

TP 6

Algorithmes Dichotomiques – Sujet

Recherche dichotomique dans une liste triée

On se donne une liste L de nombres de longueur n , triée dans l'ordre croissant, et un nombre x_0 . Pour chercher x_0 , on va couper la liste en deux moitiés et chercher dans la moitié de tableau qui encadre x_0 et ainsi de suite... On appelle g l'indice de l'élément du début de la sous-liste dans laquelle on travaille et d l'indice de l'élément de fin. Au début, $g = 0$ et $d = n-1$.

On utilise la méthode suivante :

- ▶ on compare x_0 à « l'élément du milieu » $L[m]$ avec $m = (g + d) // 2$;
- ▶ si $x_0 = L[m]$, on a trouvé x_0 , on peut alors s'arrêter;
- ▶ si $x_0 < L[m]$, c'est qu'il faut chercher entre $L[g]$ et $L[m-1]$;
- ▶ si $x_0 > L[m]$, c'est qu'il faut chercher entre $L[m+1]$ et $L[d]$.

On poursuit jusqu'à ce qu'on a trouvé x_0 ou lorsque l'on a épuisé la liste L .

Question 1 Illustrer la méthode avec les deux exemples suivants :

- ▶ $x_0 = 5$ et $L =$

-3	5	7	10	11	14	17	21	30
----	---	---	----	----	----	----	----	----
- ▶ $x_0 = 11$ et $L =$

-2	1	2	7	8	10	13	16	17
----	---	---	---	---	----	----	----	----

.

Question 2 Si x_0 n'est pas dans la liste L , donner un test d'arrêt du processus de dichotomie portant sur g et d .

Question 3 Écrire une fonction `dichotomie(x0,L)` qui renvoie `True` ou `False` selon que x_0 figure ou non dans L par cette méthode. On utilisera une boucle `while` que l'on interrompra soit lorsque l'on a trouvé x_0 , soit lorsque l'on a fini de parcourir la liste.

Question 4 Écrire une fonction `dichotomie_indice(x0,L)` qui renvoie l'indice de x_0 s'il existe, -1 sinon.

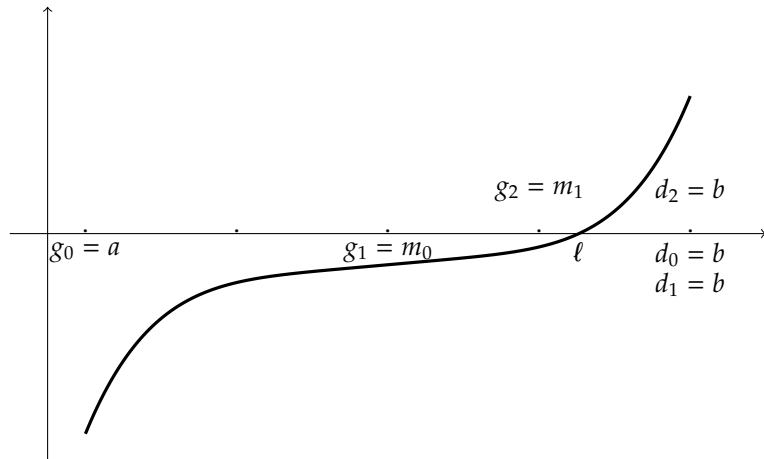
Question 5 Tester votre fonction sur les exemples de listes ci-dessus.

Question 6 (Facultatif) Combien vaut $g-d$ au i^e tour de boucle ? Si x_0 ne figure pas dans L , montrer que le nombre de tours de boucles nécessaires pour sortir de la fonction est de l'ordre de $\ln n$ où $n = \text{len}(L)$ (cela rend la fonction beaucoup plus efficace qu'une simple recherche séquentielle pour laquelle le nombre de comparaisons pour sortir de la boucle serait de l'ordre de n).

Recherche d'un zéro d'une fonction

Soit une fonction $f : [a, b] \rightarrow \mathbb{R}$ ($a < b$) vérifiant : f continue sur $[a, b]$ et $f(a) \cdot f(b) \leq 0$ c'est-à-dire que $f(a)$ et $f(b)$ de signes opposés.

Le théorème des valeurs intermédiaires s'applique et assure que f possède au moins un zéro ℓ entre a et b .



L'idée consiste à créer une suite d'intervalles $[g_n, d_n]$ tels que pour tout entier naturel n , $g_n \leq \ell \leq d_n$ et $0 \leq g_n - d_n = \frac{g_{n-1} - d_{n-1}}{2}$.

