La Complexité Algorithmique

-Efficacité des Algorithmes-

ENCADRE PAR:
Pr. El MEROUANI Mohamed

PRESENTE PAR:
MELLAGUI Mohammed

Année Universitaire 2014-2015

Introduction

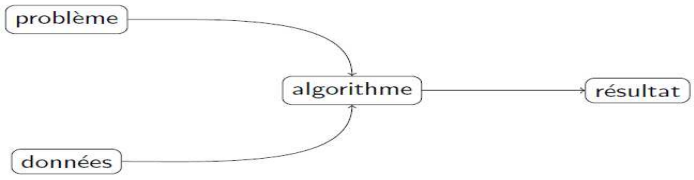
De nombreuses personnes se sont penchées sur des problèmes Algorithmiques et ont développé une théorie dite de la complexité. Nous ne verrons pas ici en détail les fondements de cette théorie mais tenterons plutôt d'acquérir une vision globale de cette problématique.

Plan

- *Définitions : Algorithme / Machine de Turing*
- *La Complexité d'un algorithme*
- *L'efficacité d'un algorithme*
- *Problème d'optimisation Combinatoire, de reconnaissance*
- *Problème facile/difficile: Décidable/Indécidable*
- *Les Catégories de problème [Les classes]*
- *Conclusion*

Qu'est-ce qu'un algorithme?

- Un algorithme est une méthode permettant de résoudre un problème donné en un temps fini ;

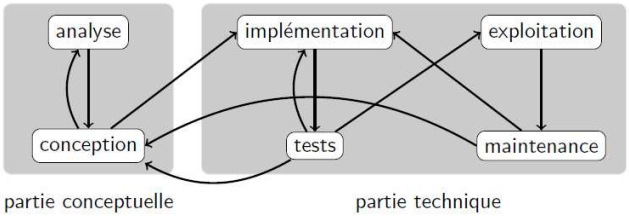


- Un algorithme n'est pas un programme !
- L'algorithme décrit une méthode qui sera ensuite implémentée dans un langage de programmation ;

Qu'est-ce qu'un algorithme?

(suite)

- Le développement d'un programme se divise en plusieurs phases :

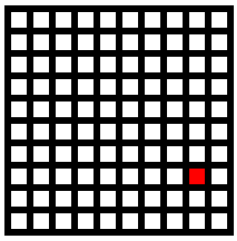


- En pratique, les choses sont souvent plus compliquées ...

Qu'est-ce qu'un algorithme?

(suite)

- En algorithmique, on veut expliquer (a un ordinateur) comment résoudre efficacement un problème ;
- Remarquons que certains problèmes sont évidents pour un humain, moins pour un ordinateur ;
- Exemple simple : ou est la case rouge ?



Qu'est-ce qu'un algorithme?

(suite)

- Les taches a accomplir étant parfois complexes, donc pour résoudre un problème il est important de les décomposer et d'établir un plan a suivre ;
- Principe : on découpé une tâche complexe en tâches plus simples ;
- Exemple : créer un réseau social (Facebook, etc.) :
  - stocker et sauvegarder les données ;
  - représenter les utilisateurs et leurs attributs ;
  - interface ;
  - gestion de la vie privée ;

  …

Qu'est-ce qu'un algorithme?

(suite)

Trois questions Essentielles

- Un algorithme a pour objectif la résolution d'un problème.

  Est-ce que l'algorithme donne...

1. une réponse ? ➔ Terminaison
2. la bonne réponse ? ➔ Correction
3. la bonne réponse en un temps acceptable ? ➔ Complexité

Qu'est-ce qu'une Machine de Turing?

Est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur et sa mémoire. Elle sert d'« étalon » à la mesure des complexités en temps et en espace des fonctions dites calculables.

Ce modèle a été imaginé par Alan Turing en 1936, en vue de donner une définition précise au concept d'algorithme ou de « procédure mécanique ».

Il est toujours largement utilisé en informatique théorique (l'étude des fondements logiques et mathématiques de l'informatique), en particulier dans les domaines de la calculabilité et de la complexité algorithmique, qui étudie formellement la quantité de ressources (en temps et en espace) nécessaire pour la résolution de problèmes au moyen de l'exécution d'un algorithme

La Complexité algorithmique

On désigne par complexité d'un algorithme le nombre d'opérations nécessaires à celui-ci pour s'exécuter. Bien évidemment, ce nombre peut varier en fonction de ce que l'on appelle les données d'entrées, c'est-à-dire les paramètres que l'on donne à l'algorithme.

