RUNSER Laure — EA3 : CMTD 5 Recherche dichotomique — 15.10.2020

→ algo du 1er cours : complexité n = bcp trop pour qqch qui est utilisé si souvent.

Dichotomie / Diviser-pour-régner :

Entrée : tableau trié t, min le premier indice, max le dernier, e un élément

Sortie : i tq t[i] = e

Algo :

```
si min = max
    si tab[min] = e renvoyer min
    sinon renvoyé NonTrouvé

mid ← (min + max) / 2

si tab[min] < e  renvoyer RechercheDicho(tab, mid+1, max, e)

sinon renvoyer RechercheDicho(tab, min, mid, e)
```

Complexité : $\log_2(n)$

Version itérative (= complexité)

```
min ← 0
max ← n-1
tant que min < max
    mid ← (min + max) / 2
    si tab[mid] = e renvoyer mid
    si tab[min] < e
        min ← mid+1
    sinon max ← mid
si tab[min] = e renvoyer min
sinon renvoyer NonTrouvé
```

Preuve par récurrence sur $m = \beta - \alpha + 1$ taille de l'intervalle

HR ∀ tableau trié tab et ∀ intervalle $[\alpha, \beta]$, on renvoie la bonne valeur.

Init si $m = 1$ alors $\alpha = \beta$
Si élément trouvé, on renvoie α
Sinon NonTrouvé
(OK)

Hérédité Soit $[\alpha, \beta]$ tq $\beta - \alpha + 1 = m + 1$ et $\gamma = (\alpha + \beta)/2$

Alors l'exécution renvoie l'algo évalué :
soit sur (tab, γ, β, e)
soit sur (tab, α, γ, e)

Comme $\beta - (\gamma + 1) + 1 \leq m$

EA TD 5

Exercice 1

1) RECURSIF

a) 1 2 42 57 99

1 2 42

2 42 retourn 2

3 comparaisons ✓

b) 1 2 3 4 42

4 42 retourn 4

2 comparaisons ✓

c) 1 2 3 57 99

57 99

retourn NonTrouvé

2 comparaisons ✓

Exercice 2

1)

```
for (int i = 0 ; i<n; i++){
    if (t[i] == e) return i;
}
return -1;
```

Pire des cas = l'élément n'est pas dans le tableau

↳ n comparaisons ✓

2) En moyenne : $\frac{n}{2}$ comparaisons

↳ = l'élément est au milieu du tableau. ✓

3) a) n recherches linéaires = $n.n$ ✓

b) tri par insertO = n^2

n recherches dichotomiques = $n \log_2(n)$

} au total $n^2 + n\log_2(n)$ ✓

Exercice 3

1) n = 4 2 passes

0 | 1 | 2 | 3

n = 8 3 passes

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

n = 9 4 passes

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

2) $n = 2^k$ pièces

```
PieceFausseDicho (ensemble de n pièces, min le 1er index, max le dernier)
    si max == min renvoyer min
    mid ← (max + min) / 2
    si (pesée(min, mid) < pesée(mid + 1, max)
        retourner PieceFausseDicho(min, mid)
    sinon retourner PieceFausseDicho(mid + 1, max)
}
```

$\log_2(n)$ pesées

3) même chose mais la pesée doit être coefficientée pour prendre en compte le fait que la partie gauche a 1 pièce de plus.

$\log_2(n)+1$ pesées dans le pire des cas (tjrs à gauche)

Exercice 4

1) pré : -1 0 1
post : 1 3 4 8

2) · triés
· plus gd/petit ou égaux au pt fixe
= au plus petit/gd indice des éléments.

3)
```
public static boolean aPtFixe(int [] t){
    int min = 0;
    int max = t.length;
    while( min < max) {
        int mid = (min + max) /2;
        if (t[mid] <= mid) max = mid;
        else min = mid+ 1;
    }
    return t[min] == min;
}
```

Dans le pire des cas : $\log_2(n)$ comparaisons