

# Table des matières

<b>1</b>	<b>Présentation du projet</b>	<b>1</b>
1.1	Sujet . . . . .	1
1.2	Notations utilisées . . . . .	1
1.3	Exemple . . . . .	1
<b>2</b>	<b>Machine learning</b>	<b>5</b>
2.1	Etude du sujet . . . . .	5
2.2	Problème posé . . . . .	5
2.3	Solution proposée . . . . .	5
2.4	Réalisation . . . . .	5

# 1 Présentation du projet

## Intro<sup>1</sup>

L'étude de la complexité algorithmique tient une place prépondérante lorsqu'on s'intéresse à des programmes devant traiter un grand nombre de données. En effet, entre deux programmes ayant la même finalité il se peut que la différence de temps d'exécution soit comptée en jours. C'est pourquoi, il est important de choisir un algorithme optimal pour chaque programme. Mais, bien que la complexité de certains programmes soit bien connue (telle que les tris de tableau), certaines complexités sont plus compliquées à prédire. Le but de ce stage fut de trouver une façon de connaître la complexité de tout algorithme en se basant seulement sur leur temps d'exécution.

## 1.1 Sujet

Comme dit dans la courte introduction, le but de ce stage fut de trouver la complexité des algorithmes juste en connaissant leur temps d'exécution. Le projet c'est donc décomposé en deux phases. Durant la première, le but fut de retrouver la complexité d'algorithmes bien connus (Tri à bulles par exemple) afin de vérifier que la méthode utilisée donnait des résultats corrects. Ensuite, la seconde partie

## 1.2 Notations utilisées

**n** : taille du tableau

**m** : nombre de tailles différentes

**k** : taille acutél

**q** : nombre de features

**Régression linéaire** :  $H_\theta(n) = \theta_0 + \theta_1 \log(n) + \theta_2 n + \theta_3 n \log(n) + \theta_4 n^2 + \theta_5 n^2 \log(n) + \theta_6 n^3$

## 1.3 Exemple

Afin d'expliciter mes propos, je vais donner un exemple que je vais utiliser tout le long du rapport à chaque fois que j'aurais besoin d'illustrer mes propos.

**Algorithme** : Tri à bulles

**Tailles de tableau utilisées** : 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000

**Complexité théorique** :  $O(n^2)$

$\log(n)$	$n$	$n \log(n)$	$n^2$	$n^2 \log(n)$	$n^3$	Temps CPU
9.21	10000	92103.40	$1.0 * 10^8$	$9.0 * 10^8$	$1.0 * 10^{12}$	56ms
9.90	20000	198068.75	$4.0 * 10^8$	$4.0 * 10^9$	$8.0 * 10^{12}$	187ms
10.31	30000	309268.58	$9.0 * 10^8$	$9.0 * 10^9$	$2.7 * 10^{13}$	491ms
10.60	40000	423865.39	$1.6 * 10^8$	$2.0 * 10^{10}$	$6.4 * 10^{13}$	884ms
10.82	50000	540988.91	$2.5 * 10^9$	$3.0 * 10^{10}$	$1.3 * 10^{14}$	1482ms
11.00	60000	660126.00	$3.6 * 10^9$	$4.0 * 10^{10}$	$2.2 * 10^{14}$	1612ms
11.16	70000	780937.54	$4.9 * 10^9$	$5.0 * 10^{10}$	$3.4 * 10^{14}$	2139ms
11.29	80000	903182.55	$6.4 * 10^9$	$7 * 10^{10}$	$5.1 * 10^{14}$	2962ms
11.41	90000	1026680.85	$8.1 * 10^9$	$9.0 * 10^{10}$	$7.3 * 10^{14}$	3708ms
11.51	100000	1151292.55	$1.0 * 10^{10}$	$1.0 * 10^{11}$	$1.0 * 10^{15}$	4608ms

FIGURE 1 – Tableau récapitulatif des résultats obtenus pour le tri à bulles

Ce tableau est obtenu par l'exécution successive de l'algorithme pour les différentes tailles en mesurant le temps d'exécution à chaque fois.

Une fois ce tableau de résultat obtenu, le but est de procéder à une régression linéaire sur celui-ci avec les 6 premières colonnes comme variables et la dernière comme variable cible.

La régression linéaire est obtenue par la méthode des moindres carrés et aura la forme suivante :

$$\mathbf{H}_{\theta}(\mathbf{n}) = \theta_0 + \theta_1 \log(\mathbf{n}) + \theta_2 \mathbf{n} + \theta_3 \mathbf{n} \log(\mathbf{n}) + \theta_4 \mathbf{n}^2 + \theta_5 \mathbf{n}^2 \log(\mathbf{n}) + \theta_6 \mathbf{n}^3$$

Où chaque  $\theta_i$  permettra de déterminer la complexité de l'algorithme testé.

