

MÉTHODES PAR DICHOTOMIE

Objectifs

A la fin de la séquence d'enseignement les élèves doivent :

- pouvoir citer les avantages à adopter une approche par dichotomie en comparaison avec une approche linéaire
- reconnaître un cas où une approche par dichotomie est envisageable
- savoir coder en **Python** un algorithme de recherche dichotomique dans un tableau trié
- avoir quelques notions de coûts linéaire et logarithmique

Table des matières

| | | |
|-----|--|---|
| 1 | Principe | 1 |
| 2 | Exemples classiques | 1 |
| 2.1 | Recherche dichotomique d'une valeur dans un tableau trié | 1 |
| 2.2 | Exponentiation rapide | 2 |

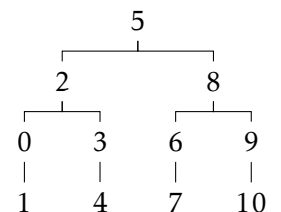
1 Principe

Il est des situations où un problème de dimension n se trouve réduit à un problème de dimension $n//2$ à l'itération suivante.

EXEMPLE : trouver un nombre entre 0 et 100 choisi par un opérateur (pas forcément humain) qui indique si la proposition est plus grande ou plus petite que la solution.

Tester les valeurs par ordre croissant peut conduire à 101 propositions.... Avec une méthode par dichotomie, le nombre d'itérations est au maximum 7.

Ci-contre, arbre d'appel pour une valeur comprise entre 0 et 10.



Le principe est donc de diviser le problème en deux sous problèmes. On parle alors de **méthode par dichotomie**. Elle fait partie des stratégies de type *diviser pour mieux régner*.

L'étude de la complexité au second semestre montrera qu'une telle méthode a un coût en $\log(n)$ quand une approche naïve a un coût en n (coût linéaire).

2 Exemples classiques

2.1 Recherche dichotomique d'une valeur dans un tableau trié

ENONCÉ : Recherche dichotomique (ou binary search) d'une valeur dans un tableau de valeurs trié

|| Écrire une fonction `recherche_dicho(T, x)` qui prend en argument un tableau trié T de flottants et x un flottant. La fonction renvoie un indice de x si x est une des valeurs du tableau et -1 sinon.

PRINCIPE :

Prendre une valeur au milieu du tableau. Si c'est la valeur recherchée, c'est gagné, sinon on la compare à la valeur recherchée et on prendre la moitié du tableau où est potentiellement la valeur recherchée.

Algorithm 1 Recherche dichotomique dans un tableau trié

| | |
|--|---|
| entrée: T un tableau de n valeurs et x un élément (pas forcément du tableau) résultat: -1 si $x \notin T$, l'indice d'un x dans le tableau T sinon 1: recherche_dicho(T, x) 2: $n \leftarrow \text{taille}(T)$ 3: $\text{min}, \text{max} \leftarrow 0, n-1$ 4: $\text{mil} \leftarrow (\text{min} + \text{max}) // 2$ | 5: tant que $\text{min} < \text{max}$ et $T[\text{mil}] \neq x$ faire 6: si $T[\text{mil}] < x$ alors 7: $\text{min} \leftarrow \text{mil} + 1$ 8: sinon 9: $\text{max} \leftarrow \text{mil} - 1$ 10: fin si 11: $\text{mil} \leftarrow (\text{min} + \text{max}) // 2$ 12: fin tant que 13: si $T[\text{mil}] \neq x$ alors 14: renvoi: $\leftarrow -1$ 15: sinon 16: renvoi: $\leftarrow \text{mil}$ 17: fin si |
|--|---|

2.2 Exponentiation rapide**ENONCÉ : Exponentiation rapide**

Écrire une fonction `puiss(x, n)` qui prend en argument un flottant x et un entier n et qui renvoie x^n .

EXEMPLE : pour calculer x^5 , on calcule x^2 puis x^4 et finalement x^5 .

On remarque que $x^9 = x \cdot (x^4)^2 = x \cdot ((x^2)^2)^2$.

On calcule x^2 puis x^4 puis x^8 . On passe ainsi de n multiplications à $\log_2(n)$ multiplications.

PRINCIPE :

Se servir des résultats intermédiaires pour accélérer le processus de calcul de x^n en décomposant n en puissance de 2.

Algorithm 2 Exponentiation rapide itérative

entrée: x un nombre réel et n un entier naturel
résultat: un nombre réel $r = x^n$
`puiss(x, n)`
1: **si** $n == 0$ **alors**
2: **renvoi:** 1
3: **fin si**
4: $r \leftarrow 1$
5: **tant que** $n > 0$ **faire**
6: **si** $n \bmod 2 == 1$ **alors**
7: $r \leftarrow r \cdot x$
8: **fin si**
9: $x \leftarrow x \cdot x$
10: $n \leftarrow n // 2$
11: **fin tant que**
12: **renvoi:** r

Algorithm 3 Exponentiation rapide récursive

entrée: n un entier positif et x un nombre réel
résultat: un nombre réel $r = x^n$
`puis_rec(x, n)`
1: **si** $n == 0$ **alors**
2: **renvoi:** 1
3: **sinon**
4: **si** $n \bmod 2 == 0$ **alors**
5: **renvoi:** `puis_rec(x, n/2)`
6: **sinon**
7: **renvoi:** $x \cdot \text{puis_rec}(x, (n-1)/2)$
8: **fin si**
9: **fin si**
