Complexité Algorithmique

Dr Ahmed Wade

awade@ept.sn

Sommaire

Machine de Turing

- Définit par Alan Turing (1936)
- Modélise le fonctionnement des appareils mécaniques de calcul.
- Donne une définition précise au concept d'algorithme
  - Thèse de Church-Turing : « tout traitement réalisable mécaniquement peut être accompli par une machine de Turing »
- Fournit un modèle théorique pour étudier la complexité des problèmes

Machine de Turing

- Elle est constituée de
  - Un ruban infini à droite et à gauche
    - Chaque case contient un symbole parmi un alphabet fini
  - Une tête de lecture/écriture qui peut lire et écrire des symboles sur le ruban et se déplacer vers la gauche ou vers la droite
  - Un registre d'état qui mémorise l'état courant de la machine
    - Le nombre d'états est fini et il existe un état de départ
  - Une table d'actions

Machine de Turing

- La table d'action associe une (ou plusieurs) action(s) à chaque couple (état, caractère lu)
- Une action indique à la machine quel caractère écrire sur le ruban et comment déplacer la tête de lecture/écrire
- Tant qu'il existe une action à appliquer, la machine l'applique, sinon elle s'arrête

Machine de Turing

Machine de Turing déterministe

On parle de machine de Turing déterministe si chaque couple (état, caractère lu) de la table d'action est associée une et une seule transition possible

Machine de Turing non déterministe

On parle de machine de Turing non déterministe si à chaque couple (état, caractère lu) est associées une ou plusieurs transitions possibles.

Sommaire

Notations et simplification

- Le calcul des complexités est "complexe"
- Pour les même raisons que la terminaison
    - Il n'existe pas de méthode générale (ou algorithme) pour calculer la complexité d'un algorithme encore moins celle d'un problème
- Nous montrerons comment évaluer la fonction complexité d'un algorithme
    - en évaluant non pas précisément cette fonction
    - mais en évaluant la classe à laquelle elle appartient

Complexité en temps

- Soit A un algorithme qui résout un problème P
- La complexité en temps de l'algo. A est
  - une fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le nombre maximum d'étapes de calcul de A sur toutes les entrées de taille n
- On dit que A s'exécute en temps $f(n)$
- Idem pour la complexité en espace

Rappel : Notation Landau

- Soit deux fonctions f, g : $N \rightarrow R^+$, on dit que
  - f(x) = O(g(x)) (quand x tend vers ∞) s'il existe deux nombres réels c et x_0 tels que pour tout $x \geq x_0$

$$f(x) \leq c \cdot g(x)$$

- Exemple : $3x^4 + 2x^3 + x + 4 = ?$

Rappel : Notation Landau

- Soit deux fonctions f, g : $N \to R^+$, on dit que
  - f(x) = O(g(x)) (quand x tend vers ∞) s'il existe deux nombres réels c et x_0 tels que pour tout $x \geq x_0$

$$f(x) \leq c \cdot g(x)$$

- Exemple : $3x^4 + 2x^3 + x + 4 = \quad O(x^4)$
- Pour les polynômes, il suffit de garder le terme de plus grand ordre et supprimer son coeff.

Complexité en temps

- Un algorithme A résout un problème P en temps $O(T(n))$ si, pour tout n et pour toute instance de taille n, l'algorithme retourne une solution admissible après $O(T(n))$ étapes

Exercice

Considérant qu'une opération élémentaire se fasse en 10^{-8} seconde (un milliardième de seconde), rappelant qu'une année comporte à peu près 3.10^7 seconde. Calculer le temps nécessaire à l'exécution d'un algorithme sur une entrée de taille 100 de complexité :

1. n, $2n$,
2. $log(n)$, $3log(n)$
3. n^2, $2n^2$,
4. n^3,
5. 2^n,
6. $n!$,

Type de complexité

Notation	Type de complexité	Pour n = 100 avec 10^8 ops/sec
$O(1)$	constante	Temps constant
$O(\log(n))$	logarithmique	10^{-7} secondes
$O(n)$	linéaire	10^{-6} secondes
$O(n\log(n))$	quasi-linéaire	10^{-5} secondes
$O(n^2)$	quadratique	10^{-4} secondes
$O(n^3)$	cubique	10^{-2} secondes
$O(n^p)$	polynomiale	11 jours si $p = 7$
$O(2^n)$	exponentielle	10^{14} années
$O(n!)$	factorielle	10^{142} années

Classes de complexité (1)

- $O(1)$: complexité constante, pas d'augmentation du temps d'exécution quand le paramètre croit
- $O(log(n))$: complexité logarithmique, augmentation très faible du temps d'exécution quand le paramètre croit.
  - Exemple : algorithmes qui décomposent un problème en un ensemble de problèmes plus petits (dichotomie)
- $O(n)$: complexité linéaire, augmentation linéaire du temps d'exécution quand le paramètre croit (si le paramètre double, le temps double)
  - Exemple : algorithmes qui parcourent séquentiellement des structures linéaires

