Big Data. TD 2

Natalia Kharchenko, Sergey Kirgizov

ESIREM

Complexité des algorithmes

L'objectif est de déterminer empiriquement les capacités maximales d'un système de calcul.
Pendent ce TD vous découvrez ce que la Big Data signifie dans votre cas.

Veuillez utiliser votre langage préféré.

☞ **EXERCICE 1 :** Réaliser un algorithme de tri de complexité quadratique, par exemple, le tri à bulles. L'algorithme doit prendre en entrée une liste d'entiers et retourner une liste triée.

☞ **EXERCICE 2 :** Tester votre implémentation en utilisant :
- un petit exemple,
- une liste vide,
- une liste aléatoire de 1000 nombres.

Pour vous assurer que votre solution fonctionne bien pour les grandes listes, utilisez une fonction build-in pour vérifier votre résultat (par exemple, sorted() en python).

☞ **EXERCICE 3 :** Réaliser et tester de la même manière une fonction de tri de complexité $O(n \log(n))$, par exemple, tri fusion. Si vous avez besoin de fonctions d'aide pour implémenter votre algorithme, par exemple, si vous avez besoin d'une fonction merge() pour fusionner deux tableaux triés, implémentez une classe. Cette classe doit être initialisée avec une liste de nombres et doit contenir une fonction qui renvoie une liste triée.

☞ **EXERCICE 4 :** Félicitations, vous avez maintenant deux algorithmes de tri ! Généralisez-les pour qu'ils puissent trier des entrées triables arbitraires, et pas seulement des nombres. Pour ce faire, ajoutez un argument key à votre fonction de tri. Cet argument est une fonction de transformation qui peut être appliquée à vos données avant les comparaisons. Par défaut, ce devrait être une fonction identité (lambda x : x en python).

☞ **EXERCICE 5 :** Testez votre solution sur une petite liste de chaînes de caractères en les triant par longueur.

☞ **EXERCICE 6 :** Maintenant, testez vos algorithmes sur les données du TD précédent et obtenez des mesures empiriques de la complexité temporelle. En utilisant les données du TD précédent, préparez plusieurs fichiers de tailles différentes tels que :
- les fichiers contiennent un sous-ensemble aléatoire de lignes du fichier d'origine `https://kirgizov.link/teaching/esirem/bigdata/dataset/wikirank-fr-v2.tsv.zip`
- la colonne de popularité ne contient que des flottants, pas de NaN ou des str
- préparer de 5 à 10 fichiers de tailles différentes pour chaque algorithme
- la taille du plus grand fichier doit être telle que le temps d'exécution estimé de l'algorithme pour ce fichier est d'environ 2 à 5 minutes. Utilisez le nombre d'opérations par seconde de votre langage et la complexité théorique d'algorithme pour obtenir une estimation.

☞ **EXERCICE 7 :** Créez une fonction qui lit un fichier tsv et renvoie une liste de dicts. Chaque dictionnaire représente une ligne du fichier tsv.

☞ **EXERCICE 8 :** Lire vos fichiers dans des listes de dicts et les trier par valeur de popularité en utilisant l'algorithme de tri correspondant. Mesurez et enregistrez le temps d'exécution de chaque fichier.

☞ **EXERCICE 9 :** Visualiser la dépendance entre la taille du fichier et le temps du tri pour les deux algorithmes.

☞ **EXERCICE 10 :** Trouver une approximation de la fonction de cette dépendance pour les deux algorithmes.

☞ **EXERCICE 11 :** Utiliser l'approximation pour prédire le temps nécessaire pour trier un fichier contenant 50M lignes.

☞ **EXERCICE 12 :** Quelle est la taille maximales du fichier que vous pouvez trier en 24 heures sur votre ordinateur ? Tenez compte à la fois du temps et de la mémoire.