|  |  |  |
| --- | --- | --- |
| Numérique et Sciences Informatiques | | |
| 2h30 | **Recherche dichotomique** |  |
| **Objectif** : trouver le plus rapidement possible si un élément recherché existe dans une liste. | | |
| **Matériel**: feuille de papier puis Python | | |

Le prof prépare des nombres écrits chacun sur une petite feuille.

**Jeu** **n°1** : Chaque élève de la classe (sauf un élève maitre) prend un nombre.

Les élèves se mettent en ligne **dans n'importe quel ordre** en montrant leur nombre.

Le maitre va rechercher le nombre 33 en parcourant les nombres en commençant par la gauche.

Déterminez l'algorithme permettant de trouver si un élément recherché existe dans une liste.

**Code 1\*** : Ecrivez la fonction parcours(L,val) qui permet de retourner l'indice de l'élément val dans la liste L.

Si val n'existe pas dans L la fonction retournera la valeur -1.

Pour le test : L=[27,15,10,7,5,12,4,17,20,2,31] et val=17.

**Code 2\*** : Modifiez la fonction parcours(L,val) pour qu'elle compte le nombre de coups réalisés lors de la recherche.

Cette fonction retournera en plus de l'indice le nombre de coups (nombre de tours de boucle).

Pour le test : L=[27,15,10,7,5,12,4,17,20,2,31] et val=17.

**Jeu** **n°2** : Chaque élève de la classe (sauf un élève maitre) prend un nombre.

Les élèves se mettent en ligne **dans n'importe quel ordre** en montrant leur nombre.

Le maitre va rechercher le nombre 12 en utilisant la **méthode dichotomique** (la dichotomie c'est la division de quelque chose en 2 éléments).

Déterminez l'algorithme permettant de trouver si un élément recherché existe dans une liste.

**Code 3\*\*** : Ecrivez la fonction dichotomie(L,val) qui permet de retourner l'indice de l'élément val dans la liste L en utilisant la dichotomie.

Si val n'existe pas dans L la fonction retournera la valeur -1.

Vous pouvez utiliser la méthode sort() pour trier la liste.

Pour le test : L=[27,15,10,7,5,12,4,17,20,2,31] et val=17.

**Code 4\*** : Modifiez la fonction dichotomie(L,val) pour qu'elle compte le nombre de coups réalisés lors de sa recherche.

Cette fonction retournera en plus de l'indice le nombre de coups (nombre de tours de boucle).

Pour le test : L=[27,15,10,7,5,12,4,17,20,2,31] et val=17.

**Code 5\*** : Ecrivez une ligne de code permettant de générer aléatoirement une liste de 64 valeurs comprises entre 1 et 1025 sans qu'apparaisse 2 fois la même valeur.

**Rappel** : création de liste par compréhension : L=[p for p in range(1,1025)]

**Aide** : utilisez la méthode sampe() de la bibliothèque random

**Pour remplir le tableau suivant prenez les programmes corrigés.**

Remplissez le tableau suivant avec des **listes triées** de 64 valeurs générées aléatoirement :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de coups** | Liste 1 : | Liste 2 : | Liste 3 : | Liste 4 : | **Liste 5\* :** |
| parcours(L,val) | 508, 24 | 589, 38 | 808, 52 | 991, 62 |  |
| dichotomie(L,val) | 508, 5 | 589, 4 | 808, 3 | 991, 4 |  |

**\*Liste 5** : faites une recherche dans les 5 premiers éléments de la liste.

* Qu'observez-vous d'après le tableau précédent ?
* Expliquez pourquoi la rechercher dichotomique d'un élément dans une liste trié est presque toujours plus rapide que la recherche par parcours ?

**Activités complémentaires :**

* Remplissez le tableau suivant avec des listes de **n** valeurs générés aléatoirement :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre de coups** | Liste 1 :  64 valeurs | Liste 2 : 128 valeurs | Liste 3 :  256 valeurs | Liste 4 :  512 valeurs | Liste 5 :  1024 valeurs |
| dichotomie(L,val) |  |  |  |  |  |
| log2(n) = |  |  |  |  |  |

**log2(n)** donne le nombre de coups maximum nécessaire dans le pire des cas.

Pour calculer log2(n) en python :

import math

print(math.log2(n))

* Qu'observez-vous d'après le tableau précédent ?
* Quel est le pire des cas, en recherche dichotomique sur L=[2,4,5,7,10,12,15,17,20,27,31] ?
* Ecrivez le jeu du **juste Prix** en utilisant la **dichotomie**.

Le joueur choisira un nombre entre 1 et 100 : **nbJoueur**

Le programme proposera aléatoirement un nombre entre 1 et 100 : **nbProg**

Le programme doit trouver **nbJoueur** en demandant au joueur si **nbProg** est "+", "-" ou "=" au **nbJoueur**

Le programme fera des propositions jusqu'a ce que **nbProg=nbJoueur.**

*Exemple :*

*Le joueur choisi 47.*

*Le programme propose 50, donc le joueur répond "-".*

*Le programme propose 25, donc le joueur répond "+".*

*...*