Trier une liste : tri par sélection et tri par insertion

Activité – Première NSI

① Trier à la main

Exercice 1 *Deux façons de trier des cartes*

Vous avez devant vous les cartes suivantes, posées face visible sur la table dans cet ordre :

12	9	3	7	14	11
pos. 0	pos. 1	pos. 2	pos. 3	pos. 4	pos. 5

Méthode A : à chaque étape, repérez la plus petite carte parmi celles qui restent et échangez-la avec la première carte non encore placée.

Méthode B : prenez les cartes une par une (de gauche à droite) et insérez chacune à sa place parmi les cartes déjà en main.

1. Effectuez le tri avec la méthode A. À chaque étape, notez quelle carte est sélectionnée comme minimum et avec quelle carte elle est échangée.
2. Effectuez le tri avec la méthode B. À chaque étape, notez quelle carte est insérée et combien de décalages sont nécessaires.
3. Laquelle de ces deux méthodes utilisez-vous spontanément pour trier un jeu de cartes ?

② Tri par sélection

Définition 1 **(Tri par sélection)** Le **tri par sélection** consiste, à chaque étape i, à *sélectionner* le plus petit élément de la partie non triée (indices i à $n-1$) et à l'échanger avec l'élément d'indice i.

Exercice 2 *Tracer le tri par sélection*

On considère la liste [15 ; 8 ; 3 ; 11 ; 2 ; 7].

Pour chaque étape, indiquez le minimum trouvé dans la partie non triée, l'échange effectué, puis l'état de la liste. Complétez le tableau :

Étape i	Min trouvé	Échange effectué	Liste résultante
0			
1			
2			
3			
4			

Exercice 3 *Écrire l'algorithme du tri par sélection*

1. Complétez le pseudo-code :

```
Fonction tri_selection(liste) :
    Pour i allant de 0 à longueur(liste) - 2 :
        ind_min ← ..............
        Pour j allant de .............. à longueur(liste) - 1 :
            Si liste[j] < liste[ind_min] :
                ind_min ← ..............
        Échanger liste[i] et liste[..............]
```

2. Combien de comparaisons effectue la boucle intérieure lors de l'étape $i = 0$ pour une liste de n éléments ? Lors de l'étape $i = 1$? Lors de l'étape i quelconque ?
3. En déduire le nombre total de comparaisons. On rappelle que $1 + 2 + \cdots + k = \dfrac{k(k+1)}{2}$.
4. Ce nombre dépend-il de l'ordre initial de la liste ?

③ Tri par insertion

Définition 2 **(Tri par insertion)** Le **tri par insertion** consiste, à chaque étape i, à prendre l'élément d'indice i et à l'*insérer* à sa bonne place parmi les éléments déjà triés (indices 0 à $i-1$), en décalant vers la droite les éléments plus grands.

Remarque 1. C'est la méthode que l'on utilise naturellement pour trier un jeu de cartes : on prend les cartes une par une et on les range à leur place dans la main.

Exercice 4 *Tracer le tri par insertion*

On considère la même liste [15 ; 8 ; 3 ; 11 ; 2 ; 7]. Complétez le tableau :

Étape i	Élément à insérer	Partie déjà triée	Nb de décalages	Liste résultante
1	8	[15]		
2	3			
3	11			
4	2			
5	7			

Indication : pour chaque étape, recopiez la partie triée, puis déterminez combien d'éléments doivent être décalés vers la droite pour faire de la place à l'élément à insérer.

Exercice 5 *Écrire l'algorithme du tri par insertion*

1. Complétez le pseudo-code :

```
Fonction tri_insertion(liste) :
    Pour i allant de 1 à longueur(liste) - 1 :
```

```
        en_cours ← liste[i]
        j ← ..............
        Tant que j > 0 ET liste[j - 1] > ..............  :
            liste[j] ← ..............   (décalage)
            j ← ..............
        liste[j] ← ..............   (insertion)
```

2. **Meilleur cas.** Combien de comparaisons sont effectuées si la liste est déjà triée ? Justifiez.
3. **Pire cas.** Combien de comparaisons sont effectuées si la liste est triée dans l'ordre décroissant ? Justifiez.
4. Que peut-on dire de l'efficacité du tri par insertion sur une liste « presque triée » ?

4 Comparaison des deux algorithmes

Exercice 6 *Compter les opérations*

On trie la liste [5 ; 1 ; 4 ; 2 ; 3] de taille $n = 5$.

1. Effectuez le tri par sélection et comptez le nombre total de **comparaisons** et d'**échanges**.
2. Effectuez le tri par insertion et comptez le nombre total de **comparaisons** et de **décalages**.
3. Refaites le même exercice avec la liste déjà triée [1 ; 2 ; 3 ; 4 ; 5]. Que constatez-vous ?
4. Refaites le même exercice avec la liste triée en ordre décroissant [5 ; 4 ; 3 ; 2 ; 1].
5. Résumez vos observations dans le tableau suivant :

	Liste quelconque	**Liste déjà triée**	**Liste inversée**
Sélection (comparaisons)			
Insertion (comparaisons)			

5 Tri stable ou non stable ?

Définition 3 **(Tri stable)** Un tri est dit **stable** s'il préserve l'ordre relatif des éléments considérés comme égaux par la relation d'ordre.

Exercice 7 *Stabilité des tris*

On trie la liste suivante par **valeur numérique**. Les indices a et b permettent de distinguer les deux éléments égaux à 4 :

$$[9 \ ; \ 4_a \ ; \ 5 \ ; \ 4_b \ ; \ 1 \ ; \ 3]$$

1. Triez cette liste par **sélection**. Notez l'ordre dans lequel apparaissent 4_a et 4_b dans le résultat.
2. Triez la même liste par **insertion**. Notez l'ordre de 4_a et 4_b.
3. Lequel des deux algorithmes est stable ? Justifiez.

⑥ Ce qu'il faut retenir

Propriété 1 **(Complexité des tris naïfs)** Le tri par sélection et le tri par insertion ont une complexité dans le pire des cas en $O(n^2)$. Le nombre total de comparaisons est de l'ordre de $\frac{n^2}{2}$.

Propriété 2 **(Avantage du tri par insertion)** Contrairement au tri par sélection, le tri par insertion est **sensible à l'ordre initial** : sur une liste déjà triée ou presque triée, sa complexité descend à $O(n)$ (linéaire).

Algorithme	Meilleur cas	Pire cas	Stable	En place
Tri par sélection	$O(n^2)$	$O(n^2)$	Non	Oui
Tri par insertion	$O(n)$	$O(n^2)$	Oui	Oui