DICHOTOMIE

Diviser pour régner

Comment rechercher efficacement la présence d'un élément dans un tableau et le localiser ?

In [2]:

```
#L est une liste contenant N entiers choisis au hasard entre 0 et N.
#n est entier dont on est sûr qu'il est dans la liste L
from random import *
N=10**5
L=[randint(0,N) for i in range(N)]
n=choice(L)
print(n)
```

```
62797
```

1. Par balayage

Exercice 1 :

1. `n` est un entier, `L` est une liste. Décrire ce que fait la fonction `recherche(n,L)` ci-dessous :

In [15]:

```
def recherche(n,L):
    i=0
    while i<len(L):
        if L[i] == n:
            return True
        i=i+1

    return False

recherche(n,L)
```

Out[15]:

```
True
```

Réponse :

-
-
-

1. Le modifier pour qu'il renvoie l'index de l'élément recherché, si l'élément est présent et `False` sinon.

In [3]:

```
#Réponse :
def recherche_index(n,L):




    return False

recherche_index(n,L)
```

Out[3]:

```
False
```

1. Quelle est la complexité de cet algorithme ?

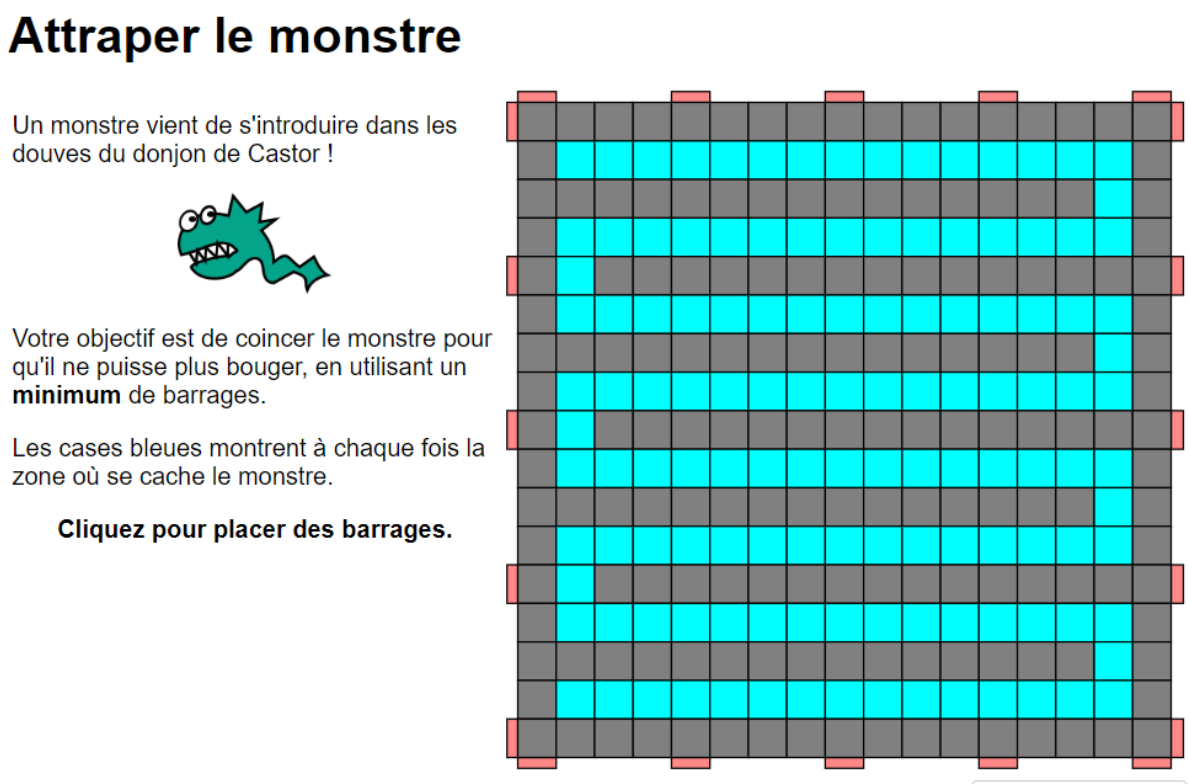
Réponse :

-
-

2. Par recherche dichotomique

Diviser pour régner

Exercice 2:



(http://castor-informatique.fr/questions/2014/2014-FR-03-monster/)

1. Cliquer sur l'image ci-sessus et résoudre l'énigme: Quel est le plus petit nombre de barrages à poser ?

Réponse :

-

1. Quelle stratégie est la plus efficace ?

Réponse :

-
-
-

Exercice 3 :

Exécuter le code du jeu ci-dessous et gagner une partie en le moins de coups possibles en utilisant une recherche par dichotomie.

1. Dans le pire des cas, combien de propositions sont nécéssaires pour gagner en utilisant cette stratégie ?
2. A chaque étape, comment calculer la proposition à formuler ?

In [29]:

```
%%html
<h3 id='h3'>J'ai choisi un nombre entier entre 0 et 128. Deviner ce nombre.</h3>
<form id='form'>
<label for="proposition">Quelle est votre proposition ? </label>
<input type="text" id="proposition" name="proposition"><br>
<input id='button' type="button" value="Valider", onclick=calcul()>
</form>
<h4 id='h4'></h4>
<h4 id='h4-2'></h4>
<h4 id='h4-3'></h4>
<script type="text/javascript" src="devine.js"></script>
```

J'ai choisi un nombre entier entre 0 et 128. Deviner ce nombre.

Quelle est votre proposition ?

Valider

Réponses:

1.
2.