En résumé, on mesure l'efficacité d'un algorithme par une expression mathématique qui indique le nombre d'opérations indispensables à l'exécution de l'algorithme en fonction de la taille des données en entrées tout en supposant le pire des cas.

Le critère de rapidité n'est pas toujours celui qui nous intéresse. Dans ce cas, on parle de complexité spatiale alors que jusqu'à présent on considérait la complexité temporelle.

La Complexité algorithmique

Définitions (complexités temporelle et spatiale)

- ***complexité temporelle*** *: (ou* ***en temps****) : temps de calcul ;*
- ***complexité spatiale*** *: (ou* ***en espace****) : l'espace mémoire requis par le calcul.*

Définitions (complexités pratique et théorique)

- *La* ***complexité pratique*** *est une mesure précise des complexités temporelles et spatiales pour un* ***modèle de machine donné****.*
- *La* ***complexité (théorique)*** *est un* ***ordre de grandeur*** *de ces couts, exprimé de manière la plus* ***indépendante*** *possible des conditions pratiques d'exécution.*

La Complexité algorithmique

Objectifs des calculs de complexité :

- pouvoir prévoir le temps d'exécution d'un algorithme
- pouvoir comparer deux algorithmes réalisant le même traitement

Exemples :

- si on lance le calcul de la factorielle de 100, combien de temps faudra t-il attendre le résultat?

- quel algorithme de tri vaut-il mieux utiliser pour retrier un tableau où on vient de changer un élément?

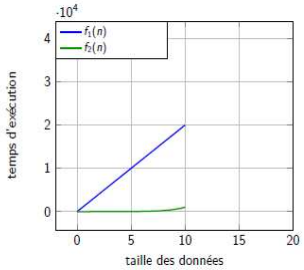
La Complexité algorithmique

- La complexité d'un algorithme est une mesure de sa performance asymptotique dans le pire cas
- Que signifie « asymptotique » ?
  - ➔ on s'intéresse a des données très grandes ;
    - pourquoi ?
      - ➔ les petites valeurs ne sont pas assez informatives ;
- Que signifie « dans le pire cas » ?
  - ➔ on s'intéresse a la performance de l'algorithme dans les situations ou le problème prend le plus de temps a résoudre ;
    - pourquoi ?
      - ➔ on veut être sûr que l'algorithme ne prendra jamais plus de temps que ce qu'on a estime ;

La Complexité algorithmique

Intuition : Le Comportement Asymptotique

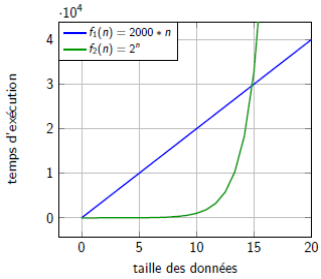
- Soit
  1. un problème a résoudre sur des données de taille n, et
  2. deux algorithmes résolvant ce problème en un temps f1(n) et f2(n), respectivement ;
- Quel algorithme préférez-vous ?
- La courbe verte semble correspondre à un algorithme beaucoup efficace ...



La Complexité algorithmique

Intuition : Le Comportement Asymptotique

- Soit
  1. un problème a résoudre sur des données de taille n, et
  2. deux algorithmes résolvant ce problème en un temps f1(n) et f2(n), respectivement ;
- Quel algorithme préférez-vous ?



- La courbe verte semble correspondre à un algorithme beaucoup efficace …
- ... mais seulement pour de très petites valeurs !

Algorithme Efficace

Une fois la complexité définie de manière succincte, on peut tenter de définir ce qu'est un algorithme efficace. **Un algorithme sera dit efficace si sa complexité est bornée par un polynôme ayant la taille des données comme variable.** Par exemple, un algorithme qui a en entrée un tableau de n éléments et qui a une complexité de n^2 est un algorithme efficace.

On dit aussi que l'algorithme est polynomial. Cette définition est justifiée par le fait qu'on s'intéresse aux performances des algorithmes quand la taille des données en entrée devient très importante.

