Documentation

Recherche dichotomique :

```
ALGORITHME
        // cette fonction renvoie vrai si x est présente dans tab, faux sinon
        // le tableau tab est supposé trié par ordre croissant
        fonction avec retour booléen rechercheElementDichotomie(chaine tab[],
        chaine e)
                entier i, j
                booléen trouve
        début
                trouve <- faux
                i <- 0
                j <- tab.longueur-1
                tantque (i <= j et non trouve) faire
                        si (tab[(j+i)/2] = e) alors
                                trouve <- vrai
                        sinon
                                si (tab[(j+i)/2] > e) alors
                                        j <- (j+i)/2 - 1
                                sinon
                                        i <- (j+i)/2 + 1
                                finsi
                        finsi
                fintantque
                retourne trouve
        fin
```

Le tableau est supposé trié par ordre croissant et on cherche un élément e dans un tableau t.

Principe de l'algorithme :

- on regarde l'élément situé au milieu de t ;
- s'il s'agit de e c'est gagné ;
- s'il est plus grand que e, on cherche dans la moitié gauche ;
- s'il est plus petit que e, on cherche dans la moitié droite.

Remarques :

- ça ne peut marcher que si le tableau est trié
- on coupe le tableau en deux parties pas forcément égales en taille

Recherche interpolation :

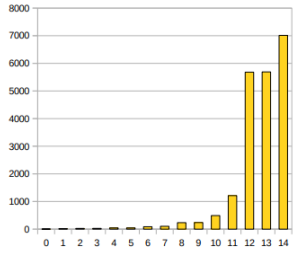
```
ALGORITHME
        // cette fonction renvoie vrai si x est présente dans tab, faux sinon
        // le tableau tab est supposé trié par ordre croissant
        fonction avec retour booléen rechercheElementInterpolation(chaine tab[],
        chaine e)
                entier i, j, pos
                booléen trouve
        début
                trouve <- faux
                i <- 0
                j <- tab.longueur-1
                tantque (i <= j et non trouve) faire
                        si (i = j et tab[i] = e) alors
                                trouve <- vrai
                        Finsi
                        pos = i + (((double)(j-i) / (tab[j]-tab[i])) * (e tab[i]))
                        si(tab[pos] == e)
                                trouve <- vrai
                        sinon si(tab[pos] < e)
                                i = pos+1
                        sinon
                                j = pos-1
                        Finsi
                fintantque
                retourne trouve
        fin
```

La recherche par interpolation reprend le principe de la recherche dichotomique, mais cette fois-ci on coupe le tableau à un endroit proche de la valeur recherché.

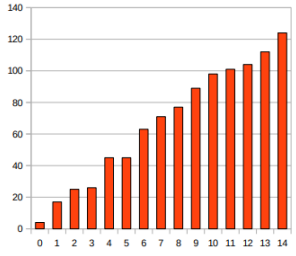
Exemple : Faire une recherche dans le dictionnaire est dite par interpolation.

4	17	25	26	45	45	87	102	234	237	490	1213	5681	5690	7012
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4	17	25	26	45	45	63	71	77	89	98	101	104	112	124
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



indice interpolé de 490 = 1



indice interpolé de 71 = 8

Recherche trie :

```
ALGORITHME
              fonction avec retour booléen testTrie1(entier tab[])
                     entier i;
                     booléen b;
              début
                     b <- VRAI;
                     pour (i allant de 0 à tab.longueur-2 pas 1) faire
                            si (tab[i] > tab[i+1]) alors
                                   b <- FAUX;
                            finsi
                     finpour
                     retourne b;
              fin
```

On cherche un entier i dans un tableau d'entier. La fonction retourne vrai si il existe.