Classes de complexité (2)

- $O(nlog(n))$: complexité quasi-linéaire, augmentation un peu supérieure à $O(n)$
- $O(n^2)$: complexité quadratique, quand le paramètre double, le temps d'exécution est multiplié par 4
  - Exemple : algorithmes avec deux boucles imbriquées
- $O(n^i)$: complexité polynomiale, quand le paramètre double, le temps d'exécution est multiplié par 2^i
  - Exemple : algorithme utilisant i boucles imbriquées

Classes de complexité (3)

- $O(2^n)$: complexité exponentielle, quand le paramètre double, le temps d'exécution est élevé à la puissance 2
- $O(n!)$: complexité factorielle, asymptotiquement équivalente à n^n

Exercice 1

Algorithm 1 Recherche séquentielle

```
fonction rechercheElement(entier[] tab, entier x) : booléen
    entier i
    i← 0
    n ← tab.longueur
    tant que i < n faire
        si tab[i] = x alors
            retourne VRAI
        fin si
        i ← i+1
    fin tant que
    retourne FAUX
fin fonction
```

Exercice 2

Algorithm 2 Recherche dichotomique

```
fonction rechercheElement(entier[] tab, entier x) : booléen
    i← 0, j← tab.longueur-1
    tant que i ≤ j faire
        si tab[(j+i)/2] = x) alors
            retourne VRAI
        sinon
            si tab[(j+i)/2] > x alors
                j ← (j+i)/2 - 1
            sinon
                i ← (j+i)/2 +1
            fin si
        fin si
    fin tant que
    retourne FAUX
```

Exercice 3

Algorithm 3 Tri à bulles

```
fonction triBulle(entier[] tab)
    entier i, j, n=taille tab
    pour i allant de n-1 à 1 faire
        pour j allant de 0 à i-1 faire
            si tab[j] > tab[j+i]  alors
                échanger (tab[j+i], tab[j])
            fin si
        fin pour
    fin pour
fin fonction
```

Notations Ω et Θ

- Soit deux fonctions f, g : $N \rightarrow R^+$, on dit que
  - f(x) $= \Omega(g(x))$ (quand x tend vers ∞) s'il existe deux nombres réels c et x_0 tels que pour tout $x \geq x_0$
  $$f(x) \geq c \cdot g(x)$$
  - Si f(x) $= \Omega(g(x))$ et f(x) $=$ O(g(x)) alors f(x) $= \Theta(g(x))$

Sommaire

La classe P

- La classe P est l'ensemble des problèmes qui peuvent être résolus en temps polynomial par une machine de Turing déterministe
- Elle correspond en pratique à l'ensemble des problèmes qui sont traitables en un temps raisonnable par un ordinateur

Exemples de problèmes de la classe P

- Ce sont tous les problèmes que l'on peut résoudre sur une machine réelle en temps polynomial
  - PGCD de deux entiers
  - Tri d'un tableau
  - Recherche d'un élément dans un tableau ordonné
  - ...

La classe NP

- La classe NP est l'ensemble des problèmes qui peuvent être résolus en temps polynomial par une machine de turing non déterministe
- NP est une abréviation pour Nondeterministic Polynomial

Exemples de problèmes de la classe NP (1)

- Somme d'un sous ensemble
  - Donnée : un ensemble de n entiers
  - Sortie : existe-il un sous ensemble dont la somme de ses éléments est égal à B ?
- Exemple : supposer l'ensemble $\{4, 5, 8, 13, 15, 24, 33\}$
  - $B = 36$
  - $B=14$

Exemples de problèmes de la classe NP (2)

- SAT : satisfiabilité d'une expression Booléenne
  - Donnée : n, un entier, et Φ une expression booléenne avec n variables booléennes, $x_1, x_2, \cdots, x_n$
  - Sortie : oui, si Φ est satisfiable, i.e. il existe une valuation v telle que $v(\Phi) = \text{Vrai}$
- Exemple : si $\Phi = (x_1 + x_2 + x_3)(x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})$

Exemples de problèmes de la classe NP (2)

- SAT : satisfiabilité d'une expression Booléenne
  - Donnée : n, un entier, et Φ une expression booléenne avec n variables booléennes, $x_1, x_2, \cdots, x_n$
  - Sortie : oui, si Φ est satisfiable, i.e. il existe une valuation v telle que $v(\Phi) = \text{Vrai}$
- Exemple : si $\Phi = (x_1 + x_2 + x_3)(x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})$
  - Φ est elle satisfiable ?