Afin de vérifier le résultat obtenu, on trace sur le même graphique la courbe du temps CPU par rapport à la taille du tableau ( $\mathbf{n}$ ).

Le graphique obtenu ressemble donc à ça :

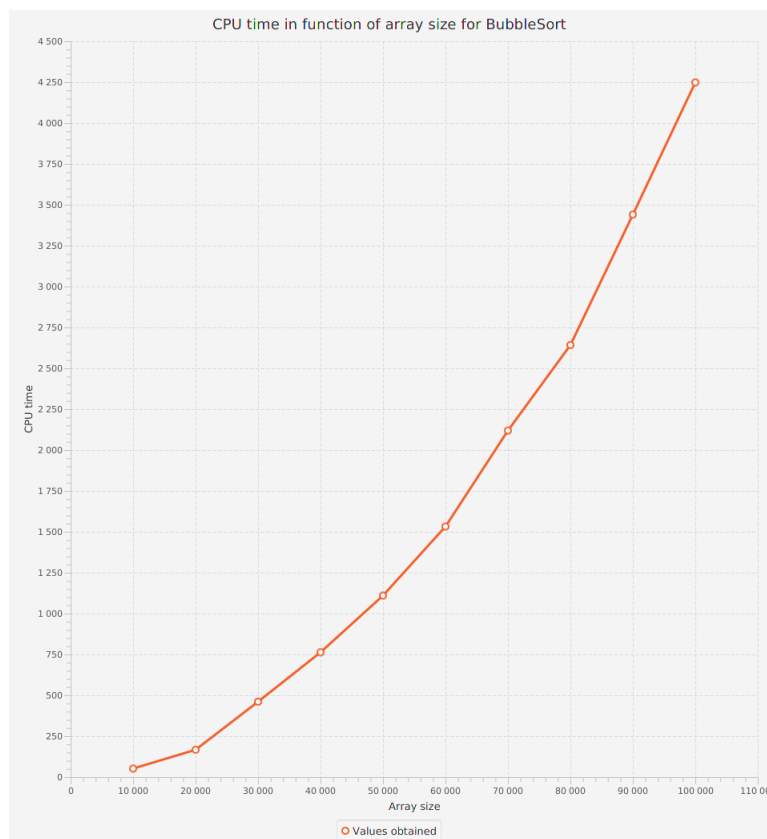


FIGURE 2 – Graphique regroupant le temps CPU en fonction de la taille du tableau

Ensuite, on calcul l'équation de la régression linéaire obtenu.

On obtient le resultat suivant :

$$\mathbf{H}(\mathbf{n}) = 257498.66 - 35867.89\log(\mathbf{n}) + 54.14\mathbf{n} - 5.30\mathbf{n}\log(\mathbf{n}) + (7.10 * 10^{-4})\mathbf{n}^2 \\ - (5.52 * 10^{-5})\mathbf{n}^2\log(\mathbf{n}) + (1.13 * 10^{-10})\mathbf{n}^3$$

Or, si on choisit de prendre le plus grand  $\theta$  afin de déterminer la complexité, ici on obtient une complexité en  $O(n)$  ce qui est incorrecte, on doit donc trouver une autre méthode afin de déterminer la complexité à partir de la régression linéaire.

Afin de déterminer la complexité plus précisément, on calcul la regression linéaire pour plusieurs degrés de complexité à partir du tableau de résultat :

$$\mathbf{H}_{\theta}^1(\mathbf{n}) = \theta_0 + \theta_1\log(\mathbf{n})$$

$$\mathbf{H}_{\theta}^2(\mathbf{n}) = \theta_0 + \theta_2\mathbf{n}$$

$$\mathbf{H}_{\theta}^4(\mathbf{n}) = \theta_0 + \theta_4\mathbf{n}^2$$

$$\mathbf{H}_{\theta}^6(\mathbf{n}) = \theta_0 + \theta_6\mathbf{n}^3$$

Et pour chaque'une de ces regressions linéaires, on calcul la somme des carrés de l'erreur résiduelle.

Logiquement, l'erreur résiduelle obtenu la plus faible correspond à la régression de la complexité de l'algorithme.

