Mini-projet 1 :

Trouver un élément commun entre 2 listes d’entiers

Sommaire

[Introduction 2](#_Toc185190387)

[Algorithme 1 : 2](#_Toc185190388)

[Modification de l’algorithme en utilisant une boucle while 3](#_Toc185190389)

[Performances de l’algorithme 3](#_Toc185190390)

[Nouvelle algorithme avec structure de tas 3](#_Toc185190391)

[Comparaison des performances 3](#_Toc185190392)

## Introduction

L’objectif de ce mini-projet est de créer et implémenter un algorithme permettant de trouver un élément commun entre deux listes d’entier. Nous l’implémenterons d’abord en utilisant une structure de liste classique, et ensuite utilisant une structure de tas, puis nous comparerons les performances des deux structures.

Nous partirons de l’algorithme donné ci-dessous pour le modifier puis l’implémenter.

Algorithme 1 :

**Data :** Deux listes d’entiers *L1*, *L2*

**Résultat :** *x* un élément commun aux deux listes s’il y en a, sinon *NAN*

**Etape 1 :** On détermine *m1* le plus petit élément de *L1* et *m2* le plus petit élément de *L2*;

**Etape 2 :** Si *m1* = *m2* alors on pose *x = m1* et on s’arrête;

**Etape 3 :** Si *m1 < m2* on enlève *m1* de *L1*;

**Etape 4 :** Si *m2 < m1* on enlève *m2* de *L2*;

**Etape 5 :** Si les 2 listes sont non vides on retourne à l’étape 1;

**Etape 6 :** Si (au moins) une liste est vide on s’arrête en renvoyant *NAN*;

## Modification de l’algorithme en utilisant une boucle ‘‘while’’

Voici ci-dessous l’algorithme donné dans l’introduction modifié pour ajouter l’utilisation d’une boucle ‘‘while’’, permettant par la suite une implémentation plus rapide.

### Algorithme 2 : Boucle ‘‘while’’

**Data :** Deux listes d’entiers *L1, L2*

**Résultat :** *x* un élément commun aux deux listes s’il y en a, sinon *NAN*

**Tant que** aucunes des 2 listes n'est vides :

**Etape 1 :** On détermine *m1* le plus petit élément de *L1* et *m2* le plus petit élément de *L2*;

**Etape 2 :** Si *m1 = m2* alors on pose *x = m1* et on s’arrête;

**Etape 3 :** Si *m1 < m2* on enlève *m1* de *L1*;

**Etape 4 :** Si *m2 < m1* on enlève *m2* de *L2*;

**Fin Tant que**

**Etape 5 :** On renvoie *NAN*;

## Performances de l’algorithme

## Nouvelle algorithme avec structure de tas

## Comparaison des performances