Algorithmique et Programmation 1 – TD - TP 7

ALGORITHMIQUE

Exercice 1 - Calculs élémentaires de complexité

Soit l'algorithme, exprimé en Python, de recherche séquentielle dans une liste triée vu en cours.

```
1  def recherche_sequentielle_liste_triee(liste, elem):
2      """List x Elem --> Bool
3      Vérifie si l'élément elem appartient à la liste triée"""
4      if elem > liste[len(liste) - 1] :
5          return False
6      else :
7          i = 0
8          while liste[i] < elem :
9              i = i + 1
10         if liste[i] == elem :
11             return True
12         else :
13             return False
```

1. Quelle est la mesure de complexité à utiliser pour évaluer la complexité de cet algorithme ?
2. Calculer la complexité de cet algorithme dans le meilleur cas
3. Calculer la complexité de cet algorithme dans le pire cas
4. Calculer la complexité moyenne de cet algorithme

Exercice 2 - Calculs élémentaires de complexité

Soit l'algorithme suivant, exprimé en Python.

```
1  def mystere(liste, elem):
2      """ """
3      resultat = [0, -1, -1]
4      for i in range(len(liste)):
5          if liste[i] == elem:
6              resultat[0] = resultat[0] + 1
7              resultat[2] = i
8              if resultat[1] == -1:
9                  resultat[1] = i
10     return resultat
```

1. Que fait cet algorithme ? Donnez la chaine de documentation correspondante
2. Quelle est la mesure de complexité à utiliser pour évaluer la complexité de cet algorithme ?
3. Calculer la complexité de cet algorithme dans le meilleur cas
4. Calculer la complexité de cet algorithme dans le pire cas
5. Calculer la complexité moyenne de cet algorithme

Exercice 3 - Recherche de nombres

1. Générer un tableau de 100 nombres aléatoires, tous distincts, compris entre 0 et 1000.
2. Écrire et implémenter l'algorithme permettant de rechercher le nombre maximal de la liste.
3. Écrire une fonction permettant de calculer la moyenne des nombres de la liste.
4. Ecrire une fonction qui renvoie le nombre d'éléments de la liste strictement inférieurs à la moyenne de la liste.
5. Ecrire une fonction qui renvoie la liste des carrés des éléments de la liste.
6. La *variance* d'une liste de nombres est égale à la différence entre la moyenne des carrés des éléments de la liste et le carré de la moyenne des éléments de la liste.
   Ecrire une fonction qui renvoie la variance de la liste.
7. L'*écart-type* d'une liste de nombres est égal à la racine carrée de la variance de la liste.
   Ecrire une fonction qui renvoie l'écart-type de la liste.
8. La méthode `mean` du module prédéfini `numpy` permet de calculer la moyenne d'un tableau [1].
   La méthode `var` permet elle de calculer la variance d'un tableau [2].
   La méthode `std` permet elle de calculer l'écart type d'un tableau [3].
   Comparez les résultats que vous obtenez avec ceux obtenus par ces méthodes.
9. Grâce à la méthode `time()` du module prédéfini `time` [4], donner le temps pour faire les calculs de moyenne avec votre fonction et celles de Numpy.
10. Que conclure sur la complexité temporelle ?
11. La *médiane* d'une liste de nombre entiers tous différents de longueur paire est la moyenne des valeurs centrales de la liste après classement en ordre croissant.
    Par exemple, `mediane([4, 3, 7, 9, 12, 1 ])` = $\frac{4+7}{2}$ = `5,5`.
    Nous voulons calculer la mediane d'une liste, sans avoir à trier cette liste. Pour cela :
    (a) Ecrire une fonction `delta(liste, elem)` qui calcule la différence entre le nombre de valeurs de la liste supérieures et inférieures à `elem`. Par exemple :

```
>>> delta([4,3,7,9,12,1], 3)
3
>>> delta([4,3,7,9,12,1], 4)
1
>>> delta([4,3,7,9,12,1], 7)
-1
```

    (b) Ecrire une fonction qui calcule la médiane d'une liste de longueur paire contenant des entiers tous distincts.
    (c) Comparez votre résultat avec celui obtenu par la méthode `median` du module Numpy [5]
12. Après avoir généré un nombre aléatoire, écrire l'algorithme qui permet de dire que ce nombre est dans le tableau. Implémenter cet algorithme.
13. La méthode `sort` du module prédéfini `numpy` permet de trier un tableau par ordre croissant [6].
    Après avoir trié le tableau, écrire l'algorithme qui permet de dire, en utilisant la recherche séquentielle, si un nombre aléatoire est dans le tableau. Implémenter cet algorithme.
14. Utilisez à présent la recherche dichotomique pour effectuer cette tâche
15. Comparer le temps de calcul de ces trois algorithmes. Que dire de leur complexité ?

1. https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html
2. https://docs.scipy.org/doc/numpy/reference/generated/numpy.var.html
3. https://docs.scipy.org/doc/numpy/reference/generated/numpy.std.html
4. https://docs.python.org/fr/3/library/time.html#module-time
5. https://docs.scipy.org/doc/numpy/reference/generated/numpy.median.html
6. https://docs.scipy.org/doc/numpy/reference/generated/numpy.sort.html