Complexité algorithmique

Notion de problème et de classe de problèmes
Complexité algorithmique
Fonction temporelle de complexité
Notation asymptotique
Classe de complexité

Version de travail : rentrée 2021

Olivier Raynaud

Sensibilité :

https://www.pourlascience.fr/tags/complexite-algorithmique

1

Algorithme

❑ **Rappel historique**

Le mot *algorithme* vient du nom d'un mathématicien perse du IX^e siècle, Al-Khwârizmî

Définition Algorithme : Un *algorithme* est une procédure finie de calcul, composée d'opérations et d'instructions, non ambigues, pas à pas, permettant de résoudre une classe de problèmes.

Wikipedia

Un algorithme est une méthode générale pour résoudre un type de problèmes. Il est dit *correct* lorsque, pour chaque instance du problème, il se termine en produisant la bonne sortie, c'est-à-dire qu'il résout le problème posé

2

Problème

❑ **Problème algorithmique**

Les problèmes algorithmiques jouent un rôle central et forment un domaine à part entière, à côté de celui des algorithmes

Définition Problème : Un **problème** est un objet mathématique qui représente une question ou un ensemble de questions auxquelles une machine devrait répondre. Un problème est de la forme : étant donné un objet (l'instance), effectuer une certaine action ou répondre à telle question.

Wikipedia

On distingue en particulier deux types de problèmes :
- **Les problèmes de décision**, qui consistent à répondre oui ou non à une question
- **Les problèmes d'évaluation** ou de construction, qui consistent à produire un objet spécifié par l'énoncé du problème.

3

Complexité

❑ **Complexité algorithmique**

La complexité algorithmique permet de prédire l'évolution en temps calcul nécessaire pour amener un algorithme à son terme, en fonction de la quantité de données à traiter.

La **théorie de la complexité** est le domaine de l'informatique théorique qui étudie formellement la *quantité de ressources* (temps, espace mémoire, etc.) dont a besoin un algorithme pour résoudre un problème algorithmique.

Wikipedia

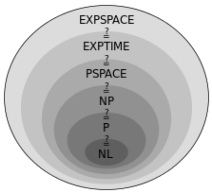
➢ ***Question :*** étant donnés deux algorithmes qui calculent les solutions à un même problème. Comment comparer ces algorithmes? Autrement dit, quel est le meilleur?

4

Classe de complexité de problème

Définition Classe de complexité : Une **classe de complexité** est un ensemble de problèmes algorithmiques dont la résolution nécessite la même quantité d'une certaine ressource.

Wikipedia



5

Temps d'exécution

❑ **Comment évaluer le temps d'exécution d'un algorithme?**

Idée : compter le nombre d'opérations élémentaires effectuées lors de l'exécution.

Définition Copération élémentaire : Une *opération élémentaire* est une opération qui s'effectue en temps constant sur tous les calculateurs usuels.

❑ Sur des types simples
- **Comparaison,**
- **Opération arithmétique**
- **Entrée-sorties**

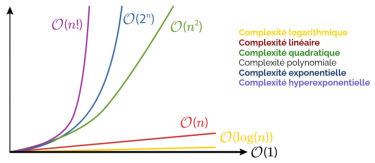
Remarque : une opération qui se compose de plusieurs opérations élémentaires peut être considérée comme une OE si le nombre d'OE qui la compose est constant

6

Fonction de complexité

Définition : La **fonction temporelle de complexité** d'un algorithme exprime le temps requis par l'algorithme pour calculer la solution correspondant à une instance, en fonction de la taille de celle-ci.

❑ **Cette fonction de complexité dépend donc**
- du codage retenu pour évaluer la taille de l'instance
- du modèle de machine utilisé pour l'évaluation du temps d'exécution d'une opération élémentaire.



https://www.mathweb.fr/euclide/complexite-algorithmique

7

Hypothèse de simplification

❑ Pour évaluer le nombre d'opérations élémentaires on s'appuie sur les paramètres de description de la donnée:

- nombre d'éléments d'un tableau,
- nombre de caractères d'une chaîne;
- nombre d'éléments d'un ensemble;
- profondeur et largeur d'un arbre;
- dimension d'une relation d'ordre;
- nombre de sommets et d'arêtes d'un graphe;
- taille ou valeur des nombres caractéristiques du problème;
- modèle de machine utilisé pour l'évaluation du temps d'exécution d'une EO.

