Université du Québec en Outaouais **12 novembre 2020**

Département d'informatique et d'ingénierie

INF4063/ Devoir2 de programmation

C'est un travail de groupe de **2 étudiants** (binôme), et il compte **pour 10 points**.

À soumettre en ligne sur Moodle au plus tard le 8 décembre à 23 : 55 sous forme d'un fichier zip. Le nom du fichier zip doit être une concaténation des noms de famille et prénoms des membres du binôme séparés par des « - » et suivi de « -Devoir2 ». Exemple de nom de fichier «CoutuJean-CruiseTom-Devoir2.zip ». Les dossiers extraits du fichier zip auront tous des noms qui se termine par noms et prénoms des étudiants.

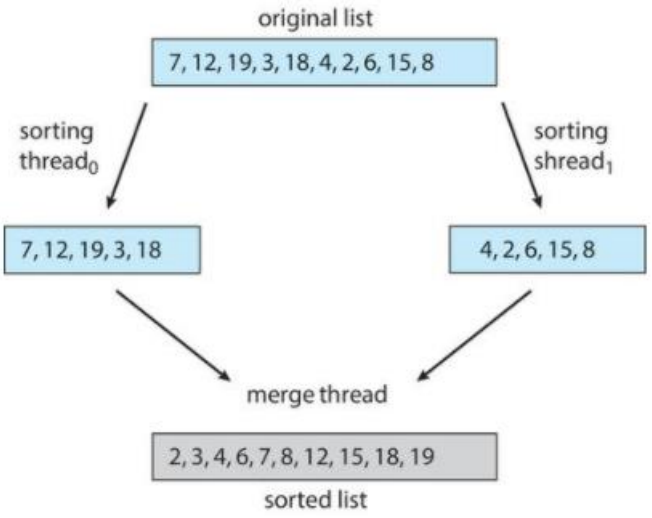
Énoncé

Veuillez lire cet énoncé au complet et la liste des pièces jointes avant de commencer votre travail.

On veut écrire un programme multithread qui trie des listes d'entiers (tableaux). Le programme doit fonctionner comme suit : le programme divise la liste des entiers en des listes de taille égale (dites sous-listes) et les trie séparément pour ensuite les fusionner en une liste complète triée. Les détails de cette procédure sont expliqués dans le paragraphe suivant avec un premier choix de division en deux sous-liste.

Dans le cas de division en deux sous-listes, le programme crée deux threads distincts (que nous appellerons **sorting thread0** et **sorting thread1*)*** qui vont chacun trier une sous-liste à l'aide d'un algorithme de tri par sélection (code java en pièce jointe). Les deux sous-listes sont ensuite fusionnées par un troisième thread, qu'on appelle thread de ***fusion (merge thread),*** qui fusionne les deux sous-listes en une seule liste triée. Un pseudocode de fusion de deux sous-listes est fourni en pièces jointes.

Vu que les données globales sont partagées sur tous les threads, la façon la plus simple de configurer les données est de créer un tableau global. Chaque thread de tri fonctionnera sur la moitié de ce tableau. Un deuxième tableau global de la même taille que le tableau entier non trié sera également établi. Le thread de fusion (merge thread) fusionnera ensuite les deux sous-listes triés dans ce deuxième tableau. Graphiquement, le fonctionnement de programme dans le cas où l'on choisit de diviser seulement en deux sous-listes est illustré à la figure ci-dessous avec un exemple d'un certain tableau (original list) à trier :



Cette division du tableau initial en deux sous-listes représente le mode d'exécution du programme quand on choisit une division en seulement deux sous-liste. Le programme doit permettre de choisir une division de tableau initial en un nombre n de sous-listes qui seront triées en parallèle par n différents sorting threads qu'on dénote par $thread_0$, … , $thread_{n-1}$. Les sous-listes ont même nombre d'éléments à un élément près.

Même si le nombre de sous-liste est supérieure à 2, un seul merge thread est demandé pour faire une seule opération merge des sous-listes.

Détails du programme

Nomenclatures et précisions

C'est un projet java nommé TriThreadsNomsPrénomsÉtdutiants

Le fichier contenant la méthode main doit être nommé TriThreadsMain.java

Les noms des threads de tri doivent être **$thread_0$, … , $thread_{n-1}$**

Le thread de fusion sera nommé : **mergeThread**

Les noms des listes : **listOr** pour liste originale, et list0, list1, .., listk pour les sous-listes.

Le nombre de sous-listes doit être désigné par une variable nommée **NLISTS**

Votre code doit être bien commenté pour le rendre compréhensible par le correcteur. (Commentaires concises et utiles, éviter trop de commentaires qui alourdissent le code plutôt que le rendre plus lisible)

Il est préférable d'adopter un style de code standardisé de votre choix et que vous le nommer dans ce cas.

Utilisez un nombre convenable de fichiers, un minimum nécessaire est recommandé.

Remarque : Le correcteur va utiliser la version jdk-14.0.2 pour la correction.