Algorithme Efficace

Par exemple, considérons les algorithmes A, B et C. Leur complexité sont les suivantes :

- $C_a = 80n$,
- $C_b = 10n^2$,
- $C_c = n!$.

➔ Avec 4 éléments, il faut respectivement 320, 160 et 24 opérations aux algorithmes A, B et C pour s'exécuter. Le plus efficace pour 4 éléments est donc C.

Considérons maintenant 20 éléments, il faut respectivement 1600, 4000 et 2.4 1018 opérations pour réaliser l'exécution. On s'aperçoit tout de suite que l'algorithme C n'est plus utilisable. Par contre, A et B restent applicables.

Maintenant, avec 100 éléments, il faut respectivement 8000 et 100000 opérations. Bien évidemment, l'algorithme A est le plus performant, mais l'algorithme B reste applicable. Cet exemple justifie la notion d'efficacité.

REMARQUE : **Si le nombre d'opérations "n'explose pas" avec une augmentation de la taille des données, l'algorithme est considéré efficace.**

Problème d'optimisation combinatoire, de reconnaissance

- Problème d'optimisation combinatoire :

Un problème d'optimisation combinatoire est un problème qui consiste à chercher une meilleure solution parmi un ensemble de solutions réalisables.

- Problème de reconnaissance :

Un problème de reconnaissance est un problème qui consiste à apporter une réponse "oui" ou "non" à une question.

Un problème d'optimisation combinatoire est au moins aussi difficile que le problème de reconnaissance associé, cela signifie qu'un problème d'optimisation combinatoire est souvent du même niveau de difficulté que le problème de reconnaissance associé.

Problème Facile, Difficile

Tout d'abord, on fait une distinction entre les problèmes décidables et les problèmes indécidables.

- Problème Indécidable :

Les problèmes indécidables sont ceux pour lesquels aucun algorithme, quel qu'il soit, n'a été trouvé pour les résoudre.

- Problème Décidable :

les problèmes décidables sont ceux pour lesquels il existe au moins un algorithme pour les résoudre.

Catégories de problèmes [Les classes]

On distingue 4 classes de problèmes

- La classe NP :

Parmi les problèmes décidables, les plus simples à résoudre sont regroupés dans la classe NP.

Un problème appartient à la classe NP si quelqu'un ayant la solution au problème peut démontrer que c'est la solution en un temps polynomial.

La classe NP est également décomposée en trois catégories.

- La classe P :

La classe P, qui regroupe les problèmes les plus simples de la classe NP, contient les problèmes pour lesquels on connaît au moins un algorithme polynomial pour les résoudre.

Catégories de problèmes [Les classes]

La notion de la Réduction Polynomiale :

Soit deux problèmes P1 et P2. On dit que P1 est réduit au problème P2 si on peut résoudre P1 en utilisant un algorithme pour P2 comme sous-routine. Cette réduction est dite polynomiale si l'algorithme pour P1 est polynomial en comptant l'appel à la sous-routine de P2 comme une opération élémentaire.

Catégories de problèmes [Les classes]

- La classe NP-Complet :

La classe NP-Complet regroupe les problèmes les plus difficiles de la classe NP. Elle contient les problèmes de la classe NP tels que n'importe quel problème de la classe NP leur est polynomialement réductible.

- La classe NP-Difficile :

La classe NP-Difficile regroupe les problèmes (pas forcément dans la classe NP) tels que n'importe quel problème de la classe NP leur est polynomialement réductible.

Catégories de problèmes [Les classes]

➔ Le Tableau récaputilatif

<table>
<tr><td rowspan="4">Décidables</td><td rowspan="3">Classe NP</td><td>Classe P
(plus court chemin, arbre de poids min, flot maximum, flot de coût minimum,...)</td><td rowspan="3">Classe NP-Difficile</td></tr>
<tr><td></td></tr>
<tr><td>Classe NP-Complet
(Voyageur de commerce, Sac à dos,...)</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Indécidables</td><td></td><td></td><td></td></tr>
</table>

Conclusion

Les machines ont des limites théoriques : elles ne peuvent pas résoudre TOUS les problèmes. Néanmoins, elles en résolvent un grand nombre !

☺Merci pour votre attention☺