Recherche dichotomique dans un tableau [re04]
Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 21 mai 2018

Table des matières

Java - Recherche dans un tableau (Solution)

Mots-Clés Algorithmes de recherche ■
Requis Axiomatique impérative sauf Fichiers ■
Fichiers UtilsTB ■
Difficulté • ◦ ◦ (30 min) ■

Objectif
Cet exercice réalise l'algorithme de la recherche dichotomique dans un tableau. Dans le même ordre d'idées, l'exercice @[Recherche séquentielle dans un tableau] construit un algorithme dans le cas où le tableau n'est pas trié.

AJOUT RECURSIVITE JUIN 2017 ■

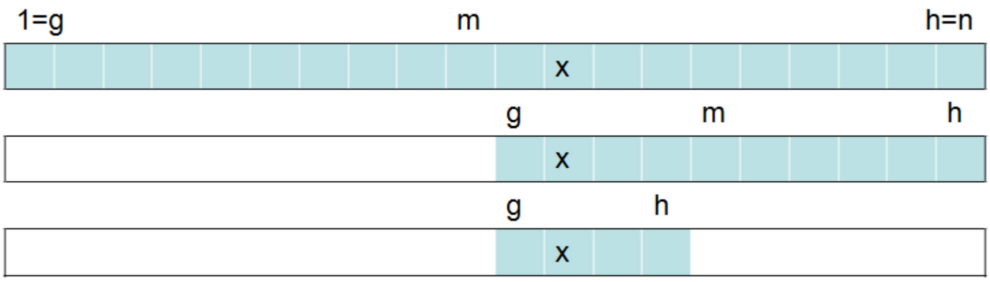
1 Algorithme de la recherche dichotomique

Soit une structure tabulaire `A[1..n]` triée en ordre croissant. On effectue une recherche dichotomique d'une valeur `x` comme suit.
Soient :

- `g` l'indice de gauche initialisé à 1.
- `h` l'indice de droite initialisé à `n`.
- `trouve` le booléen de la recherche initialisé à `Faux`.
- `m` l'indice milieu de l'intervalle `[g..h]`.

Alors :

1. Si `A[m]=x` alors on fixe `trouve` à `Vrai`.
2. Sinon si `A[m]<x` alors la position de `x` est forcément après `m` d'où on fixe `g` en `m+1`. Dans le cas contraire, on fixe `h` en `m-1` (car `A[m]>x`).
3. On répète les opérations (1) et (2) jusqu'à ce que `trouve` soit `Vrai` ou que `g` soit plus grand que `h`.



2 Recherche dichotomique / pgdichotab

On considère un tableau d'entiers `t` *trié* par ordre croissant de `n` éléments. On cherche à construire un algorithme permettant de savoir à quel endroit se trouve une valeur `x`.

On suppose que `x` est dans le tableau.

Écrivez une fonction `rechDicho1(t,n,x)` qui effectue une recherche dichotomique de `x` (entier) parmi les `n` éléments d'un `Tableau t` **trié** et qui renvoie l'indice d'une occurrence (pas forcément la première) de `x`.

Validez votre fonction avec la solution.

Solution Java @[pgdichotab.java]

```java
static int rechDicho1(/*const*/int[] t, int n, int x){
  boolean trouve = false;
  int g = 0;
  int h = n-1;
```

```
  int m;
  do{
    m = (g+h)/2;
    if (t[m] == x){
      trouve = true;
    }
    else if (t[m] < x){
      g = m+1;
    }
    else{
      h = m-1;
    }
  } while (!trouve);
  return (m);
}
```

Exécutez votre algorithme sur les données suivantes :
`n=10, t=[1,7,8,9,12,15,15,22,30,31]| et \lstinline`x=15@.

Solution simple

On représente les valeurs des variables `g`, `h`, `m`, `t[m]`, `trouve` à l'initialisation et à la fin de chaque itération (F signifie `Faux` et V signifie `Vrai`).

	initialisation	fin itération 1	fin itération 2	fin itération 3
g	1	1	6	6
h	10	10	10	7
m	-	5	8	6
t[m]	-	12	22	15
trouve	F	F	F	V

Comment faut-il modifier l'algorithme si l'on n'est pas sûr que `x` appartienne au tableau ?

Aide simple

Il faut ajouter la possibilité de terminer la boucle quand `g>h` ce qui signifie que `x` n'est pas dans le tableau.

Copiez/collez la fonction `rechDicho1` en la fonction `rechDicho2(t,n,x)` puis modifiez-la de sorte que la fonction renvoie -1 en cas de recherche infructueuse.

Validez votre fonction avec la solution.

Solution Java @[pgdichotab.java]

```
static int rechDicho2(/*const*/int[] t, int n, int x){
  boolean trouve = false;
  int g = 0;
  int h = n-1;
```

```
  int m = -1;
  while (g <= h && !trouve){
    m = (g+h)/2;
    if (t[m] == x){
      trouve = true;
    }
    else if (t[m] < x){
      g = m+1;
    }
    else{
      h = m-1;
    }
  }
  return (trouve ? m : -1);
}
```

3 Recherche récursive / pgdichotab2

Cet exercice réalise la recherche dichotomique en table.

3.1 Recherche dichotomique récursive

On appelle *milieu* de deux entiers N et M l'entier $\frac{N+M}{2}$ lorsque $N + M$ est pair et l'entier $\frac{N+M-1}{2}$ lorsque $N + M$ est impair. Soient n un entier positif, $t[1..NMAX]$ un tableau d'au moins n éléments tous distincts classés par ordre croissant et e une valeur. On chercher le rang p de e par la stratégie suivante.

- Au départ $1 \leq p \leq n$. On note m le milieu de n et de 1.
- Si $t[m] == e$: on a terminé.
- Dans le cas contraire, si $t[m] < e$, il ne peut apparaître qu'après $m + 1 \leq p \leq n$; sinon $(e < A[m])$ et donc $1 \leq p \leq m - 1$. On recommence en prenant le milieu de $m + 1$ et de n ou le milieu de 1 et de $m - 1$, et ainsi de suite jusqu'à ce que l'on trouve le rang p de e.

Écrivez une fonction récursive `rechdicho(t,p,r,e)` qui recherche le rang d'un entier de valeur `e` dans un tableau `t[p..r]` en utilisant la stratégie énoncée.

Calculez sa complexité.

Prouvez la validité de la stratégie.

Solution simple

La stratégie définit deux suites d'entiers $(u_k$ et (v_k) telles que pour tout k, on a : $u_k \leq p \leq v_k$ avec $u_1 = 1$ et $v_1 = n$. Soit n_k le milieu de u_k et de v_k. On compare $t[n_k]$ à e. Suivant les résultats on aura :

- Soit $p == n_k$ (cas $t[m] == e$)
- Soit $u_{n+k} == u_k$ et $v_{k+1} == n_k - 1$ (cas $e < t[m]$)
- Soit $u_{k+1} == n_k + 1$ et $v_{k+1} == v_k$ (cas $t[m] < e$)

Comme $v_{k+1} - u_k - 1 < \frac{v_k - u_k}{2}$, il existe forcément k_0 tel que $n_{k_0} = p$.

4 Références générales

Comprend [Moliner-ML1 :c2 :ex11] ■