Math93.com

Cours - NSI
Recherche Dichotomique

1 Trouver un nombre

Faire le petit jeu Ici : Un jeu simple pour comprendre la dichotomie

Exercice 1

Écrire le même jeu en python. c'est à dire une fonction **DevinerNombre()** qui tire un nombre au hasard entre 0 et N, demande à l'utilisateur de deviner ce nombre. A chaque fois le programme répond par "plus petit" ou "plus grand" si la proposition est fausse. Il répond "trouvé" si la réponse est bonne. Et renvoi le nombre de coups utilisés.

Code Python

```
# Dans l'éditeur PYTHON
import random
def DevinerNombre(N):
    """
    In: N, un entier int
    Out: nbCoup, un entier int
    """
    ...

    return nbCoup
```

Exercice 2

D'aprés vous qu'elle serait la stratégie la plus efficace pour trouver le nombre à deviner ?

Exercice 3

Maintenant imaginez que l'ordinateur soit schizophrénique. Il joue le rôle de l'utilisateur humain. Transformer le programme précèdent en une fonction en python **DevinerNombreAutomatique(N)** qui tire un nombre au hasard entre 0 et N. Le programme fait comme s'il avait oublié la valeur à trouver et essaye de deviner, avec la stratégie de l'exercice précédent, ce nombre avec des comparaisons strictes si le nombre est différent de la solution (< ou >). A chaque fois le programme affiche les propositions et "plus petit" ou "plus grand" si la proposition est fausse. Il répond "trouvé" si la réponse est bonne. Et renvoi le nombre de coups utilisés.

Vous devez avoir un affichage de ce genre :

```
# Dans la console Python
>>>  DevinerNombreAutomatique(100)
Mon moi propose 50. Mon surmoi dit : plus grand
Mon moi propose 75. Mon surmoi dit : plus grand
Mon moi propose 87. Mon surmoi dit : plus petit
Mon moi propose 81. Mon surmoi dit  : plus petit
Mon moi propose 78. Mon surmoi dit  : trouvé
5
```

Code Python

```
# Dans l'éditeur PYTHON
import random
def DevinerNombreAutomatique(N):
    """
    In: N, un entier int
    Out: nbCoup, un entier int
    """
    ...

    return nbCoup
```

Exercice 4

Enlever les print de la fonction **DevinerNombreAutomatique(N)** et gardez là pour plus tard. On va comparer avec un autre type de recherche. Transformer la fonction **DevinerNombreAutomatique(N)** en une fonction **DevinerNombreBrut(N)** pour qu'il essaye chaque valeur à partir de 1 et renvoi le nombre de coup utilisé.

```
# Dans l'éditeur PYTHON
import random
def DevinerNombreBrut(N):
    """
    In: N, un entier int
    Out: nbCoup, un entier int
    """
    ...

    return nbCoup
```

2 Recherche dichotomique dans une liste triée

2.1 Exemple

La méthode utilisée précédemment pour deviner un nombre entre 0 et N est appelée méthode dichotomique. Nous allons l'utiliser pour faire une recherche dans un tableau trié.
On commence par un exemple (A faire avec des cartons numérotés au tableau) :

Exemple

On va chercher l'élément 15 dans la liste triée [10,12,15,19,20,24,37,54,61] :

10	12	15	19	20	24	37	54	61

On regarde l'élément du milieu c'est 20 > 15. On va donc limiter la recherche à la liste [10,12,15,19].

10	12	15	19	20	24	37	54	61

On regarde l'élément du milieu c'est 12 < 15. On va donc limiter la recherche à la liste [15,19].

10	12	15	19	20	24	37	54	61

On regarde l'élément du milieu c'est 15. On a trouvé l'élément chercher.

Avant d'aborder l'algorithme, regarder l'exemple Recherche Dichotomique en vidéo.

2.2 Algorithme de Recherche Dichotomique

Algorithm 1 rechercheDichotomie(A,x) :

ENTRÉE : A est une liste triée, x une valeur du même type que les éléments de A

```
Si La liste est vide alors
    Retourner FAUX
Sinon
    N = longueur de la liste A
    debut=0
    fin = N − 1
    Tant que debut≤ fin faire
        monIndice = (debut+fin)/2 #division euclidienne
        valeur=A[monIndice]
        Si valeur = x alors
            Retourner VRAI
        Fin Si
        Si valeur < x alors
            debut = monIndice + 1
        Sinon
            fin=monIndice- 1
        Fin Si
    Fin Tant que
Fin Si
Retourner FAUX
```

Exercice 5

Avant d'écrire l'algorithme précédent en python, on construit les tests. La fonction **random.randrange(0, 200)** du module **random** renvoie un entier aléatoire entre 0 et 200. Pour tester votre programme, écrire une fonction **aleaTrier(N)** qui renvoie une liste triée de N entiers compris entre 0 et 200. Vous pouvez utiliser la méthode **sort()** pour trier votre liste.

Code Python

```
# Dans l'éditeur PYTHON
import random                       #Pour faire de l'aléatoire

def aleaTrier(N):
    '''In : N un entier, la taille du tableau
      Out : une liste de N entiers de 0 à 200'''
    return ...
```

Exercice 6

Ecrire l'algorithme de recherche dichotomique en python dans la fonction **rechercheDichotomique(A,x)**. La fonction retourne True si l'élément est dans la liste sinon False. Pour le milieu on utilisera la division euclidienne.

Code Python

```
# Dans l'éditeur PYTHON
def rechercheDichotomique(A,x):
    """
    In: A une liste de int, x un int
    Out: xinA booléen
    """
    ...

    return xinA
```

2.3 Complexité

Exercice 7

Pour une liste de taille 64 et d'une recherche séquentielle, c'est à dire en regardant chaque élément de la liste en partant du premier; combien de comparaison faudrait-il faire? Et en utilisant la dichotomie? Remplissez le tableau suivant pour analyser plus de cas.

Taille de la liste	1	2	4	8	16	32	64	128	N
Recherche séquentielle									
Recherche Dichotomique									$f(N)$

Écrire la première ligne en puissance de 2. Comparer la première et la dernière ligne du tableau. Que remarquez vous?

Remarque

Quand on écrit un nombre N en puissance de 2 (par exemple $512 = 2^9$), la puissance est appelée logarithme en base 2 de N et on la note $log_2(N)$. D'après le tableau précédent la complexité est de l'ordre de $log_2(N)$.

2.4 Algorithme dichotomique en version récursive

Algorithm 2 rechercheDichotomie(A,N,x) :

ENTRÉE : A est une liste triée, N la taille de la liste, x une valeur du même type que les éléments de A

```
Début Procédure
Si La liste est vide alors
    Retourner False
Sinon
    Trouver la valeur la plus centrale de la liste et comparer cette valeur à l'élément x
    Si La valeur est celle cherchée alors
        Retourner True
    Fin Si
    Si La valeur est strictement inférieure à x alors
        Reprendre La procédure avec la seconde moitié de la liste
    Sinon
        Reprendre La procédure avec la première moitié de la liste
    Fin Si
Fin Si
Fin Procédure
```