Autour des algorithmes de tri : Le tri par sélection

Plan de l'activité :

I- Mise en situation

Les élèves manipulent pendant 15 min, l'enseignant fait émerger des stratégies, par exemple en examinant les différents nombres d'opérations.

But : mettre les élèves immédiatement en activité sur un thème délicat au travers d'une activité stimulante.

On va certainement voir arriver la méthode intuitive du tri par sélection

II- Le tri par sélection

Algorithme explicitement au programme avec preuve et complexité.

A la suite de l'activité précédente, on montre les barres et l'animation pour les trier.

Un exemple vaut mieux qu'un long discours et ils doivent comprendre le principe (min en premier puis recherche du min sur les n-1 restant….).

Si le principe n'est pas bien perçu, reprise collective et mise au point avant de passer à la suite.

III- Implémentation python

Terminé le jeu, maintenant on programme !

Demander d'implémenter l'algo en python.

1) Ecrire une fonction **tableau_hasard(n , i , j)** qui permet de générer un tableau de taille **n** dont les indices sont pris au hasard entre **i** et **j**.
2) Ecrire une fonction **echange (T , i , j)** qui permet d'échanger les termes d'indices **i** et **j** d'un tableau **T**.
3) Ecrire l'algo complet (éventuellement donner un algo incomplet et demander de boucher les trous, à voir en fonction des réactions des élèves, c'est une partie qui peut être délicate pour eux).
4) Faire tourner sur quelques exemples, le pas à pas peut être pertinent pour certains.

IV- Le problème de la complexité

Expliquer aux élèves que pour des grandes valeurs de n (cf génome) ce problème est crucial :

- **Présentation de la courbe montrant la complexité du tri par sélection**

1) Reprendre l'ex du tonneau vierge (sans chiffre) demander le nbr d'opérations pour 5 tonneaux puis 6
2) Ensuite retour sur le python, demander de créer un compteur **nbr_tests** et vérifier le programme pour n = 5 et n = 6

- **On remarque que le nombre de tests ne dépend que de n.**
  Essayer de trouver une formule.

3) La tester pour 10, 100.
4) Que se passe-t-il si n double ??? (Faire émerger la complexité quadratique)
5) Démonstration cas général : somme arithmétique en regardant chaque étage de la première boucle : $n-1+n-2+\ldots+1=n(n-1)/2=O(n^2)$
6) Faire remarque que, à taille de tableau donné, la complexité est constante et ne dépend en rien du fait que le tableau soit plus ou moins trié (ce ne sera pas le cas pour l'insertion ou pour un tableau trié on peut avoir une complexité en O(n).
   Vérifier avec le programme python en testant sur un tableau de 10 éléments déjà triés par exemple.

V- invariant : preuve de correction et terminaison :

1- Définition générale d'un invariant de boucle dans un algorithme.

On appelle invariant d'une boucle une propriété qui, si elle est vraie avant l'exécution d'une itération, le demeure après l'exécution de l'itération.

2- Tri par sélection.

On va travailler sur la liste L=[3, 1, 7, 9, 4, 12, 5] que l'on cherche à trier par sélection.

Nous ne reviendrons pas ici sur le principe d'un tel tri. Voici, en revanche, son exécution après les différentes étapes de la boucle « pour » :

	L=[3, 1, 7, 9, 4, 12, 5]
Après l'étape 1	L=[1, 3, 7, 9, 4, 12, 5]
Après l'étape 2	L=[1, 3, 7, 9, 4, 12, 5]
Après l'étape 3	L=[1, 3, 4, 9, 7, 12, 5]
Après l'étape 4	L=[1, 3, 4, 5, 7, 12, 9]
Après l'étape 5	L=[1, 3, 4, 5, 7, 12, 9]
Après l'étape 6	L=[1, 3, 4, 5, 7, 9, 12]

Après observé ce tableau, on choisit comme invariant de boucle la propriété : « après l'exécution de l'étape i, les i premiers termes de liste, que Python note L[0 :i], sont triés ».

- Après une première itération de la boucle « pour » (c'est-à-dire $i=1$), la liste L[0 : i] =L[0] qui ne comporte qu'un seul élément, est donc triée.
- Après l'exécution de l'étape k, on suppose les k premiers termes de la liste sont triés, c'est-à-dire que la liste L[0 :k] est triée.
  On va montrer que après l'étape $k+1$, les $k+1$ premiers termes seront triés également.
  Après l'exécution de l'étape $k+1$, la liste L[0 : $k+1$], obtenue en concaténant la liste L[0 : k] avec l'élément minimal de la liste des restants (qui reste supérieure aux k premiers), est donc triée aussi.
- La boucle se termine après l'étape 6 : les 6 premiers termes triés L[0 :6] et le dernier L[6] sont donc classés dans l'ordre croissant. Le tableau d'origine est donc trié : l'algorithme se finit et est correct.

3- Terminaison.

Le nombre d'éléments triés augmente de 1 à chaque étape et est majoré par la longueur de la liste, ce qui prouve la terminaison de l'algorithme.

VI- Prolongements :

Faire sentir la médiocrité du $O(n^2)$ pour de grandes valeurs et donner l'exemple du jeu de carte avec le tri fusion comme on a vu en cours pour faire sentir l'existence d'algo plus performant.