4️⃣

Tri de données médicales: stabilité et performance de Timsort

Laura Calem - laura.calem@pm.me

13 Février 2024

Le tri est une opération essentielle dans le traitement des données, en particulier dans les bases de données médicales où l'organisation efficace des informations est cruciale pour la recherche et l'analyse. Différents algorithmes de tri offrent des performances variées en fonction du volume des données et des contraintes de mémoire.

Dans ce projet, nous allons comparer trois algorithmes de tri : **Merge Sort, QuickSort et Timsort**, en appliquant ces méthodes sur un dataset médical réel : **Diabetes 130-US Hospitals Dataset**. Ce dataset contient des informations sur plus de **100 000 hospitalisations de patients diabétiques**, avec des attributs comme l'âge, les diagnostics et la durée d'hospitalisation.

L'objectif est d'implémenter **Timsort**, d'étudier ses performances et de le comparer à **Merge Sort** et **QuickSort**, qui vous seront fournis en code de base. Nous allons également explorer l'impact de la stabilité des algorithmes de tri sur les données médicales en réalisant un tri multi-critères.

Marche à suivre

1. **Lecture des données** : écrire une fonction capable de charger le dataset CSV
2. **Implémentation de Timsort** : coder l'algorithme complet en Python
3. **Analyse comparative** : exécuter les trois algorithmes de tri sur le dataset et mesurer :
    a. Le temps d'exécution
    b. Le nombre de comparaisons
    c. L'utilisation mémoire
4. **Optimisation et contraintes mémoire** : tester Timsort avec différentes tailles de "min run" et analyser l'impact de la mémoire sur le tri.

Travail attendu

- Implémentation Python de Timsort avec annotations de type
- Comparaison des performances de Merge Sort, QuickSort et Timsort
- Définition et analyse de la stabilité des algorithmes et son impact sur les données médicales (définir et réaliser un test soit sur les données réelles)
- Présentation finale

Dataset

Option 1: téléchargement direct: https://archive.ics.uci.edu/static/public/296/diabetes+130-us+hospitals+for+years+1999-2008.zip

Option 2: depuis python

1. Installer le package `ucimlrepo`
2. Importer le dataset (sous forme de table pandas) dans le code:

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
diabetes_130_us_hospitals_for_years_1999_2008 = fetch_ucirepo(id=296)

# data (as pandas dataframes)
X = diabetes_130_us_hospitals_for_years_1999_2008.data.features
y = diabetes_130_us_hospitals_for_years_1999_2008.data.targets

# metadata
print(diabetes_130_us_hospitals_for_years_1999_2008.metadata)

# variable information
print(diabetes_130_us_hospitals_for_years_1999_2008.variables)
```

Code de base

Merge sort

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1
```

```
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1
```

Quicksort

```
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quicksort(left) + middle + quicksort(right)
```