Algorithmie

2ème année - 2024/2025

PROJET : ANALYSE DE PERFORMANCE D'ALGORITHMES

Tri et Recherche - Mesure et Comparaison

OBJECTIF

Implémenter FROM SCRATCH et comparer les performances de 7 algorithmes sur des données réelles d'immobilier, puis analyser vos résultats pour comprendre les différences théoriques vs pratiques.

CE QUE VOUS DEVEZ FAIRE :

1. Implémenter 4 algorithmes de tri (sélection, insertion, fusion, rapide)

2. Implémenter 3 algorithmes de recherche (linéaire, binaire, min/max)

3. Tester sur 3 tailles de données (100, 500, 1000 éléments)

4. Mesurer temps et nombre d'opérations pour chaque algorithme

5. Analyser et comparer les résultats dans un rapport

CE QUI VOUS EST FOURNI :

- Fichier CSV uniquement : transactions_immobilieres.csv

- Cet énoncé avec les spécifications techniques

CE QUE VOUS DEVEZ IMPLÉMENTER VOUS-MÊMES :

- TOUT LE CODE -

- Lecture du CSV sans utiliser de bibliothèque

- Les 7 algorithmes avec comptage des opérations

- Les tests de performance et affichage des résultats

ALGORITHMES À IMPLÉMENTER

1. ALGORITHMES DE TRI (4 obligatoires)

A. Tri par Sélection

- Principe : Trouver le minimum, l'échanger avec le premier élément, répéter

- À retourner : (tableau_trié, nb_comparaisons, nb_échanges, temps_execution)

B. Tri par Insertion

- Principe : Insérer chaque élément à sa place dans la partie déjà triée

- À retourner : (tableau_trié, nb_comparaisons, nb_décalages, temps_execution)

C. Tri Fusion

- Principe : Diviser le tableau en deux, trier récursivement, fusionner

- À retourner : (tableau_trié, nb_comparaisons, temps_execution)

D. Tri Rapide

- Principe : Choisir un pivot, partitionner, trier récursivement les sous-tableaux

- À retourner : (tableau_trié, nb_comparaisons, nb_échanges, temps_execution)

2. ALGORITHMES DE RECHERCHE (3 obligatoires)

A. Recherche Linéaire

- Principe : Parcourir le tableau de gauche à droite jusqu'à trouver l'élément

- À retourner : (position_trouvee, nb_comparaisons, temps_execution)

B. Recherche Binaire

- Principe : Dans un tableau trié, diviser par 2 à chaque étape selon la comparaison

- À retourner : (position_trouvee, nb_comparaisons, temps_execution)

C. Recherche Min/Max

- Principe : Trouver la valeur minimale ET maximale en un seul parcours

- À retourner : (valeur_min, valeur_max, nb_comparaisons, temps_execution)

DONNÉES À UTILISER

Source : Fichier transactions_immobilieres.csv (fourni)

- Colonnes disponibles : date_mutation, prix, surface, commune, type_local, nb_pieces, code_postal, prix_m2

- Format : CSV avec en-tête, séparateur virgule

Tailles de test obligatoires :

- 100 éléments : Les 100 premiers biens du fichier

- 500 éléments : Les 500 premiers biens du fichier

- 1000 éléments : Les 1000 premiers biens du fichier

TESTS CONCRETS À EFFECTUER

1. TESTS DES TRIS (sur chaque taille : 100, 500, 1000)

IMPORTANT : Chaque algorithme de tri (Sélection, Insertion, Fusion, Rapide) doit être testé sur TOUS les critères ci-dessous pour permettre la comparaison des performances.

Test A - Tri par Prix :

- Objectif : Trier les biens du moins cher au plus cher

- Données : Colonne prix (2ème colonne)

- Algorithmes à tester : TRI SÉLECTION, TRI INSERTION, TRI FUSION, TRI RAPIDE

- Affichage pour chaque algorithme : "Tri [Nom] par PRIX : [temps]s | [nb_comp] comparaisons | [nb_op] opérations"

Test B - Tri par Surface :

- Objectif : Trier les biens de la plus petite à la plus grande surface

- Données : Colonne surface (3ème colonne)

- Algorithmes à tester : TRI SÉLECTION, TRI INSERTION, TRI FUSION, TRI RAPIDE

- Affichage pour chaque algorithme : "Tri [Nom] par SURFACE : [temps]s | [nb_comp] comparaisons | [nb_op] opérations"

EXEMPLE DE SORTIE ATTENDUE (pour 100 éléments) :

=== TRI PAR PRIX (100 éléments) ===

Tri SÉLECTION par PRIX : 0.001s | 4950 comparaisons | 99 échanges

Tri INSERTION par PRIX : 0.0008s | 2475 comparaisons | 1250 décalages

Tri FUSION par PRIX : 0.0005s | 316 comparaisons

Tri RAPIDE par PRIX : 0.0003s | 350 comparaisons | 45 échanges

=== TRI PAR SURFACE (100 éléments) ===

Tri SÉLECTION par SURFACE : 0.001s | 4950 comparaisons | 99 échanges

Tri INSERTION par SURFACE : 0.0009s | 2680 comparaisons | 1350 décalages

Tri FUSION par SURFACE : 0.0005s | 316 comparaisons

Tri RAPIDE par SURFACE : 0.0004s | 380 comparaisons | 50 échanges

2. TESTS DES RECHERCHES

Test A - Recherche de TOUTES les Maisons à Paris (500 et 1000 éléments) :

- Recherche Linéaire : Comptez combien de maisons sont situées à PARIS dans les données

- Critère : type_local == "Maison" ET commune == "PARIS"

