INF4705 – Analyse et conception d’algorithmes

TP1 – Hiver 2018

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Tremblay, David, 1748125  Souli, Taleb, 1770491 |
| **Note finale / 13** | 0 |

# Informations techniques

* Répondez directement dans ce document ODT avec LibreOffice. Veuillez ne pas inclure le texte en italique servant de directive.
* La correction se fait à même le rapport.
* Avant votre troisième séance de laboratoire, vous devez faire une remise électronique en suivant les instructions suivantes:
  + Le dossier remis doit se nommer matricule1\_matricule2\_tp1 et doit être compressé sous format zip.
  + À la racine de ce dernier, on doit retrouver :
    - Ce rapport sous format ODT.
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - L’exemplaire *ex.txt*. Voir énoncé plus bas.
    - Un fichier texte nommé *emails.txt* contenant le courriel de chaque membre de l’équipe
    - Le code source et les exécutables
* Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Le code et les exécutables soumis devront être compatible avec les ordinateurs de la salle L-4714.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.
* Votre chargé de laboratoire vous encourage très fortement à lire le guide intitulé “guide bash” sur moodle pour faire vos graphiques.

# Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l’analyse empirique et hybride des algorithmes. À la section 3.2 des notes de cours, trois approches d’analyse de l’implantation d’un algorithme sont décrites. Vous les mettrez en pratique pour des algorithmes de tri.

Vous implanterez 3 algorithmes de tri:

1. Insertion sort
2. Mergesort
3. Mergesort + seuil de récursivité

De plus, vous devrez comparer ces algorithmes avec *std::sort* de la librairie standard en C++.

Pour l’algorithme 3, vous devez déterminer un seuil de récursivité expérimentalement. Les exemplaires dont la taille est en deça de ce seuil ne sont plus résolus récursivement mais plutôt directement avec le insertion sort.

# Jeu de données

Pour tester les algorithmes, vous devez générer un jeu de données avec 10 exemplaires

pour les tailles: 1000, 5000, 10000, 50000, 100000 et 500000. Vous pouvez utiliser le script suivant:

for n in {"1000","5000","10000","50000","100000","500000"}; do

for i in {1..10}; do

shuf -i 1-$n > testset\_${n}\_${i}.txt

done

done

*Mettez ce script dans un fichier (disons gen.sh), donnez lui les permissions avec chmod +x gen.sh, puis exécuter ce programme dans un nouveau dossier pour générer votre dataset.*

# Présentation des résultats

|  |  |
| --- | --- |
| 0 | / 4 pt |

### Tableau des résulats

*Pour chacun des trois algorithmes, mesurez le temps d’exécution pour chaque exemplaire et rapportez dans un tableau le temps moyen par taille d’exemplaire. Vous vous servirez de ces résultats pour l’analyse qui suit. Lorsque vous calculez les temps d’exécution, vous devez séparer le temps de chargement du jeu de test du temps d’exécution de votre algorithme. Vous devrez donc insérer les sondes temporelles à l’intérieur de votre code.*

|  |  |  |
| --- | --- | --- |
| Algorithme | Temps execution | exemplaire |
|  |  |  |
|  |  |  |
|  |  |  |

### Tests de puissance

*Pour chacun des algorithmes, appliquez le test de puissance et rapportez les graphiques ici.*

### Test du rapport

*Pour chacun des algorithmes, appliquez le test du rapport et rapportez les graphiques ici.*

### Test des constantes

*Pour chacun des algorithmes, appliquez le test des constantes et rapportez les graphiques ici.*

# Analyse et discussion

|  |  |
| --- | --- |
| 0 | / 6 pt |

### Que pouvez-vous déduire du test de puissance?

### Citez la consommation théorique du temps de calcul pour les algorithmes, en notation asymptotique.

*Nul besoin de faire une preuve, on demande seulement de citer.*

### Que pouvez-vous déduire du test du rapport?

### Que pouvez-vous déduire du test des constantes?

### Discutez de l’impact du seuil de récursivité.

### Suite à cette analyse, indiquez sous quelles conditions (taille d’exemplaire ou autre) vous utiliseriez chacun de ces algorithmes. Justifiez.

*Vous pouvez ignorer std::sort, la question concerne les 3 algorithmes que vous avez implémenté.*

### Créer un exemplaire pour lequel un de vos trois algorithmes est plus rapide que *std::sort*. Pourquoi est-il plus rapide?

*Vous devez inclure l’exemplaire à la racine de la remise sous le nom “ex.txt”.*

# Autres critères de correction

### Respect de l’interface tp.sh

|  |  |
| --- | --- |
| 0 | / 1 pt |

Utilisation

tp.sh -a [insertion | merge | mergeSeuil | stdsort] -e [path\_vers\_exemplaire]

Arguments optionnels

-p affiche les nombres triés, sans texte superflu

-t affiche le temps d’exécution en ms, sans unité ni texte superflu

Important: l’option -e doit accepter des fichiers avec des paths absolus.

### Qualité du code

|  |  |
| --- | --- |
| 0 | / 1 pt |

### Présentation générale

|  |  |
| --- | --- |
| 0 | / 1 pt |

* Concision
* Qualité du français

### Pénalité retard

|  |
| --- |
| 0 |

* -1 pt / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.