|  |  |  |
| --- | --- | --- |
|  | **Complexité algorithmique**  **TP2 : Tri par sélection** | logo touchard |

LE CREN Anthony

**Objectifs :**

- Comprendre le tri par sélection.

- Déterminer la complexité du tri par sélection de manière expérimentale.

|  |  |
| --- | --- |
| Pour s’y retrouver plus facilement.  En effet si le matériel était rangé, on perdrait moins de **temps** à le rechercher. |  |
| Imaginez une bibliothèque avec les livres dans le **désordre** et ce que cela implique… |  |
| Si les prénoms ci-contre étaient triés par ordre alphabétique, cela serait **plus facile** d’accéder à leur numéro de téléphone ou leur adresse. | Justine Christopher  Mathis  Michaël Emma Gabriel  Léo Raphaël  Olivia Sofia  Ludovic Cédric  Antoine Coralie |
| On va s’intéresser au tri de nombres entiers positifs dans un tableau. Il existe plusieurs méthodes de tris. (**Sélection**, insertion, fusion, etc…). Chaque méthode présente des performances différentes. | Chiffres dans le désordre :  45, 71, 56, 12, 54, 47, 51  Chiffres triés par ordre croissant :  **12, 45, 47, 51, 54, 56, 71** |

**Partie 1 : Introduction au tri par sélection.**

|  |  |
| --- | --- |
| **Q1** Regardez la vidéo suivante :  <https://www.youtube.com/watch?v=qpeeRU_K90k>  Par binôme, reproduire le tri en utilisant un jeu de 54 cartes. |  |
|  |

Description du principe de fonctionnement avec de nouveaux nombres à trier :

**Etape 1** : Sélection de la 1ere valeur dans le tableau, celle-ci sera utilisée comme référence.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | 47 | 18 | 8 | 21 | 34 |
| Indice | 0 | 1 | 2 | 3 | 4 |

**Etape 2** : Recherche de la plus petite valeur dans la partie droite du tableau < 47.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | 47 | 18 | 8 (min) | 21 | 34 |
| Indice | 0 | 1 | 2 | 3 | 4 |

**Etape 3** : Permutation de la valeur à l’indice 0 (référence) avec la plus petite valeur trouvée précédemment.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | 8 | 18 | 47 | 21 | 34 |
| Indice | 0 | 1 | 2 | 3 | 4 |

**Etape 4** : Sélection de la valeur à l’indice 1 dans le tableau, celle-ci sera utilisée comme référence.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | 8 | 18 | 47 | 21 | 34 |
| Indice | 0 | 1 | 2 | 3 | 4 |

**Etape 5** : Recherche de la plus petite valeur dans la partie droite du tableau < 18.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre | 8 | 18 | 47 | 21 | 34 |
| Indice | 0 | 1 | 2 | 3 | 4 |

Il n’y en a pas, donc pas de permutation, On sélectionne la valeur à l’indice 2 dans le tableau, on recherche un minimum dans la partie droite et si c’est le cas on permute les valeurs.

**Partie 2 : Elaboration de l’algorithme.**

N= longueur(tableau)

fonction tri\_Selection (tableau)

pour i de 0 à N-2

indmini←i

trouver l’indice indmini le plus petit élément du tableau dans l’intervalle i+1,N-1

si i≠indmini alors

échanger le contenu des indices i et indmini

finsi

finpour

Exemple pour la valeur sélectionnée à l’indice 0 d’un tableau N de 5 éléments :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Référence | Ou se trouve dans le reste du tableau, le nombre le plus petit et inférieur à la valeur de référence 47 ? | | | |
| Nombre | 47 | 18 | 8 | 21 | 34 |
| Indice | 0 (i) | 1 | 2 (indmini) | 3 | 4 |

**Q2** Elaborer l’algorithme permettant de trouver l’indice indmini le plus petit élément du tableau dans l’intervalle i+1,N-1.

Pour j de i+1 à N‐1 faire

Si tableau[j] < tableau[indmini] alors

indmini←j

Fin Si

Fin Pour

**Q3** Elaborer l’algorithme permettant d’échanger le contenu des indices i et indmini.

Astuce : Pour permuter le contenu de deux seaux d’eau, il faut forcément un 3éme seau vide que l’on appellera temp (abréviation de variable temporaire).

Temp←tableau[i]

tableau[i]←tableau[indmini]

tableau[indmini]←temp

**Q4** Réunir les 3 parties de l’algorithme.

fonction tri\_Selection (tableau)

pour i de 0 à N-2

indmini←i

pour j de i+1 à N‐1 faire

si tableau[j] < tableau[indmini] alors

indmini←j

fin Si

fin Pour

si i≠indmini alors

temp←tableau[i]

tableau[i]←tableau[indmini]

tableau[indmini]←temp

finsi

finpour

**Q5** Traduire l’algorithme de la fonction tri\_Selection en python.

**import** time  
**import** random  
  
N=5 *#longueur du tableau*VALEUR\_MAX=50 *#valeur maximale pouvant être contenue dans le tableau***def** creationValeursNonTrie(N): *#fonction de génération d'un tableau de valeurs aléatoires* tableau=[]  
 **for** indice **in** range(0,N):  
 tableau.append(random.randint(1,VALEUR\_MAX))  
 **return** tableau  
  