On obtient alors les résultats suivants :

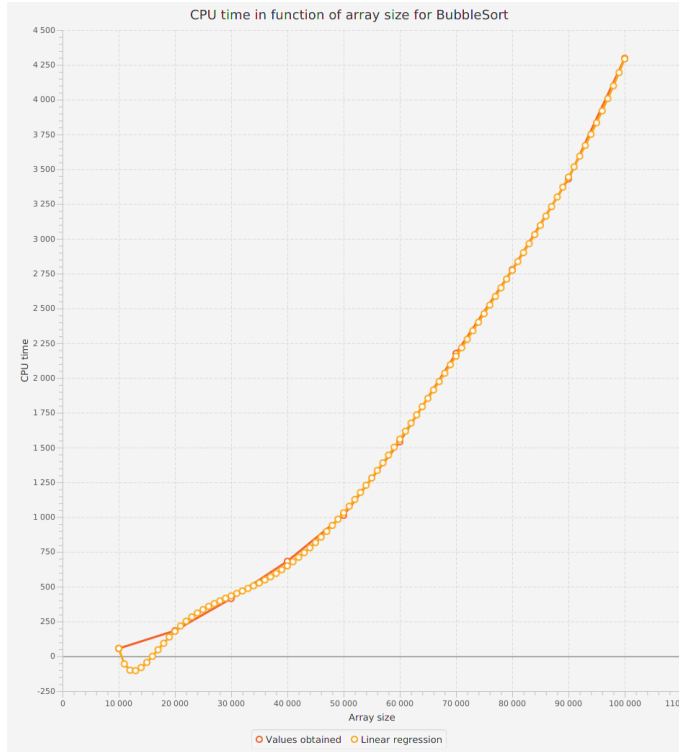


FIGURE 3 – Régression linéaire

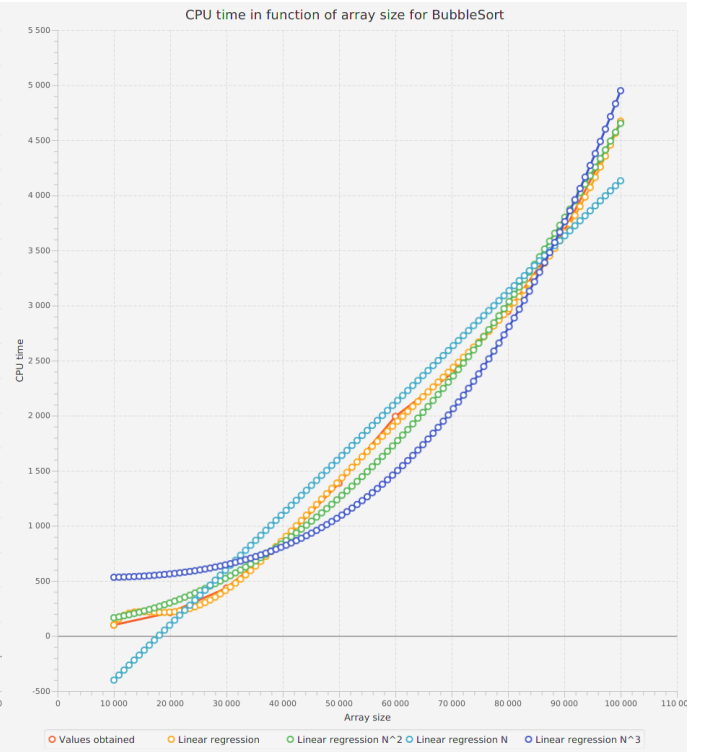


FIGURE 4 – Toutes les régressions linéaire

$H_{\theta}^i(n)$	Erreur résiduelle
Régression linéaire	79.07
$H_{\theta}^1(n)$	6129745.36
$H_{\theta}^2(n)$	1202786.76
$\theta_0 + \theta_3 n \log(n)$	1001921.46
$H_{\theta}^4(n)$	374170.4
$H_{\theta}^6(n)$	1347475.21

FIGURE 5 – Tableau récapitulatif des erreurs résiduelles

L'erreur résiduelle la plus faible étant pour la régression de la forme  $\mathbf{H}_{\theta}^4(\mathbf{n}) = \theta_0 + \theta_4 \mathbf{n}^2$ , on obtient bien une complexité en  $O(n^2)$  comme voulu.

## 2 Machine learning

Suite au calcul de la somme des carrés de l'erreur résiduelle, on obtient le bon résultat avec une complexité en  $O(n^2)$  ici pour le tri à bulles. Cependant, il reste un petit problème à résoudre, comment distinguer une complexité en  $O(n)$  et une en  $O(n \log(n))$  ainsi que une complexité en  $O(n^2)$  et une en  $O(n^2 \log(n))$  car celles-ci sont très proche l'une de l'autre ce qui complique le calcul de l'erreur résiduelle.

### 2.1 Etude du sujet

Intro

### 2.2 Problème posé

Bla

### 2.3 Solution proposée

Bla

Transition

### 2.4 Réalisation