8

Critères d'évaluation

❑ **Difficulté:**

Pour une même taille de données, le nombre d'opérations élémentaires exécutées reste variable. On propose plusieurs critères dévaluation :

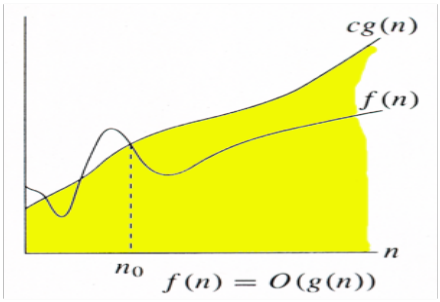
✓ L'analyse dans le pire des cas : **t(n) = maximum des temps d'exécution** de l'algorithme pour toutes les instances de taille n.

✓ L'analyse *moyenne* : $\mathbf{t_{moy}(n)}$ **= moyenne des temps d'exécution** de l'algorithme pour toutes les instances avec une répartition probabiliste des tailles de données.

✓ L'analyse lisse d'algorithme, plus récente, se veut plus proche des situations réelles en calculant une complexité des le pire des cas sur des instances légerement bruitées.

9

Ordre de grandeur – notation $\mathcal{O}$



10

Notation asymptotique

❑ **Borne supérieure asymptotique**

✓ La notation $\mathcal{O}$ de Landau dénote le caractère **dominé** d'une fonction par rapport à une autre.

Définition Soient *f* et *g* deux fonctions de la variable réelle *x*. On dit que ***f* est dominée par *g*** en + ∞, ou que *g* domine *f* en +∞, si il existe une constante c positive et non nulle telle que **f(n) <= c.g(n)** pour n assez grand. Cette relation de domination est notée :

$$f(x) = O(g(x)) \ (x \to \infty)$$

- On étudie donc le comportement de la fonction de complexité pour les grandes valeurs de n.
- On parle alors de comportement à la limite ou d'ordre de grandeur de la fonction f.

11

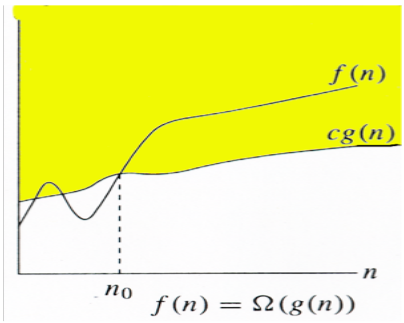
Tâche de référence

Complexité	Nature de la tâche
O(1)	Accès direct à un élément de E
O(log n)	Divisions successives par deux d'un ensemble E
O(n)	Parcours d'un ensemble E
O(n log n)	Divisions successives par deux et parcours de toutes les parties de E
$O(n^2)$	Parcours d'une matrice carrée de taille n
$O(2^n)$	Génération des parties d'un ensemble E
O(n!)	Génération des permutations d'un ensemble E

Note: E un ensemble de tallaie n (|E|=n)

12

Ordre de grandeur – notation Ω



13

Notation asymptotique

❑ Borne asymptotique inférieure

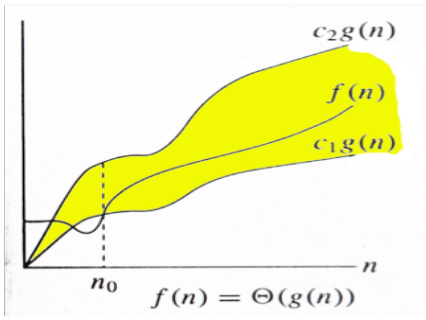
✓ La notation Ω introduite pat Knuth en 1976.

Définition : Soient *f* et *g* deux fonctions de la variable réelle *x*. On dit que ***f* est minorée par *g*** en + ∞, ou que *g* minore *f* en +∞, si il existe une variable c non nulle telle que 0 < c.g(n) <= f(n) pour n assez grand. Cette relation de minoration est notée :

$$f(x) = \Omega(g(x)) \ (x \to \infty)$$

14

Ordre de grandeur – notation θ



15

Notation asymptotique

❑Borne asymptotique inférieure

✓ La notation θ permet de définir un encadrement.

Définition : Soient *f* et *g* deux fonctions de la variable réelle *x*. On dit que ***f* dominée et soumise à *g*** en + ∞ si il existe deux variables k1 et k2 telles que k1.g(n) < f(n) < k2.g(n) pour n assez grand. Cette relation est notée :

$$f(n) \in \Theta(g(n))$$

Théorème Pour deux fonctions quelconques *f(n) et g(n),*

f(n) = θ(g(n)) ***si et seulement si*** f(n) = $\mathcal{O}$(g(n)) et f(n) = Ω(g(n))

16

Codage des données

❑Question

✓ Quelle peut-être l'influence du codage pour l'évaluation de la complexité algorithmique?