**def** tri\_Selection(tableau): *#fonction de tri par sélection à compléter* **for** i **in** range(0, N - 1):  
 indmini = i  
 **for** j **in** range(i + 1, N):  
 **if** (tableau[j] < tableau[indmini]):  
 indmini = j  
 **if** (i != indmini):  
 temp = tableau[i]  
 tableau[i] = tableau[indmini]  
 tableau[indmini] = temp  
 print(tableau[indmini], **'<-->'**, tableau[i], **':'**, tableau)  
  
  
tab=creationValeursNonTrie(N) *#génération d'un tableau de valeur aléatoire*print(**'tableau non trié'**,tab)  
debut = time.time() *#mémorisation de l'heure de début*tri\_Selection(tab)  
fin = time.time() *#mémorisation de l'heure de fin*print(**"longueur tableau \t durée de traitement"**)  
print(N,**'\t'**,fin-debut)  
print(**'tableau trié'**,tab)

**Partie 3 : Mesure du temps de traitement**

Exemple de résultat affiché après exécution du programme python.

tableau non trié [8, 47, 38, 19, 19]

47 <--> 19 : [8, 19, 38, 47, 19]

38 <--> 19 : [8, 19, 19, 47, 38]

47 <--> 38 : [8, 19, 19, 38, 47]

longueur tableau durée de traitement

5 0.0

tableau trié [8, 19, 19, 38, 47]

On remarque que la durée de traitement est nulle (0.0). En réalité, elle est très faible car la longueur du tableau est seulement de 5 valeurs (N=5).

**Q6** Compléter le tableau suivant en faisant varier la longueur du tableau N en utilisant toujours VALEUR\_MAX=100000.

Commenter la ligne suivante (ajouter un # au début de la ligne)

*#print(tableau[indmini], '<-->', tableau[i], ':', tableau)*

|  |  |
| --- | --- |
| **Nombre de données N à trier** | **Temps de traitement** |
| 1000 | 0,124800205 |
| 2000 | 0,499201059 |
| 4000 | 2,230803967 |
| 8000 | 8,580014706 |
| 10000 | 12,6516223 |
| 20000 | 51,58928967 |

**Q7** A l’aide d’un tableur, tracer la courbe du Temps de traitement en fonction du nombre de données à trier (Utiliser le fichier simulation.xlsx)

Corrigé ci-dessous (temps de traitement variable d’une machine à une autre)

**Q8** Conclure en utilisant le document ressource (TP algorithmes\_complexité-ressource.docx)

**Q9** Compléter la fiche résumée en **annexe, page 6**

**Annexe : Tri par sélection**

|  |  |
| --- | --- |
| **Énoncé** | Trier un tableau de valeurs dans l’ordre croissant. |
| **Exemple** | tableau non trié [47, 8, 38, 19, 20]  tableau trié [8, 19, 20, 38, 47] |
| **Solution** | **Utiliser la méthode de tri par sélection.** |
| **Entrée** | **Tableau de valeurs non triées.** |
| **Sortie** | **Tableau de valeurs triées.** |
| **Pré-condition** | **Le tableau de valeurs est de taille N. Les valeurs sont entières et positives. Le tableau possède au moins 2 valeurs.** |
| **Post-condition** | **tableau[i] ≤ tableau[i+1]. Avec 0 ≤ i < N-1** |

|  |  |
| --- | --- |
| **Algorithme** | **Programme en python** |
| fonction tri\_Selection (tableau)  pour i de 0 à N-2  indmini←i  pour j de i+1 à N‐1 faire  si tableau[j] < tableau[indmini] alors  indmini←j  fin Si  fin Pour  si i≠indmini alors  temp←tableau[i]  tableau[i]←tableau[indmini]  tableau[indmini]←temp  finsi  finpour | *#N est la longueur du tableau*  **def** tri\_Selection(tableau)  **for** i **in** range(0, N - 1):  indmini = i  **for** j **in** range(i + 1, N):  **if** (tableau[j] < tableau[indmini]):  indmini = j  **if** (i != indmini):  temp = tableau[i]  tableau[i] = tableau[indmini]  tableau[indmini] = temp |

**Complexité de l’algorithme :** O(n²) car pour chaque indice de référence, il faut parcourir tout le reste du tableau

**Résultats du test :**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Indice dans le tableau** | | | **0** | **1** | **2** | **3** | **4** |
| **Étape en pseudo-code** | **i** | **indmin** | **47** | **8** | **38** | **19** | **20** |
| **Avant de rentrer dans la boucle** | X | X | **47** | **8** | **38** | **19** | **20** |
| **Dans la boucle (1ère itération)** | **0** | **1** | **8** | **47** | **38** | **19** | **20** |
| **Dans la boucle (2ème itération)** | **1** | **3** | **8** | **19** | **38** | **47** | **20** |
| **Dans la boucle (3ème itération)** | **2** | **4** | **8** | **19** | **20** | **47** | **38** |
| **Dans la boucle (4ème itération)** | **3** | **4** | **8** | **19** | **20** | **38** | **47** |

**Conclusion :** L’algorithme se termine en un temps fini et produit la sortie désirée (Tri des valeurs dans l’ordre croissant).