On considère $m_0 = \frac{g_0 + d_0}{2}$ et on évalue $f(m_0)$:

- ▶ Si $f(m_0) \times f(d_0) \geq 0$, on va poursuivre la recherche d'un zéro dans l'intervalle $[g_1, d_1] = [g_0, m_0]$.
- ▶ Sinon, on poursuit la recherche dans l'intervalle $[g_1, d_1] = [m_0, d_0]$.
- ▶ On recommence alors en considérant $m_1 = \frac{g_1 + d_1}{2}$...

Question 7 Si l'on souhaite que g_n et d_n soient des solutions approchées de ℓ à une précision ε , quelle est la condition d'arrêt de l'algorithme? Préciser alors la valeur approchée de ℓ qui sera renvoyée par la fonction.

Question 8 Écrire une fonction `recherche_zero(f, a, b, epsilon)` qui renvoie une valeur approchée du zéro de f sur $[a, b]$ à epsilon près.

Question 9 Tester la fonction avec $f : x \mapsto x^2 - 2$ sur $[0, 2]$ et $\varepsilon = 0,001$.

Valeur d'un polynôme par plusieurs méthodes

Question 10 Écrire une fonction `exponaif(x, n)` d'arguments un réel x et un entier naturel n , qui renvoie la valeur de x^n par la méthode naïve $x^n = x \times x \times \dots \times x$ (n termes). Compter le nombre d'opérations dans `exponaif`.

Une autre méthode, celle de l'exponentiation rapide consiste à remarquer que $x^n =$

$$\begin{cases} (x^2)^{n/2} & \text{si } n \text{ pair} \\ x \times (x^2)^{(n-1)/2} & \text{si } n \text{ impair} \end{cases}$$

Le code itératif correspondant est le suivant :

```

1 def expo_rapide(x,n) :
2     p,res,y = n,1,x
3     while p>0 :
4         if p%2==1 :
5             res=res*y
6             p=p//2
7             y=y*y
8     return(res)

```

Question 11 Quel est le nom de la variable locale dont le contenu est retourné par la fonction ?

Question 12 Faire tourner « à la main » la fonction pour $x = 2$ et $n = 10$ en complétant le tableau suivant puis encadrer le nombre d'opérations dans `exporapide` en fonction de $\ln(n)/\ln(2)$.

	p	res	y
sortie du 1er tour de boucle			
sortie du 2 ^e tour de boucle			
...			
...			

On considère un polynôme $P(x) = \sum_{k=0}^n a_k \cdot x^k$ que l'on modélisera en Python par la liste $P = [a_0, a_1, \dots, a_n]$. Dans la suite, on prendra pour tout $k \in \mathbb{N}$, $a_k = k$.

Question 13 Ecrire une fonction `Pnaif(x,P)` d'arguments un réel x et P la liste des coefficients du polynôme, qui renvoie $P(x)$ à l'aide de la fonction `exponaif`. Compter le nombre d'opérations.

Question 14 Faire de même pour une fonction `Prapide(x,n)` qui renvoie $P(x)$ à l'aide de la fonction `exporapide`. On admettra que la complexité est en $O(n \ln(n))$.

Une dernière méthode consiste à utiliser le schéma de Hörner : $P(x) = ((a_n x + a_{n-1})x + a_{n-2})x + \dots + a_1)x + a_0$

Question 15 Écrire une fonction `horner(x,L)` de paramètres un réel x et une liste L représentant un polynôme P , renvoie la valeur de $P(x)$ par la méthode de Hörner. Compter le nombre d'opérations.

On désire maintenant visualiser les temps d'exécution des trois fonctions précédentes pour des grandes valeurs de n .

Question 16 Définir la liste N des entiers naturels compris entre 0 et 100.

Question 17 Grâce à la fonction `perf_counter` de la bibliothèque `time`, écrire une fonction `Temps_calcul(x)` qui :

- ▶ définit 3 listes `Tn`, `Tr` et `Th` contenant les temps de calcul de $P(x)$ pour $P = \sum_{k=0}^n k \cdot x^k$ lorsque n décrit N avec respectivement la méthode naïve, la méthode rapide puis la méthode de Hörner.
- ▶ trace les trois courbes `Tn`, `Tr` et `Th` en fonction de N (on prendra $x = 2$). Interpréter le résultat (on pourrait démontrer que les temps d'exécution des trois programmes sont de l'ordre de n^2 pour la méthode naïve (on parle de complexité quadratique), de l'ordre de $n \ln(n)$ pour l'exporapide, et de l'ordre de n pour la méthode de Hörner (complexité linéaire)).