- Important : Parcourir TOUT le tableau pour compter toutes les occurrences

- Affichage : "Recherche linéaire MAISONS PARIS : [temps]s | [nb_comp] comparaisons | Trouvées: [nombre]"

Test B - Recherche Binaire d'un Prix Cible (500 et 1000 éléments) :

- Objectif : Chercher un bien à exactement 350000€ dans le tableau trié par prix

- Données : Tableau trié par prix (colonne prix)

- Affichage : "Recherche binaire PRIX 350000€ : [temps]s | [nb_comp] comparaisons | Position: [pos]"

Test C - Recherche Min/Max des Prix au m² (500 et 1000 éléments) :

- Objectif : Trouver le prix au m² le plus bas ET le plus élevé

- Données : Colonne prix_m2 (8ème colonne)

- Affichage : "Min/Max PRIX_M2 : [temps]s | [nb_comp] comparaisons | Min: [min]€/m² | Max: [max]€/m²"

Test D - Recherche de TOUS les Appartements 3 Pièces (500 et 1000 éléments) :

- Recherche Linéaire : Comptez combien d'appartements ont exactement 3 pièces

- Critère : type_local == "Appartement" ET nb_pieces == 3

- Important : Parcourir TOUT le tableau pour compter toutes les occurrences

- Affichage : "Recherche APPART 3P : [temps]s | [nb_comp] comparaisons | Trouvés: [nombre]"

MESURES OBLIGATOIRES

Pour CHAQUE algorithme, vous devez mesurer :

1. Temps d'exécution (utilisez import time et time.time())

2. Nombre de comparaisons effectuées

3. Nombre d'échanges/déplacements (selon l'algorithme)

Important :

- Comptez MANUELLEMENT chaque comparaison et échange dans vos boucles

- Testez chaque tri sur une copie du tableau original

- Mesurez le temps avant et après l'exécution de l'algorithme

LIVRABLES ATTENDUS

Structure de votre projet :

votre_nom_prenom/

```
├── main.py                  # Programme principal qui lance tous les tests
├── algorithmes_tri.py       # Vos 4 algorithmes de tri
├── algorithmes_recherche.py # Vos 3 algorithmes de recherche
├── utilitaires.py           # Fonctions pour lire CSV, mesurer temps, etc.
├── transactions_immobilieres.csv # Données fournies (ne pas modifier)
├── resultats.txt            # Résultats de vos tests
└── analyse.txt              # Vos réponses aux questions (rédigé par vous)
```

Votre programme doit :

- Lire le CSV et extraire les colonnes demandées sans bibliothèque externe

- Tester TOUS les algorithmes de tri (4) sur TOUS les critères (prix ET surface) pour TOUTES les tailles (100, 500, 1000)

- Tester tous les algorithmes de recherche sur les tailles spécifiées

- Afficher les résultats à l'écran avec le format demandé

- Sauvegarder les résultats dans resultats.txt

CALCUL TOTAL : Vous devrez effectuer 4 algorithmes × 2 critères × 3 tailles = 24 tests de tri + les tests de recherche

QUESTIONS D'ANALYSE

Dans analyse.txt, répondez à ces 10 questions :

Questions sur les TRIS :

1. Y a-t-il une différence de performance entre trier par prix vs trier par surface ? Laquelle et pourquoi ?

2. Quel algorithme de tri est le plus rapide sur 1000 biens immobiliers ? Le classement change-t-il selon le critère (prix/surface) ?

3. Le tri fusion est-il plus stable que le tri rapide pour les données immobilières ? Observez plusieurs exécutions.

4. Pour trier 10 000 annonces immobilières par prix, quel algorithme recommanderiez-vous et pourquoi ?

5. Avez-vous observé des différences significatives entre O(n²) et O(n log n) sur vos données réelles ?

Questions sur les RECHERCHES :

6. Combien de maisons à Paris avez-vous trouvées dans 500 vs 1000 éléments ? Y a-t-il une différence de temps proportionnelle ?

7. La recherche binaire du prix 350000€ est-elle plus rapide que la recherche linéaire ? Par quel facteur ?

8. Quels sont les prix au m² min/max que vous avez trouvés ? Dans quelles villes ?

9. Pour compter tous les appartements 3 pièces, pourquoi la recherche binaire n'est-elle pas adaptée ?

Question de RÉFLEXION :

10. Si vous créiez un site immobilier, quels algorithmes utiliseriez-vous pour :

- Trier les annonces par prix croissant

- Rechercher par ville

- Filtrer par nombre de pièces

- Afficher les biens les plus chers/moins chers

ÉVALUATION (100 points)

Implémentation (60 points)

- Tris (40 pts) : 4 algorithmes corrects qui trient prix ET surface + comptent opérations

- Recherches (20 pts) : 3 algorithmes corrects qui trouvent les biens + comptent opérations

Tests et mesures (25 points)

- Programme principal (15 pts) : Lit le CSV, teste tous les critères, affiche correctement

- Fichier résultats (10 pts) : Format correct, données cohérentes avec critères immobiliers

Analyse (15 points)

- Questions 1-5 (10 pts) : Analyse pertinente des tris sur données immobilières

- Questions 6-10 (5 pts) : Analyse pertinente des recherches et réflexion pratique

---

Contraintes techniques :

- INTERDIT : sorted(), .sort(), csv, pandas, numpy

- AUTORISÉ : time, open(), .split(), int(), len(), fonctions de base Python

Important pour les recherches :

- Maisons à Paris : Vous devez lire ET commune ET type_local

- Prix exact : Assurez-vous que le tableau est bien trié avant la recherche binaire

- Prix au m² : Attention aux valeurs très élevées (Paris) vs très basses (province)