Atelier d'informatique

Épisode IV : Algorithmes de tri

1er février 2017

1 Introduction

2 Tri par sélection

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).
Les algorithmes de tri, suites d'instructions permettant de résoudre ce problème, sont multiples et constituent des sujets d'étude classique pour l'informaticien en herbe :

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).
Les algorithmes de tri, suites d'instructions permettant de résoudre ce problème, sont multiples et constituent des sujets d'étude classique pour l'informaticien en herbe : efficacité,

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).
Les algorithmes de tri, suites d'instructions permettant de résoudre ce problème, sont multiples et constituent des sujets d'étude classique pour l'informaticien en herbe : efficacité, complexité temporelle (temps d'exécution),

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).
Les algorithmes de tri, suites d'instructions permettant de résoudre ce problème, sont multiples et constituent des sujets d'étude classique pour l'informaticien en herbe : efficacité, complexité temporelle (temps d'exécution), stabilité (est-ce que le temps d'exécution augmente beaucoup si je modifie un petit peu ma liste ?).

Introduction

Dans ce chapitre, on s'intéresse au problème ultra-classique du tri des éléments d'une liste, quand ils sont comparables entre eux : des nombres pour l'ordre usuel sur $\mathbb{R}$, ou des chaînes de caractères pour l'ordre lexicographique (l'ordre des mots dans un dictionnaire).
Les algorithmes de tri, suites d'instructions permettant de résoudre ce problème, sont multiples et constituent des sujets d'étude classique pour l'informaticien en herbe : efficacité, complexité temporelle (temps d'exécution), stabilité (est-ce que le temps d'exécution augmente beaucoup si je modifie un petit peu ma liste ?).
On verra dans ce chapitre quelques algorithmes simples, que l'on essaiera de programmer en langage Python.

Tri par sélection

On va commencer par l'algorithme de tri le plus évident de tous. Étant donnée une liste `l`, on sélectionne son plus grand élément, que l'on place à la fin. Maintenant, on recommence sur ce qui reste de la liste. Et ainsi de suite.

Tri par sélection

On va commencer par l'algorithme de tri le plus évident de tous. Étant donnée une liste `l`, on sélectionne son plus grand élément, que l'on place à la fin. Maintenant, on recommence sur ce qui reste de la liste. Et ainsi de suite.
Au niveau du nombre de calculs, si n est la longueur de la liste, on effectue au plus n comparaisons pour trouver son plus grand élément. Pour le plus grand élément de ce qui reste, on fait au plus $n-1$ comparaisons. Puis $n-2$... ainsi de suite. Finalement, on fait au pire

$$n + n - 1 + n - 2 + \cdots + 1 = 1 + 2 + \cdots + n = \frac{n(n-1)}{2}$$

comparaisons (formule démontrée en classe de Première S).