Algorithmique
Recherches : dichotomie - interpolation

Les méthodes de recherche par dichotomie et interpolation linéaire se font sur des listes triées. De plus elles ne sont applicables que sur des représentations contiguës de listes (statiques).

1 Recherche dichotomique

Principe

Soit une liste L, et un élément x recherché et m le milieu de la liste L.
— Si $x = ième(L, m)$, la recherche est positive.
— Si $x < ième(L, m)$, on poursuit donc la recherche sur la moitié inférieure de la liste L.
— Si $x > ième(L, m)$, on poursuit donc la recherche sur la moitié supérieure de la liste L.
Si la recherche se termine sur une liste vide, la recherche est négative.

L'algorithme

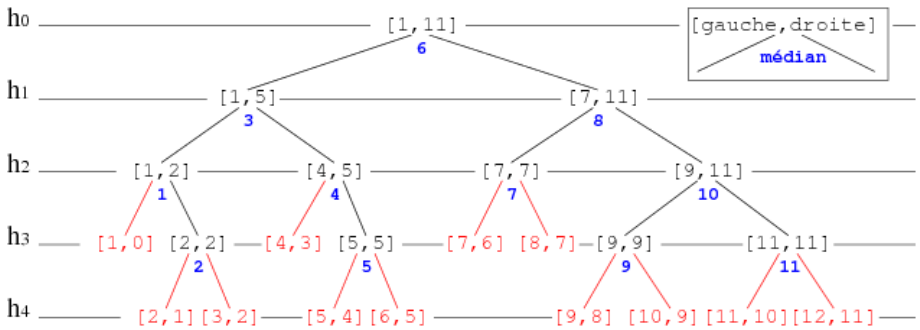
On travaille sur la sous-liste entre les bornes g et d. La fonction, appelée avec `1` et *`longueur`*`(L)`, retourne la position de x en cas de recherche positive, 0 en cas de recherche négative.

```
fonction dichotomie(liste L, élément x, entier g, d) : entier
variables
    entier m
debut
  si g > d alors
     retourne 0
  sinon
     m = (g + d) div 2     /* ou g + (d-g) div 2 */
     si x = ième(L, m) alors
        retourne m
     sinon
       si x < ième(L, m) alors
          retourne dichotomie(L, x, g, m-1)
       sinon
          retourne dichotomie(L, x, m+1, d)
       fin si
     fin si
  fin si
fin
```

```
                    /* version itérative */
g ← 1
d ← longueur(L)
tant que g <= d faire
   m = (g + d) div 2
   si x = ième(L, m) alors
      retourne m
   sinon
      si x < ième(L, m) alors
         d ← m-1
      sinon
         g ← m+1
      fin si
   fin si
fin tant que
retourne 0
```

Arbre de décision et complexité

L'arbre de décision ci-dessous représente l'exécution de cet algorithme appliqué à une liste triée de 11 éléments. En chaque nœud : l'intervalle de recherche (bornes gauche et droite) ainsi que l'indice du médian. Si l'élément se trouve à cette place la recherche se termine positivement, sinon on continue sur la branche appropriée. Si les bornes se croisent la recherche se termine négativement.



Chaque nœud interne correspond à 2 comparaisons.
Recherche négative : il faut $2h + 1$ comparaisons, avec h la hauteur/profondeur du nœud sur lequel on s'arrête.
Recherche positive : il faut $2(H + 1)$ comparaisons au pire, avec H la hauteur de l'arbre.

L'arbre étant équilibré, sa hauteur est $\lceil log_2(n) \rceil$. La complexité dans tous les cas est donc logarithmique.

Modification : recherche de la première occurrence

Dans le cas où la liste présente plusieurs occurrences des éléments, il peut arriver que l'on veuille la première occurrence de la valeur cherchée.

On modifie la fonction de la manière suivante : lorsque l'élément est trouvé, au lieu de s'arrêter on continue sur la sous-liste à gauche, mais en gardant la position actuelle.

```
fonction dichotomie_prems(liste L, élément x, entier g, d) : entier
variables
    entier m
debut                                   /* on suppose la liste non vide au départ */
    m = (g + d) div 2
    tant que g < d faire
        si x <= ième(L, m) alors
            d ← m
        sinon
            g ← m+1
        fin si
        m = (g + d) div 2
    fin tant que
    si ième(L, g) = x alors
        retourne m
    sinon
        retourne 0
    fin si
fin
```

2 Recherche par interpolation

Le principe de la recherche par interpolation est similaire à celui de la dichotomie. La différence réside dans le calcul de l'estimation de la place probable de l'élément recherché (selon une interpolation linéaire).

Soit x la valeur à rechercher entre les positions g et d dans la liste L triée en ordre croissant :
Calcul de la "pente" :

$$pente = \frac{d - g}{ième(L, d) - ième(L, g)}$$

x sera comparé à l'élément en place :

$$p = g + (x - ième(L, g)) \times pente$$

La fonction d'interpolation est la même que la dichotomie : il suffit de remplacer
`m ← (g+d) div 2`
par
`p ← g + (x - ième(L, g)) * (d-g) div (ième(L, d) - ième(L, g))`
lorsque `g < d`.

Limites et complexité

— La difficulté par rapport à la dichotomie est de donner une bonne fonction d'interpolation, l'idéal étant d'avoir une fonction de répartition uniforme des éléments dans la liste. En terme de complexité, si n est le nombre d'éléments de la liste, la dichotomie est de l'ordre $log_2(n)$ et l'interpolation (si la répartition est uniforme) de l'ordre de $log_2(log_2(n))$.

— Cela semble beaucoup plus intéressant que la dichotomie, mais il y a un bémol : il faut que les données de la liste soient numériques ou numérisables rapidement. En effet, la clé intervient dans le calcul de la place, ce qui n'est pas le cas pour la dichotomie. D'où une perte de temps de calcul pas forcément négligeable. D'autre part, il faut que la liste soit très grande pour que la différence entre $log_2(n)$ et $log_2(log_2(n))$ soit suffisamment grande.