Algorithme

- Pour pouvoir effectuer une recherche dichotomique, **il faut que la liste soit triée**.
- Voici l'écriture d'un algorithme de recherche dichotomique d'un élément dans une liste d'entiers triée en ordre croissant. Si l'élément est présent, son index est renvoyé, sinon le booléen `False` est renvoyé.

(1) L *est une liste d'entiers triée,* n *est un entier*
(2) $g \leftarrow 0$
(3) $d \leftarrow$ *dernier index de la liste*
(4) *Tant que* $g < d$
(5) $\quad m \leftarrow (g + d)//2$
(6) $\quad$ *Si* $L[m] = n$
(7) $\quad\quad$ *Renvoyer* m
(8) $\quad$ *Sinon*
(9) $\quad\quad$ *Si* $L[m] > n$
(10) $\quad\quad\quad d \leftarrow m - 1$
(11) $\quad\quad$ *Sinon*
(12) $\quad\quad\quad g \leftarrow m + 1$
(13) *Renvoyer False*

Exercice 4 :

On considère la liste `L=[2,5,8,11,15,20,32]` et l'entier `n=20` .

1. Compléter la trace de l'algorithme ci-dessous.

ligne	g	d	m
(1)	?	?	?
(2)	0	?	?
(3)	0	6	?
(4)	0	6	?
(5)	0	6	3
(6)	.	.	.
(9)	.	.	.
(12)	.	.	.
(4)	.	.	.
(5)	.	.	.
(6)	.	.	.
(7)	.	.	.

1. Combien de fois la ligne (4) est-elle lue ?

Réponse :

1. Si l'entier cherché est le dernier de la liste `L` , combien de fois la ligne (4) sera-t-elle lue ?

Réponse :

1. Combien de fois la ligne (4) est-elle lue lorsque l'entier cherché n'est pas dans la liste ?

Réponse :

Synthèse :

- Cet algorithme est très efficace, même lorsque l'élément cherché est le dernier de la liste.
- A chaque étape, le nombre d'éléments restant à analyser est divisé par 2.
- Lorsque l'élément cherché n'est pas dans la liste :
  - Les variables `g` et `d` finissent par contenir la même valeur.
  - L'instruction `Tant que g<d` est alors évaluée `False` .
  - On sort ainsi de la boucle et l'algorithme renvoie `False` .

Programmation

Exercice 5 :

En complétant la fonction `dichotomie(n,L)` ci-dessous, où `n` est entier et `L` une liste d'entiers triée par ordre croissant, programmer cet algorithme.

In [17]:

```
#algorithme de recherche dichotomique
def dichotomie(n,L):
    pass

dichotomie(n,sorted(L))
```

Complexité

Exercice 6 :

1. Modifier la fonction de l'exercice 5 pour qu'elle renvoie également le nombre de fois ou l'instruction `while` est lue.

In [ ]:

```
N=10**5
L=[randint(0,N) for i in range(N)]
n=choice(L)

def dichotomie_N(n,L):
    pass

print(n) #entier recherché
dichotomie_N(n,sorted(L)) #position de l'entier et nombre de passages dans la boucle
```

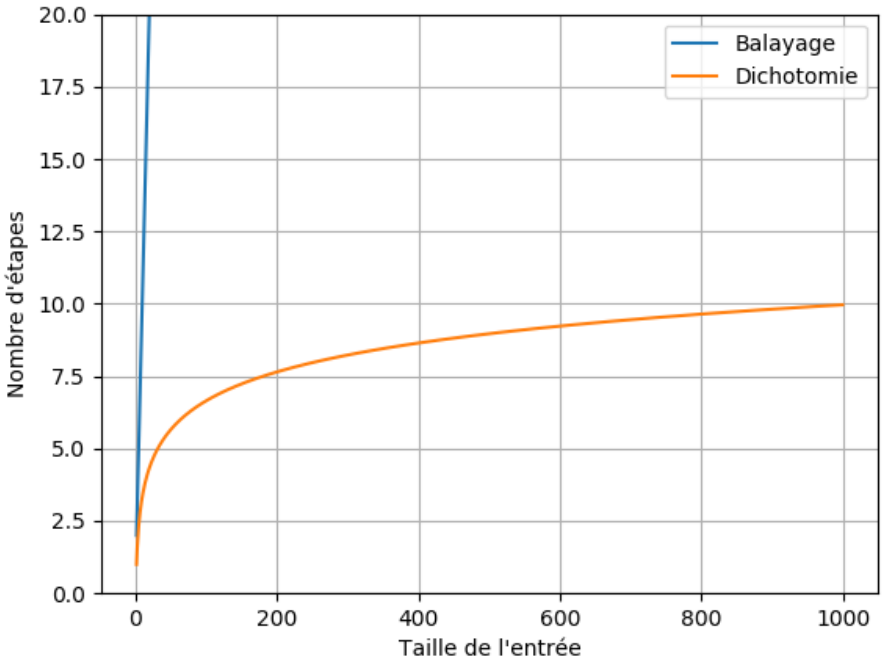
1. Combien d'éléments contient la liste testée? Combien de passages dans la boucle `while` (Exécuter le code plusieurs fois) ?

Réponse :

-
-

Synthèse

- La complexité de l'algorithme de recherche dichotomique est dite logarithmique,c'est à dire de l'ordre du logarithme en base 2 de n (notée $\log_2(n)$,du nom d'une fonction que vous étudierez en terminale) ou n est la taille de l'entrée, c'est à dire ici la longueur de la liste.
- Cela vient du fait que l'on divise par 2 le nombre de possibilités à explorer à chaque étape



- Exemples :
  - Pour une liste de 128 éléments, il faut au maximum $log_2(128) = 7$ étapes.
  - Pour une liste de 1 millions d'éléments, il faut au maximum $log_2(1000000) \approx 20$ étapes.

FIN