INF8775 – Analyse et conception d’algorithmes

TP1 – Automne 2022

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Nom, prénom, matricule  Nom, prénom, matricule |
| **Note finale / 30** |  |

# Instructions

* **Attention : Aucune équipe mixte entre les sections de laboratoire.** Si vous voulez changer de groupe, vous devez en faire la demande au registrariat. Si vous avez un conflit d’horaire, veuillez consulter un des chargés de laboratoire.
* Répondez directement dans ce document DOCX ; la correction s’y fait directement.
* Vous devez faire une remise électronique sur Moodle avant le

18 octobre à 23h59 pour le groupe 1 (b1)

25 octobre à 23h59 pour le groupe 2 (b2)

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 DOCX.
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - 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.

# Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l’analyse empirique et hybride des algorithmes. Dans la première capsule vidéo de la section 3, trois approches d’analyse de l’implantation d’un algorithme sont décrites (voir aussi le document *“approches d’analyse empirique”* dans la section 3). Vous les mettrez en pratique pour des algorithmes de recherche des deux points les plus rapprochés dans le plan avec distance euclidienne.

Vous devez analyser trois algorithmes :

1. Algorithme naïf de force brute
2. Algorithme diviser-pour-régner avec un seuil de récursivité élémentaire
3. Algorithme diviser-pour-régner avec un seuil de récursivité déterminé expérimentalement (méthode essai-erreur ; voir première capsule de la section 4)

Les exemplaires dont la taille est en deçà du seuil de récursivité ne sont plus résolus récursivement mais plutôt directement avec l’algorithme de force brute.

# Jeu de données

Nous vous fournissons un générateur d’exemplaires (le fichier gen.py) vous permettant de construire un jeu de données adéquat pour vos besoins. Les instructions d’utilisation de ce générateur se trouvent dans le code source. Les exemplaires générés ont la forme suivante :

*n*

*x1  y1*

*x2  y2*

*...*

*xn  yn*

# Fichiers fournis

Vous n’avez en théorie pas à implanter vous-mêmes les algorithmes pour ce TP. Vous pouvez par exemple ajuster l’implantation suivante en C++ ([lien](https://www.geeksforgeeks.org/closest-pair-of-points-onlogn-implementation/)) pour vos besoins, ou bien cette autre implantation en Python ([lien](https://github.com/AliceB08/closest_points)). Attention, nous ne garantissons pas la validité des implantations trouvées sur internet, même celles dont nous vous avons fourni les liens ci-haut. C’est à vous de vous assurer que les algorithmes sont corrects.

Si, par exemple, vous décidez de prendre l’implantation en C++ dont le lien est donné plus haut, il se peut que vous rencontriez des erreurs de type *segmentation fault* lorsque vous faites des tests sur des exemplaires assez gros (à partir d’un peu moins d’un million de points). Ces erreurs sont dues à la taille de la pile (stack) qui est trop petite. Vous pouvez voir la taille de la pile avec la commande ulimit   
-s (elle sera probablement de 8192). Vous pouvez augmenter sa taille pour éliminer ces erreurs (par exemple ulimit -s 64000). Les modifications apportées par la commande ulimit sont bonnes pour la session seulement. La taille du stack ne peut être augmentée qu’une seule fois par session.

Nous vous fournissons un fichier tp.sh qui servira d’interface avec vos exécutables ou scripts. Référez-vous à la fin de ce document pour le format attendu des arguments.

# Présentation des résultats

|  |  |
| --- | --- |
|  | / 1,5pt |

## Tableau des résultats

*Pour chacun des trois algorithmes, mesurez le temps d’exécution des exemplaires et rapportez dans un tableau le temps moyen pour chaque taille d’exemplaire.*

*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 des sondes temporelles à l’intérieur de votre code.*

*Pensez à indiquer l'unité de temps utilisée.*

# Analyse et discussion

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

|  |  |
| --- | --- |
|  | / 1,5 pt |

## 

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

## Tests de puissance

|  |  |
| --- | --- |
|  | / 5 pts |

## 

*Pour chacun des algorithmes, appliquez le test de puissance et rapportez les graphiques ici. Veuillez afficher vos mesures sous forme de nuage de points. Affichez la courbe de tendance et l'équation de cette courbe.*

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

* 1. *Pensez à formaliser vos conclusions sous forme d'équations de complexité. Justifiez vos conclusions. Pensez à comparer avec la consommation théorique.*

## Test du rapport

|  |  |
| --- | --- |
|  | / 5 pts |

## 

*Pour chacun des algorithmes, appliquez le test du rapport et rapportez les graphiques ici. Pensez à indiquer le rapport effectué pour chaque test.*

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

* 1. *Pensez à formaliser vos conclusions sous forme d'équations de complexité. Justifiez vos conclusions.*

## Test des constantes

|  |  |
| --- | --- |
|  | / 5 pts |

## 

*Pour chacun des algorithmes, appliquez le test des constantes et rapportez les graphiques ici. Veuillez afficher vos mesures sous forme de nuage de points. Affichez la courbe de tendance et l'équation de cette courbe.*

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

* 1. *Pensez à formaliser vos conclusions sous forme d'équations de complexité. Justifiez vos conclusions.*

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

* 1. *Parlez de l’impact théorique et de l’impact expérimental.*

|  |  |
| --- | --- |
|  | / 2 pts |

## 

## Suite à cette analyse, indiquez sous quelles conditions (taille d’exemplaire ou autre) vous utiliseriez chacun de ces algorithmes. Prenez en compte la complexité spatiale et temporelle, le temps de calcul et la difficulté d'implémentation. Justifiez.

|  |  |
| --- | --- |
|  | / 3 pts |

# Autres critères de correction

### Respect de l’interface tp.sh

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

Utilisation

tp.sh -a [brute | recursif | seuil] -e [path\_vers\_exemplaire]

Arguments optionnels

-p affiche la plus petite distance entre deux points, 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

|  |  |
| --- | --- |
|  | / 5 pts |

* + - * 1. Validité des solutions
        2. Qualité de l'implémentation

Présence de commentaires

### Présentation générale

|  |  |
| --- | --- |
|  | / 1 pts |

* 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.