Restriction sur la taille du codage : La taille du codage choisi diffère au plus polynomialement si l'on respecte les deux conditions suivantes:

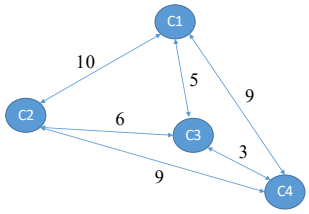
- Le codage d'une instance doit être concis et ne pas comporter d'information non nécessaire.
- Les nombres apparaissant dans la description de I ne doivent pas être représentés en base unaire.

17

Exemple de codage

❑Question

✓ Quelle peut-être l'influence du codage pour l'évaluation de la complexité algorithmique?



L'alphabet peut être : {c,[,],/,0,1,2,3,4,5,6,7,8,9}

Le codage obtenu : « c[1]c[2]c[3]c[4]//10/5/9//6/9//3 »

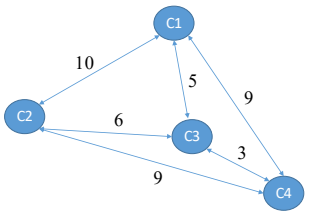
Extrait de [GJ00]

18

Exemple de codage

❑ Généralisation

✓ Soit G=(S,A) avec |S|=s et |A|=a.
Comparer la taille des codages de G en fonction de sa représentation.



Extrait de [GJ00]

19

Modèle de machine

Machine simulant

Machine simulée

	1TM	kTM	RAM
1TM		O(T(n))	O(T(n)logT(n))
kTM	O(T²(n))		O(T(n)logT(n))
RAM	O(T³(n))	O(T²(n))	

Extrait de [GJ00]

20

Classe de complexité algorithmique

Définition : Un **algorithme en temps polynomial** est un algorithme dont la fonction temporelle de complexité est en O(p(n)) avec p une fonction polynomiale quelconque.

Définition : Un algorithme est dit **en temps exponentiel** si et seulement si il n'est pas en temps polynomial.

Cette définition par exclusion à l'inconvénient de classer exponentielle une fonction telle que $n^{\log(n)}$ qui ne l'est pas pour certains.

21

Temps d'exécution

Taille des instances

Fonction temporelle de complexité	10	20	30	40	50	60
n	.00001s	.00002s	.00003s	.00004s	.00005s	.00006s
n^2	.0001s	.0004s	.0009s	.0016s	.0025	.0036s
n^3	.001s	.008s	.027s	.064s	.125s	.216s
n^5	.1s	3.2s	24.3s	1.7mn	5.2mn	13.0mn
2^n	.001s	1.0s	17.9mn	12.7j	35.7an	366 siècles
3^n	.059s	58 mn	6.5 an	3855siècles	$2\ 10^8$siècles	1.310^{13} siècles

Extrait de [GJ00]

22

Projection

Taille de l'instance	1	100	1000
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.63N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5+6.64$	$N_5+9.97$
3^n	N_6	$N_6+4.19$	$N_6+6.29$

Extrait de [GJ00]

23

Classe de problèmes

❑ **Classification**

✓ Une des problématiques majeures de l'informatique théorique concerne la classification des problèmes.

Définition : Un problème est dit **intraitable** s'il est si difficile qu' aucun algorithme polynomial ne puisse le résoudre.

A quelle classe de complexité appartient un problème donné?

- ✓ Un problème donné est-il traitable ou intraitable?
- ✓ S'il est intraitable existe-t-il de bons algorithmes d'approximation?

24

Pour résumer

- *Quelques points essentiels de l'exposé*
  - ✓ Nous avons défini les concepts de **problème** et de **classe de complexité** associée;
  - ✓ Nous avons défini les concepts de **complexité algorithmique** et **de fonction temporelle de complexité;**
  - ✓ Nous avons introduit les notations asymptotiques $\mathcal{O}$, Ω et θ.
  - ✓ Nous avons montré **l'influence du codage et du modèle de calcul** sur l'évaluation de fonctions temporelles de complexité;
  - ✓ Nous avons défini les classes d'algorithmes en temps polynomial, en temps exponentiel et la classe des problèmes dits intraitables.

25