Entrée/Sortie

Le lancement du programme doit inviter l'utilisateur à :

1. Choisir le mode d'entrée d'une liste à trier de nombres entiers, en choisissant l'une des deux méthodes suivantes :
    a. Mode : entrée directe qui correspond à une saisie d'entiers séparés par des virgules. Ceci invite l'utilisateur à saisir les entiers.
    b. Mode : entrée de la taille d'une liste que le programme crée selon la méthode indiquée dans le fichier java joint à cet énoncé.
2. Passer au point 3, ou continuer pour permettre à l'utilisateur de :
    a. Demander l'affichage des tranches de la liste d'entrée, avec des tranches de maximum 100 éléments affichables à la fois sur l'écran. Avec à tout moment la possibilité de passer de l'affichage d'une tranche vers la suivante, et à tout moment la possibilité de terminer cet affichage pour passer à l'étape suivante qui est le point numéro 3 ci-dessous.
3. Saisir le nombre NLISTS de sous-listes pour la subdivision de la liste originale, ceci détermine en conséquence le nombre de threads que le programme va créer. Ce nombre de NLISTS peut être égale à 1 et ceci implique que la liste originale sera triée sans subdivision et sans fusion.

Il faut que votre programme trie tout tableau d'entier reçu à travers la procédure ci-haut, et son exécution doit **afficher** (avec unité de temps milliseconde) :

1. Le temps du début d'exécution du programme à partir du moment de fin de lecture de l'entrée de la liste originale.
2. La durée totale de l'exécution du tri par le programme en excluant le temps de lecture de la liste originale. Cette durée va être désignée par **« Temps de tri »** dans le reste de ce document.
3. Pour chaque thread, afficher sur une ligne séparée :
    a. Le nom de chaque thread et le moment de lancement de chaque thread

    b. La durée de tri par sélection qu'a pris chaque thread pour trier une sous-liste.
4. Le tableau trié tout entier si la taille du tableau considéré est inférieure à 100, sinon on affiche seulement les premiers 50 éléments et les dernier 50 éléments du tableau trié, et on demande ensuite à l'utilisateur de choisir une tranche d'éléments dont la longueur est inférieure ou égale à 100. La tranche est à spécifier avec deux indices.

La liste originale d'entrée considérée pour toutes les expérimentations et interprétation subséquentes est par défaut une liste identique à celle qu'on peut créer par le code du fichier java joint pour produire une liste de taille 1000.

Remarque : si le temps d'exécution en nanoseconde est inférieur à une milliseconde, vous devez quand même trouver le temps d'exécution en fraction de milliseconde ; il faut alors remédier au faite que une division d'entiers résulte en un entier ignorant la partie fractionnaire.

Livrables et travail demandé :

Code source du projet java nommé TriThreadsNomsPrénomsÉtdutiants :

Un fichier pdf comprenant :

A. Une description sommaire et concise de votre structure de fichiers. Dans cette description, veuillez mettre en surbrillance le nom de votre fichier contenant la méthode main.

B. Une procédure concise et claire de compilation et d'exécution de votre programme.

C. Un tableau qui est remplit à partir du tableau illustré en annexe de cet énoncé et qui fournit les temps d'exécution du programme sur une machine multiprocesseurs selon la liste fournie à travers les pièces jointes de taille 1000 éléments.

D. **Votre interprétation** concise et pas verbeuse de la différence entre les « **Temps de tri »** pour les cas de NLISTS égale à 1, 2, 3 et 4 sous-listes. Cette durée est la durée expliquée au point 2 des affichages exigés dans la section des entrées/sorties ci-haut. Un travail supplémentaire moyennant d'autres listes originales supplémentaires de plus grandes tailles peut être déployé par l'étudiant s'il trouve que ceci aide à mieux interpréter et détecter une différence de temps de tri plus significative. Si l'exécution sur votre machine ne convient pas pour interpréter la comparaison, **vous avez la liberté** d'utiliser des exécutions sur le Cloud avec une machines virtuelle ayant un nombre de processeurs de votre choix. **Ajouter votre interprétation** de la différence entre les temps de tri.

Annexe

Voici la structure du tableau de « temps de tri » en millisecondes demandé comme livrable :

NLISTS	Machine locale Avec ….. processeur(s) ; Système d'exploitation ……	Machine locale Avec ….. processeurs ; Système d'exploitation ……	Machine virtuelle avec …. Processeur(s), Système d'exploitation ……	Machine virtuelle avec …. Processeurs ; Système d'exploitation ……
1				
2				
3				
4				
Interprétation et comparaison **concises**	Voir paragraphe no. ….	Voir paragraphe no. ….	Voir paragraphe no. ….	Voir paragraphe no. ….

Il faut une comparaison pour chaque colonne. On peut produire plusieurs tableaux si on trouve un besoin d'utiliser un tableau supplémentaire pour chaque liste originale supplémentaire que l'étudiant pourrait proposer.

Vous trouvez en pièces jointes:

Un pseudo-code de fusion de deux tableaux triés.
Un fichier java pour tri par sélection : Selection-tri.java
Un exemple de code java ExampleCreateTime.java offrant un exemple de mesure de temps d'exécution de certains lignes de code, en plus de la création d'un tableau de 1000 entiers non triés choisis pour créer la liste originale listOr.

Bon travail