|  |  |  |
| --- | --- | --- |
| Numérique et Sciences Informatiques | | |
| 2h | **Parcours séquentiel d'un tableau** |  |
| **Objectif** : savoir écrire des algorithmes de recherche dans une liste et montrer que son cout est linéaire. | | |
| **Matériel**: feuille de papier et Python | | |

**Recherche d'une occurrence dans une liste**

|  |  |
| --- | --- |
| **Code\*\*** | Ecrivez la fonction recherche\_note(L\_loc,note\_loc) qui recherchera une note comprise entre 0 et 20 dans la liste L de dimension n.  Cette fonction retournera **True** si la note est présente et **False** dans le cas contraire. |
| **Exemple** | L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16] et Note 🡨 12 alors Trouvé 🡨 True  L 🡨 [5, 14, 18, 11, 10, 10, 9, 16] et Note 🡨 12 alors Trouvé 🡨 False |
| **Pré-condition sur argument(s)** | Les valeurs dans la liste L sont comprises entre 0 et 20 inclus.  La liste L ne doit pas être vide.  La valeur de la note est comprise entre 0 et 20. |
| **Post-condition sur résultat(s)** | Un booléen est retourné (True si la note est présente dans le tableau, sinon False) |
| **Nb d'instructions exécutées** | Proportionnel au nombre n d’éléments dans la liste.  Coût linéaire en O(n) |

|  |
| --- |
| **Algorithme en pseudo-code : solution où on parcourt toute la liste** |
| Fonction recherce\_note(L\_loc, note\_loc)  Trouvé 🡨 False  Pour indice de 0 à longueur(L\_loc) - 1 faire  Si L\_loc[indice] = note\_loc faire  Trouvé 🡨 True  Fin si  Fin pour  Retourné trouvé  L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16]  Note 🡨 12  Résultat 🡨 recherche\_note(L, note) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Résultats du test :** avec la liste : L=[5,14,18,11,10,12,10,9,16] et la note à rechercher = 12 | | | | |
| **Étape** | **Note à**  **chercher** | **Variable :**  **"indice"** | **Liste[indice]** | **Variable :**  **"trouvé"** |
| **Avant de rentrer dans la boucle** | 12 | x | x | Faux |
| **Dans la boucle (1ère itération)** | 12 | 0 | 5 | Faux |
| **Dans la boucle (2ème itération)** | 12 | 1 | 18 | Faux |
| **Dans la boucle (3ème itération)** | 12 | 2 | 10 | Faux |
| **Dans la boucle (4ème itération)** | 12 | 3 | 12 | Vrai |
| **Dans la boucle (5ème itération)** | 12 | 4 | 10 | Vrai |
| **Dans la boucle (6ème itération)** | 12 | 5 | 14 | Vrai |
| **Dans la boucle (7ème itération)** | 12 | 6 | 18 | Vrai |
| **Dans la boucle (8ème itération)** |  |  |  |  |
| **En sortie de boucle** | 12 | 6 | 18 | Vrai |

|  |
| --- |
| **Algorithme en pseudo-code : solution où on arrête le parcours de la liste lorsque l'occurrence est trouvée.** |
| Fonction recherche\_note(L\_loc, note\_loc)  Trouvé 🡨 False  Indice 🡨 0  Tant que L\_loc[Indice] différent de note\_loc faire  Indice 🡨 Indice + 1  Trouvé 🡨 True  Renvoyé Trouvé  L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16]  Note 🡨 12  Résultat 🡨 recherche\_note(L, Note) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Résultats du test :** avec la liste : L=[5,14,18,11,10,12,10,9,16] et la note à rechercher = 12 | | | | |
| **Étape** | **Note à**  **chercher** | **Variable :**  **"indice"** | **Liste[indice]** | **Variable :**  **"trouvé"** |
| **Avant de rentrer dans la boucle** | 12 | 0 | x | Faux |
| **Dans la boucle (1ère itération)** | 12 | 0 | 5 | Faux |
| **Dans la boucle (2ème itération)** | 12 | 1 | 14 | Faux |
| **Dans la boucle (3ème itération)** | 12 | 2 | 18 | Faux |
| **Dans la boucle (4ème itération)** | 12 | 3 | 11 | Faux |
| **Dans la boucle (5ème itération)** | 12 | 4 | 10 | Faux |
| **Dans la boucle (6ème itération)** | 12 | 5 | 12 | Vrai |
| **En sortie de boucle** | 12 | 5 | 12 | Vrai |