Exemples de problèmes de la classe NP (2)

- SAT : satisfiabilité d'une expression Booléenne
  - Donnée : n, un entier, et Φ une expression booléenne avec n variables booléennes, $x_1, x_2, \cdots, x_n$
  - Sortie : oui, si Φ est satisfiable, i.e. il existe une valuation v telle que $v(\Phi) = $ Vrai
- Exemple : si $\Phi = (x_1 + x_2 + x_3)(x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})$
  - Φ est satisfiable : $x_1 = vrai$, $x_3 = faux$

26

Exemples de problèmes de la classe NP (2)

- SAT : satisfiabilité d'une expression Booléenne
  - Donnée : n, un entier, et Φ une expression booléenne avec n variables booléennes, $x_1, x_2, \cdots, x_n$
  - Sortie : oui, si Φ est satisfiable, i.e. il existe une valuation v telle que $v(\Phi) = Vrai$
- Exemple : si $\Phi = (x_1 + x_2 + x_3)(x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})$
  - Φ est satisfiable : $x_1 = vrai$, $x_3 = faux$
- si $\Phi = (x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})(\bar{x_1} + x_4)$

Exemples de problèmes de la classe NP (2)

- SAT : satisfiabilité d'une expression Booléenne
  - Donnée : n, un entier, et Φ une expression booléenne avec n variables booléennes, $x_1, x_2, \cdots, x_n$
  - Sortie : oui, si Φ est satisfiable, i.e. il existe une valuation v telle que $v(\Phi) = \text{Vrai}$
- Exemple : si $\Phi = (x_1 + x_2 + x_3)(x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})$
  - Φ est satisfiable : $x_1 = \mathit{vrai}$, $x_3 = \mathit{faux}$
- si $\Phi = (x_1 + x_4)(\bar{x_1} + \bar{x_3})(x_1 + \bar{x_4})(\bar{x_1} + x_4)$
  - Φ est elle satisfiable ?

Exemples de problèmes de la classe NP (3)

- Un cycle hamiltonien d'un graphe G est un cycle qui passe une et une seule fois par tous les sommets de G
- Problème du Cycle Hamiltonien
  - Étant donné un graphe G, le graphe G possède-t-il un cycle hamiltonien ?

Exemples de problèmes de la classe NP (4)

- Partition en trinômes
  - Donnée : une liste de n étudiants, et pour chaque étudiant la liste de ceux avec qui il accepte de travailler ;
  - Sortie : oui, si on peut répartir les étudiants en trinômes qui respectent les choix des étudiants.

P vs. NP

- P est la classe des problèmes dont on peut rapidement trouver une solution (rapidement = en temps polynomial)
- NP est la classe des problèmes dont on peut rapidement vérifier qu'une solution est bien une solution
- Une des plus grandes questions en informatique est la suivante : les classes P et NP sont-elles distinctes ?

Voulez-vous gagner 1 000 000 $

- Voulez-vous gagner 1 000 000 $?
  - Il "suffit" de démontrer la conjecture suivante
    - $P \neq NP$
  - ou bien de prouver que P = NP
  - `www.claymath.org/millennium-problems/p-vs-np-problem`

Problèmes NP-Complets

- Un progrès important pour résoudre la question (P vs. NP) a été fait par Stephen Cook et Leonid Levin dans les années 1970
- Ils ont découvert certains problèmes de NP dont la complexité du problème est liée à la complexité de la classe NP entière
- Si un algorithme polynomial existe pour un de ces problèmes, alors tous les problèmes de NP peuvent être résolus en temps polynomial
- Ces problèmes sont dits NP-complets

Problèmes NP-Complets

- Le premier problème qui a été montré NP-complet est le problème de satisfabilité (SAT)
- Une liste de problèmes NP-complet
  - https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Exemple : une fonction pire que la factorielle

La fonction d'Ackermann se définit de la manière suivante, pour tous entiers m et n positifs

Algorithm 4 Ackermann

```
fonction Ackermann(m,n)
    entier m, n
    si m==0 alors
        retourne n+1
    sinon
        si n==0 alors
            retourne Ackermann(m-1,1)
        sinon
            retourne Ackermann(m-1, Ackermann(m, n-1))
        fin si
    fin si
fin fonction
```

Exemple : la fonction d'Ackermann

- Calculer Ackermann(2,3)
  -
- Calculer Ackermann(5,0)
  -
- Calculer Ackermann(4,2)
  -
- Calculer Ackermann(4,4) ?

Exemple : la fonction d'Ackermann

- Calculer Ackermann(2,3)
  - Ackermann(2,3)=9
- Calculer Ackermann(5,0)
  -
- Calculer Ackermann(4,2)
  -
- Calculer Ackermann(4,4) ?

Exemple : la fonction d'Ackermann

- Calculer Ackermann(2,3)
  - Ackermann(2,3)=9
- Calculer Ackermann(5,0)
  - Ackermann(5,0)= Ackermann(4,1)=65533
- Calculer Ackermann(4,2)
  -
- Calculer Ackermann(4,4) ?

Exemple : la fonction d'Ackermann

- Calculer Ackermann(2,3)
  - Ackermann(2,3)=9
- Calculer Ackermann(5,0)
  - Ackermann(5,0)= Ackermann(4,1)=65533
- Calculer Ackermann(4,2)
  - Ackermann(4,2)=$2^{65536} - 3$
- Calculer Ackermann(4,4) ?