Recherche dans une liste : séquentielle ou dichotomique ?

Activité – Première NSI

① Le jeu du nombre mystère

Exercice 1 *Deviner un nombre*

Un camarade a choisi secrètement un nombre entier entre 1 et 100. À chaque tentative, il répond uniquement « c'est plus », « c'est moins » ou « trouvé ! ».

1. **Stratégie 1.** Essayer 1, puis 2, puis 3, etc.
   a) Combien de tentatives faut-il au maximum pour trouver le nombre ?
   b) Si le nombre mystère est 73, combien de tentatives faut-il exactement ?
2. **Stratégie 2.** Proposer toujours le milieu de l'intervalle des possibles.
   a) Quel nombre proposez-vous en premier ?
   b) Si la réponse est « c'est plus », quel intervalle reste-t-il à explorer ?
   c) Quel nombre proposez-vous ensuite ?
3. Jouez avec votre voisin(e) en utilisant la stratégie 2. Le nombre mystère est 73. Complétez le tableau ci-dessous.

Essai	Proposition	Réponse	Intervalle restant
1			
2			
3			
4			
5			
6			
7			

4. Combien de tentatives ont été nécessaires ? Comparez avec la stratégie 1.
5. Pourquoi la stratégie 2 est-elle efficace ? Que se passe-t-il à chaque étape pour le nombre de possibilités restantes ?

② Recherche séquentielle dans une liste

Définition 1 **(Recherche séquentielle)** La **recherche séquentielle** (ou recherche linéaire) consiste à parcourir une liste du début à la fin en comparant chaque élément à la valeur cherchée. Elle fonctionne que la liste soit triée ou non.

Exercice 2 *Tracer une recherche séquentielle*

On considère la liste suivante :

17	3	8	24	1	14	9	5	41
0	1	2	3	4	5	6	7	8

1. On cherche la valeur **14**. Indiquez à chaque étape l'indice examiné, la valeur comparée et la décision prise. Combien de comparaisons sont nécessaires ?
2. On cherche la valeur **7**. Combien de comparaisons sont nécessaires ? Quelle est la conclusion ?
3. Pour une liste de n éléments, combien de comparaisons faut-il :
   a) dans le meilleur des cas ?
   b) dans le pire des cas ?

③ Recherche dichotomique dans une liste triée

Définition 2 **(Recherche dichotomique)** La **recherche dichotomique** (du grec *dikhotomia*, « couper en deux ») s'applique à une liste **triée dans l'ordre croissant**. On compare l'élément cherché à celui situé au milieu de la zone de recherche, puis on élimine la moitié qui ne peut pas contenir la valeur cherchée.

Remarque 1. **Condition indispensable :** la liste doit être **triée**. Si elle ne l'est pas, l'algorithme peut donner un résultat **faux**.

Exercice 3 *Tracer une recherche dichotomique*

On considère la liste **triée** suivante :

1	3	5	8	9	14	17	24	41
0	1	2	3	4	5	6	7	8

On note `debut` et `fin` les indices délimitant la zone de recherche, et `centre` l'indice du milieu, calculé par :

$$\texttt{centre} = (\texttt{debut} + \texttt{fin}) \div 2 \quad \text{(division entière)}$$

Recherche de la valeur 14. Complétez le tableau :

Étape	`debut`	`fin`	`centre`	`liste[centre]`	Décision
1	0	8			
2					
3					

Recherche de la valeur 7 (absente). Complétez le tableau :

Étape	`debut`	`fin`	`centre`	`liste[centre]`	Décision
1	0	8			
2					
3					
4					

Comment sait-on que la valeur 7 est absente de la liste ?

④ Comparaison des deux méthodes

Exercice 4 *Comprendre l'efficacité*

1. Complétez le tableau suivant, qui donne le nombre maximal de comparaisons pour chaque méthode en fonction de la taille n de la liste :

Taille n	**1**	**4**	**8**	**16**	**64**	**1 024**	**1 000 000**
Séquentielle	1						
Dichotomique	1						

   Indication : pour la dichotomie, combien de fois peut-on diviser n par 2 avant d'atteindre 1 ?

2. Le moteur de recherche d'une bibliothèque contient 500 000 livres classés par ordre alphabétique. Un bibliothécaire utilise la recherche séquentielle, un autre la recherche dichotomique.

   a) Combien de comparaisons faudra-t-il au maximum pour chaque méthode ?

   b) Si chaque comparaison prend une microseconde (10^{-6} s), quel est le temps de recherche au pire pour chaque méthode ?

3. Pourquoi ne peut-on pas toujours utiliser la recherche dichotomique ? Donnez un exemple concret de situation où seule la recherche séquentielle est possible.

⑤ Écrire l'algorithme

Exercice 5 *Du jeu à l'algorithme*

1. Complétez le pseudo-code de la recherche dichotomique :

```
Fonction recherche_dichotomique(liste, element) :
    debut ← 0
    fin ← longueur(liste) - 1
    Tant que ..............  :
```

```
            centre ← ..............
            Si liste[centre] = element :
                Renvoyer ..............
            Sinon si liste[centre] < element :
                debut ← ..............
            Sinon :
                fin ← ..............
        Renvoyer ..............
```

2. Traduisez ce pseudo-code en Python. Votre fonction `recherche_dichotomique` prend en paramètres une liste triée et un élément, et renvoie `True` ou `False`.
3. Testez votre fonction à la main sur la liste [2 ; 5 ; 11 ; 15 ; 23 ; 30 ; 42] avec :
   - l'élément 23 ;
   - l'élément 10 ;
   - l'élément 2.

⑥ Ce qu'il faut retenir

Propriété 1 **(Complexité de la recherche séquentielle)** La recherche séquentielle dans une liste de n éléments nécessite au plus n comparaisons. Sa complexité dans le pire des cas est en $O(n)$.

Propriété 2 **(Complexité de la recherche dichotomique)** La recherche dichotomique dans une liste **triée** de n éléments nécessite au plus $\lfloor \log_2(n) \rfloor + 1$ comparaisons. Sa complexité dans le pire des cas est en $O(\log_2 n)$.

Remarque 2. La recherche dichotomique est **beaucoup plus efficace** que la recherche séquentielle pour les grandes listes, mais elle impose que la liste soit **triée au préalable**.