**Recherche d'un extremum dans une liste**

|  |  |
| --- | --- |
| **Code\*\*** | Ecrivez la fonction note\_max(L\_loc) qui recherche la valeur maximale dans la liste L\_loc de dimension n.  Cette fonction retournera la valeur maximale. |
| **Exemple** | L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16] alors Max 🡨 18 |
| **Pré-condition sur argument(s)** | Les valeurs dans la liste L sont comprises entre 0 et 20 inclus.  La liste L ne doit pas être vide. |
| **Post-condition sur résultat(s)** | Un entier est retourné (la valeur max, comprise entre 0 et 20) |
| **Nb d'instructions exécutées** | Proportionnel au nombre n d’éléments dans la liste.  Coût linéaire en O(n) |

|  |
| --- |
| **Algorithme en pseudo-code** |
| Fonction note\_max(L\_loc)  Max 🡨 L\_loc[0]  Pour indice de 0 à longueur(L\_loc) - 1 faire  Si Max < L\_loc[indice] faire  Max 🡨 L\_loc[indice]  Fin si  Fin pour  Renvoyé Max  L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16]  Résultat 🡨 note\_max(L) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Résultats du test :** avec la liste : L=[5,14,18,11,10,12,10,9,16] | | | |
| **Étape** | **Variable :**  **"indice"** | **Liste[indice]** | **Variable :**  **"max"** |
| **Avant de rentrer dans la boucle** | x | x | 5 |
| **Dans la boucle (1ère itération)** | 0 | 5 | 5 |
| **Dans la boucle (2ème itération)** | 1 | 14 | 14 |
| **Dans la boucle (3ème itération)** | 2 | 18 | 18 |
| **Dans la boucle (4ème itération)** | 3 | 11 | 18 |
| **Dans la boucle (5ème itération)** | 4 | 10 | 18 |
| **Dans la boucle (6ème itération)** | 5 | 12 | 18 |
| **Dans la boucle (7ème itération)** | 6 | 10 | 18 |
| **Dans la boucle (8ème itération)** | 7 | 9 | 18 |
| **Dans la boucle (9ème itération)** | 8 | 16 | 18 |
| **En sortie de boucle** | 8 | 16 | 18 |

**Calcul de la valeur moyenne des valeurs d'une liste**

|  |  |
| --- | --- |
| **Code\*** | Ecrivez la fonction moy\_notes(L\_loc) qui calculera la moyenne des notes de la liste L\_loc de dimension n.  Cette fonction retournera la moyenne. |
| **Exemple** | L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16] alors moyenne 🡨 11.7 |
| **Pré-condition sur argument(s)** | Les valeurs dans la liste L sont comprises entre 0 et 20 inclus.  La liste L ne doit pas être vide. |
| **Post-condition sur résultat(s)** | Un décimal est retourné (la moyenne, comprise entre 0 et 20) |
| **Nb d'instructions exécutées** | Proportionnel au nombre n d’éléments dans la liste.  Coût linéaire en O(n) |

|  |
| --- |
| **Algorithme en pseudo-code** |
| Fonction moy\_notes(L\_loc) :  Pour indice de 0 à longueur(L\_loc) – 1 faire  Moyenne 🡨 (moyenne + L\_loc[indice]d) / (indice + 1)  Retourné Moyenne  L 🡨 [5, 14, 18, 11, 10, 12, 10, 9, 16]  Résultat 🡨 moy\_notes(L\_loc) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Résultats du test :** avec la liste : L=[5,14,18,11,10,12,10,9,16] | | | |  |
| **Étape** | **Variable :**  **"indice"** | **Liste[indice]** | **Variable :**  **"somme"** | **Variable : "moyenne"** |
| **Avant de rentrer dans la boucle** | x | x | 0 | x |
| **Dans la boucle (1ère itération)** | 0 | 5 | 5 | 5 |
| **Dans la boucle (2ème itération)** | 1 | 14 | 19 | 9.5 |
| **Dans la boucle (3ème itération)** | 2 | 18 | 37 | 12.3 |
| **Dans la boucle (4ème itération)** | 3 | 11 | 48 | 12 |
| **Dans la boucle (5ème itération)** | 4 | 10 | 58 | 11.6 |
| **Dans la boucle (6ème itération)** | 5 | 12 | 70 | 11.7 |
| **Dans la boucle (7ème itération)** | 6 | 10 | 80 | 11.4 |
| **Dans la boucle (8ème itération)** | 7 | 9 | 89 | 11.1 |
| **Dans la boucle (9ème itération)** | 8 | 16 | 105 | 11.7 |
| **En sortie de boucle** | 8 | 16 | 105 | 11.7 |