temps d’exécution) dans une recherche constante (trouve du premier coup) est Θ(1).

Le pire des cas se produit lorsque l’élément n’est pas dans le tableau ou lorsqu’il est aux bornes de celui-ci. On itère alors un maximum de fois pour converger vers la position probable de l’élément. La complexité temporelle de l’algorithme est alors Θ(log(log n)) où ‘n’ se confond avec la taille du tableau.

**Variant de boucle** :

Il s’agit de la différence entre *fin* et *début* car à chaque itération de la boucle, soit *début* est incrémenté (debut = probplace + 1), soit *fin* est décrémenté (fin = probplace – 1). On assure la fin de la boucle par la condition *début* > *fin*.

**Invariant de boulce**:

Il s’agit de l’élément qui reste vrai à chaque itération de la boucle. On peut considérer que n est l’invariant car il reste toujours entre les bornes du *tab,* *debut* et *fin*. On a donc toujours *debut* < n < *fin*.