Algorithmes

2) Complexité d'un algorithme

Exemple 1

```
s ← 0
pour i allant de 1 à n faire
    s ← s +i
```

1. Exécutez l'algorithme ci-dessus à la main ; présentez les étapes dans un tableau.
   Pour $n = 3$, puis $n = 8$.
2. Que permet d'obtenir cet algorithme ?
3. Combien d'opérations ont été nécessaires pour obtenir un résultat pour $n = 3$, puis $n = 8$? De manière général, pour un entier naturel n quelconque ?
4. Implémentez cet algorithme en Python, puis testez-le pour $n = 1000$, $n = 10^6$, $n = 10^7$ et $n = 10^8$.

3a) Terminaison d'un algorithme

Exemple 2

Version a

```
q ← 0
tant que n >= 3 faire
    q ← q + 1
```

Version b

```
q ← 0
tant que n >= 3 faire
    n ← n - 3
    q ← q + 1
```

1. Exécutez à la main les algorithmes versions a) et b) ci-dessus pour la valeur $n = 10$; présentez les étapes dans un tableau. Que remarquez-vous ?
2. Quelles sont les valeurs des variables q et n à la fin de l'algorithme version b) ? Que représentent ces variables ?
3. Implémentez la version b) de l'algorithme en Python.
4. Dans la version b), comment se comporte la variable n ?

3b) Correction d'un algorithme

Exemple 1

```
s ← 0
Pour i allant de 1 à n faire
    s ← s + i
```

4) Parcours séquentiel d'un tableau

Exemple 3 :

On cherche la présence d'une valeur dans un tableau.

On compare la valeur recherchée successivement à toutes les valeurs du tableau.

L'algorithme s'arrête dès que l'élément est trouvé ou si la fin du tableau est atteinte.

1. Écrivez un algorithme illustrant la méthode présentée ci-dessus.
2. Quel va être la complexité en temps de cet algorithme ?
3. Implémentez l'algorithme en Python.

5a) Tri par sélection d'un tableau

1. Vous allez devoir **trier la liste d'entiers donnée** ci-dessous en suivant le principe décrit. Pour cela, **complétez le tableau** donné pour faire apparaître le différentes étapes.

   Le principe : On parcourt le tableau de la gauche vers la droite, en maintenant sur la gauche une partie déjà triée et à sa place définitive : | déjà trié | pas encore trié |

   À chaque étape, on cherche le plus petit élément dans la partie droite non triée, puis on l'échange avec l'élément le plus à gauche de la partie non triée.

   Ainsi, la première étape va déterminer le plus petit élément et le placer à gauche du tableau. Puis la deuxième étape va déterminer le deuxième plus petit élément et le placer dans la deuxième case du tableau, et ainsi de suite.

5	17	1	22	4	12	31	14

2. **Algorithme du tri par sélection :**
   (a) Écrivez un algorithme qui recherche le plus petit élément d'un tableau et son indice.
   (b) Écrivez un algorithme qui échange un élément d'indice i d'un tableau avec le premier élément de celui-ci.
   (c) Déduisez-en l'algorithme du tri par sélection.

3. Implémentez cet algorithme en Python.

b) Tri par insertion d'un tableau

1. Vous devez **trier la liste d'entiers donnée** ci-dessous en suivant le principe décrit. (**Complétez le tableau** donné pour faire apparaître le différentes étapes.)

   Le principe : On parcourt le tableau de la gauche vers la droite, en maintenant sur la gauche une partie déjà triée et à sa place définitive : | déjà trié | pas encore trié |

   A chaque étape on insère **la première valeur non encore triée**, dans la partie de gauche déjà triée. Pour cela on décale d'une case vers la droite tous les éléments déjà triés qui sont plus grands que la valeur à insérer, puis on place cette valeur dans la case ainsi libérée.

15	6	3	17	8	20	3	11

2. **Algorithme du tri par insertion :**
   (a) Écrivez un algorithme qui permet d'insérer un élément dans un tableau déjà trié.
   (b) Déduisez-en l'algorithme du tri par insertion.

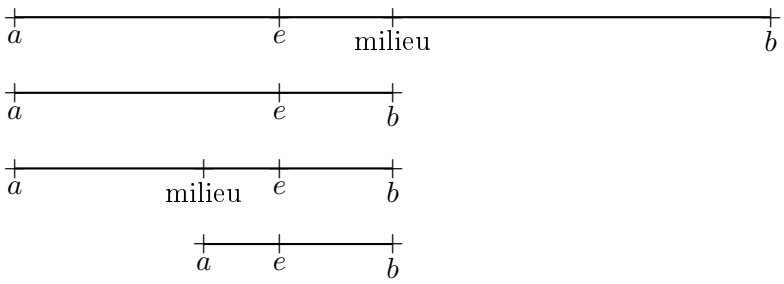
3. Implémentez cet algorithme en Python.

6) Algorithme de Dichotomie

Le fait qu'un tableau soit trié, par exemple par ordre croissant, facilite de nombreuses opérations.

Par exemple, lors de la recherche d'un élément dans un tableau trié, on peut utiliser la **recherche Dichotomique**.

Principe : À chaque étape, on coupe le tableau en deux et on effectue un test pour savoir dans quelle partie se trouve l'élément recherché. On peut ainsi restreindre la zone de recherche par deux à chaque fois. C'est le principe **diviser pour régner**.



On répète ce procédé, jusqu'à obtenir la valeur recherchée, ou un intervalle devenu vide (l'élément n'est pas dans le tableau).

1. Simulez la recherche par dichotomie du nombre 13 dans le tableau $[2, 3, 4, 6, 8, 11, 12, 13, 20, 45, 50]$, en écrivant les différentes étapes.
2. Simulez la recherche par dichotomie du nombre 5 dans le tableau $[2, 3, 4, 6, 8, 11, 12, 13, 20, 45, 50]$, en écrivant les différentes étapes.