TP 2: Étude Expérimentale des Algorithmes de Tri Simples

Ce TP est dédié à l'étude expérimentale des algorithmes de tri simples (tri par sélection, tri à bulles, tri par insertion avec deux variantes). Voici une approche structurée que vous pourriez utiliser pour guider les étudiants dans la comparaison des algorithmes :

1. Objectifs pédagogiques

- Implémenter les algorithmes de tri simples vus en TDs.
- Comparer les performances de ces algorithmes en mesurant le nombre de comparaisons, le nombre de déplacements, et le temps d'exécution CPU.
- Analyser l'impact de la disposition initiale des données (aléatoire, trié croissant, trié décroissant).

Voici un rappel des algorithmes :

- **Tri par sélection** : Trouve le minimum dans la partie non triée du tableau et l'échange avec le premier élément de cette partie.
- **Tri à bulles** : Compare chaque paire d'éléments adjacents et les échange si nécessaire, en répétant le processus jusqu'à ce que le tableau soit trié.
- **Tri par insertion (échanges)** : Insère chaque élément dans sa position appropriée en utilisant des échanges successifs.
- **Tri par insertion (déplacement)** : Déplace les éléments plus grands pour faire de la place pour l'insertion directe d'un nouvel élément.

2. Préparation des données de test

- **Génération des tableaux** : Créez des tableaux de différentes tailles (par exemple, 1,000 ; 10,000 ; 100 000 ; 1000 000 éléments) pour observer la scalabilité.
- **Scénarios de tri** : Générer trois types de tableaux pour chaque taille :
    1. **Aléatoire** : Valeurs sans ordre spécifique.
    2. **Trié dans l'ordre croissant** : Pour observer le comportement des algorithmes quand le tableau est déjà trié.
    3. **Trié dans l'ordre décroissant** : Pour analyser la performance dans le pire cas, notamment pour certains algorithmes comme le tri à bulles.

3. Implémentation des algorithmes de tri

Les étudiants devraient implémenter chaque algorithme avec un **compteur** pour :

- **Comparaisons** : Nombre de comparaisons entre les éléments.
- **Déplacements** : Nombre de fois qu'un élément est déplacé ou échangé.
- **Temps CPU** : Utiliser un chronomètre pour mesurer le temps d'exécution de chaque algorithme.

4. Mesure et Collecte des données

Pour chaque algorithme et chaque type de tableau :

- Exécuter **plus de 30 tests** pour réduire les variations statistiques.
- **Collecter :**
    - Nombre moyen de comparaisons.
    - Nombre moyen de déplacements/échanges.
    - Temps CPU moyen.

Ces données peuvent être organisées dans des **tableaux comparatifs** pour faciliter l'analyse.

5. Analyse des résultats

- **Analyse des comparaisons et déplacements** : Comparer comment les différents types de tableaux (aléatoire, trié, trié inverse) influencent le nombre de comparaisons et de déplacements.
- **Temps CPU** : Comparer les temps d'exécution et identifier les algorithmes les plus performants pour chaque type de tableau.
- **Complexité** : Relier les observations aux complexités théoriques des algorithmes, par exemple $\boldsymbol{O(n^2)}$ pour ces tris simples, et voir si les résultats expérimentaux confirment cette complexité.

6. Conclusion et interprétation

- Comparer les points forts et les faiblesses de chaque algorithme en fonction des scénarios de tri.
- Résumer les conditions dans lesquelles un algorithme serait préférable à un autre.