Première NSI

|  |  |
| --- | --- |
| NOM  : | Q.C.M Algorithmes 1  10 minutes |
| Prénom  : |
| Classe  : |

A chaque question il y a **une seule bonne réponse**. Cochez la case qui lui correspond.

Une bonne réponse donne 1 point. L’absence de réponse ou une réponse fausse n’enlève pas de point.

1. Comment prouver qu’un algorithme se termine ?
   * Il faut montrer l’existence d’un variant de boucle.
   * Il faut que l’algorithme ait une complexité linéaire.
   * Il faut montrer l’existence d’un invariant de boucle.
   * Il faut que l’algorithme utilise une boucle non bornée.
2. Le premier programme date de :
   * 1800 à 1600 av J.-C.
   * 300 av. J.-C.
   * 1843.
   * 820.
3. Quelle est la particularité d’une boucle «tant que» ?
   * Elle se termine toujours si elle effectue un nombre pair de tours.
   * Elle effectue au minimum un tour de boucle.
   * Elle pourrait ne jamais se terminer.
   * Elle se termine toujours.
4. Prenons le cas d'un tri par sélection d’un tableau d’entiers déjà trié. Pour un tableau de 6 valeurs, le nombre de comparaisons est :
   * 36.
   * 45.
   * 15.
   * 60.
5. Quel est le principe du tri par sélection d’un tableau d’entiers ?
   * Il parcourt le tableau en cherchant l’indice de la valeur la plus petite, puis l'insère à droite de la partie du tableau déjà triée.
   * Il insère chaque valeur du tableau dans une portion de tableau déjà triée.
   * Avant d'insérer la valeur la plus petite, l'algorithme déplace toutes les cartes qui sont à sa gauche.
   * Il parcourt toujours le tableau une seule fois pour le trier.
6. On considère la fonction suivante

def fonction1(liste, a):

    for i in range(len(liste)):

        if liste[i] == a:

            return i

A quoi sert ce programme ?

* + Il renvoie l'indice de la première occurrence de l'élément a dans la liste passée en argument.
  + Il renvoie l'indice de la valeur maximale de la liste passée en argument.
  + Il renvoie le nombre d'occurrences de l'élément a dans la liste passée en argument.
  + Il renvoie l'indice de la dernière occurrence de l'élément a dans la liste passée en argument.

1. L'algorithme d'Euclide date de :
   * 1800 av J.-C.
   * 300 av J.-C.
   * 820.
   * 1843.
2. Quelle est la complexité d’un algorithme de recherche du minimum dans un tableau de entiers ?
   * La complexité de cet algorithme est quadratique en .
   * La complexité de cet algorithme est logarithmique en .
   * Cela dépend de la valeur de .
   * La complexité de cet algorithme est linéaire en .
3. Dans quel cas peut-on utiliser un parcours partiel ?
   * Le comptage du nombre d'occurrences d'une valeur dans un tableau.
   * Le calcul de la moyenne des valeurs d'un tableau.
   * La recherche d'une occurrence dans un tableau.
   * La recherche du maximum ou du minimum dans un tableau de valeurs.
4. Un algorithme ne doit pas être écrit :
   * en pseudo-code.
   * en anglais, avec des phrases.
   * en Python.
   * en français, avec des phrases.
5. Qu’appelle-t-on pseudo-code ?
   * Un code très complexe.
   * Un programme écrit en langage Python.
   * Un code informatique chiffré.
   * Une façon de décrire un algorithme en langage presque naturel.
6. Le tri qui a la meilleure complexité est :
   * le tri par sélection.
   * le tri implémenté par la fonction Python sorted().
   * le tri par insertion.
   * le tri par insertion ou le tri par sélection.
7. Quel est le principe du tri par insertion d’un tableau d’entiers ?
   * Il insère chaque valeur du tableau dans une portion de tableau déjà triée.
   * Il parcourt le tableau en cherchant l’indice de la valeur la plus petite.
   * Il insère la valeur la plus petite au début du tableau.
   * Il parcourt toujours le tableau une seule fois pout le trier.
8. Quelle est la complexité d’un algorithme de tri par sélection d'un tableau de taille ?
   * La complexité de cet algorithme est logarithmique en .
   * La complexité de cet algorithme est linéaire en .
   * La complexité de cet algorithme est quadratique en .
   * Cela dépend de la valeur de .
9. Quelle est la complexité d’un algorithme de calcul de la moyenne d’un tableau de entiers ?
   * Cela dépend de la valeur de .
   * La complexité de cet algorithme est quadratique en .
   * La complexité de cet algorithme est linéaire en ).
   * La complexité de cet algorithme est logarithmique en .
10. Quelle est la complexité d’un algorithme de tri par insertion d'un tableau de taille ?
    * Cela dépend de la valeur de .
    * La complexité de cet algorithme est quadratique en .
    * La complexité de cet algorithme est linéaire en .
    * La complexité de cet algorithme est logarithmique en .
11. On considère la fonction suivante

def sommer(n):

    somme = 0

    p = 1

    while p < n:

        p = p \* 2

        somme = somme + p

    return somme

Quel argument permet d'affirmer que ce programme se termine ?

* + La variable somme croît à chaque tour de boucle.
  + La valeur de décroît à chaque tour de boucle.
  + Une boucle non bornée se termine toujours.
  + La présence d'une seule boucle garantit que le programme se termine.

1. Pour la recherche d'une occurrence dans un tableau de grande taille,
   * Aucune importance, utiliser une boucle bornée ou non bornée revient au même.
   * Il faut privilégier l'utilisation d'une boucle bornée.
   * Il faut privilégier l'utilisation d'une boucle pour.
   * Il faut privilégier l'utilisation d'une boucle non bornée.
2. On considère la fonction suivante

def rechercher(liste, valeur\_cherchee):

    for i in range(len(liste)):

        if liste[i] == valeur\_cherchee:

            return i

Quel est le coût de l'algorithme implémenté par ce programme ?

* + Constant.
  + La complexité de cet algorithme est quadratique en .
  + La complexité de cet algorithme est linéaire en .
  + La complexité de cet algorithme est logarithmique en .

1. Prenons le cas d'un tri par insertion d’un tableau d’entiers déjà trié. Pour un tableau de 8 valeurs, le nombre de comparaisons est :
   * 9.
   * 8.
   * 28.
   * 7.