Complexité Temporelle

Problème

Décision ; Recherche ; Optimisation ; Énumération ; Dénombrement

Modélisation possible

Soit ∏ un problème

(a) I ∈ D(∏) : instance

(b) $S_{\prod}(I)$: ensemble des solutions de l'instance I pour le problème ∏

Format général

Nom :

Donnée :

Question :

Exemples

Nom : CNF-SAT

Données : V un ensemble de variables booléennes, C un ensemble de clauses sur V

Question: Existe-t-il un assignement sur V rendant C vrai ?

Nom : Circuit Eulérien

Donnée : R une relation binaire

Question : Donnez, s'il existe, un circuit Eulérien dans R.

Nom : Stable-Max

Données : G=(X,E) un graphe non orienté

Question: Trouvez un stable de taille maximum dans G.

Nom : All-kStable

Données : G=(X,E) un graphe non orienté, k ∈ IN.

Question: Donnez tous les stables de taille k dans G.

Nom : Max-2SAT

Données: V un ensemble de variables booléennes, C un ensemble de clauses, de taille 2, sur V.

Question: Quel est le nombre maximum de clauses satisfiables par un assignement ?

Algorithme déterministe

Définition : Un algorithme déterministe est un ensemble d'opérations de calculs élémentaires organisé suivant des règles déterministes dans le but de résoudre un problème donné. Cet ensemble est exécutable dans un modèle de calculabilité, et à toute donnée est associé une réponse en temps fini.

Machine de Turing **$M=(Q, \sum, \delta, q_0, q_f)$**
Q : {état}, $\#Q < +\infty$;
$\sum$: alphabet, $\#\sum < +\infty$;
$q_0 \in Q$; $q_f \in Q$;
$\delta : Q \times \sum \rightarrow Q \times \sum \times \{R,N,L\}$

pour k >1 **δ :** **$Q \times \sum^k \rightarrow Q \times \sum^{k-1} \times \{R,N,L\}$**

non déterministe **δ :** **$Q \times \sum \rightarrow \wp(Q \times \sum \times \{R,N,L\})$**

Configuration (Snapshot) : (q, x) $x \in \sum^*\#\sum^*$; $(q_0, \#w)$; (q_f, x) avec $x = w_1\#w_2$
$(q_0, \#w, \#, \ldots, \#)$

Définition : Une fonction f est **T-calculable** si et seulement s'il existe une machine de Turing MT qui la réalise : Domaine(f) = langage reconnu par MT.

w **accepté** par MT si $(q_0, \#w) \vdash^*_{MT} (q_f, x)$
w **reconnu** par MT si $\exists n \in IN$, tel que $(q_0, \#w) \vdash^n_{MT} (q_f, x)$

Vision fonctionnelle avec $x = w_1\#w_2$ on obtient $f(w) = w_1w_2$

RAM ≡ T-calculable

Thèse de Church: Toute fonction mécaniquement calculable est Turing calculable.

Définition : Une machine résout un problème Π si et seulement si $\forall x \in D(\Pi)$, <u>en temps fini</u>, elle détermine si $x \in L_\Pi$.

Première Dichotomie

Problème décidable **Problème indécidable**

Problèmes indécidables

Théorème 1: Il existe une infinité de problèmes indécidables

Théorème 2: Le problème de l'arrêt d'une machine de Turing est indécidable

Définition :
Le problème Π_1 se réduit, **au sens de Turing**, au problème Π_2, ce que l'on note par $\Pi_1 <_{TM} \Pi_2$, si :

(1) $\exists\ \phi$: Données(Π_1) $\rightarrow$ Données(Π_2) une application telle que : $w \in L_{\Pi 1} \Leftrightarrow \phi(w) \in L_{\Pi 2}$
(2) $\exists$ TM qui s'arrête sur toute donnée de Π_1 et qui réalise ϕ

Proposition 1 : $\Pi_1 <_{TM} \Pi_2$ et Π_2 décidable $\Rightarrow \Pi_1$ décidable

Corollaire 1.1 : $\Pi_1 <_{TM} \Pi_2$ et Π_1 indécidable $\Rightarrow \Pi_2$ indécidable

Exemple : Convergence d'une suite sur Z

Nom : CSZ
Données : Une suite d'éléments de Z définie récursivement.
Question : Cette suite est-elle convergente ?

Proposition 2 : Le problème « Convergence d'une suite sur Z » est indécidable.

Problèmes décidables

Algorithme : Analyse et Comparaison

Critères qualitatifs : Arrêt ; Correction
Critères quantitatifs : Taille ; Lisibilité ; Coût

Classes de fonctions

$O(f)=\{g: g\in IR^{IR} : \exists\, c\in IR_+^*,\ \exists\, x_0\in IR, \forall x \geq x_0, 0 \leq g(x) \leq c \times f(x)\ \}$

O_∞ ; $O_{p.s.}$; $O_{p.p.}$

$o(f)=\{g: g\in IR^{IR} : \forall\, c\in IR_+^*,\ \exists\, x_0\in IR, \forall x \geq x_0, 0 \leq g(x) < c \times f(x)\ \}$

$\Omega(f)=\{g: g\in IR^{IR} :\exists\, c\in IR_+^*,\ \exists\, x_0\in IR, \forall x \geq x_0, 0 \leq c \times f(x) \leq g(x)\ \}$

$\omega(f)=\{g: g\in IR^{IR} :\forall\, c\in IR_+^*,\ \exists\, x_0\in IR, \forall x \geq x_0, 0 \leq c \times f(x) < g(x)\ \}$

$\Theta(f)=\{g: g\in IR^{IR} :\exists\, c_1,c_2\in IR_+^*,\ \exists\, x_0\in IR, \forall x \geq x_0, 0 \leq c_1 \times f(x) \leq g(x) \leq c_2 \times f(x)\ \}$

Exemples : $\Sigma^n_{i=1}\ i^k \in \Theta(n^{k+1})$; $lg(n!) \in \Theta(nlg(n))$; $\Sigma^n_{i=1}\ 1/i \in \Theta(lg(n))$

Règle de l'Hôpital

Soient f,g dérivables telles que $\lim_{x\to+\infty} f(x) = +\infty$ et $\lim_{x\to+\infty} g(x) = +\infty$.

Si $\lim_{x\to+\infty} f'(x)/g'(x)$ existe ($\{+\infty,-\infty\}\cup IR$), alors on obtient que $\lim_{x\to+\infty} f(x)/g(x) = \lim_{x\to+\infty} f'(x)/g'(x)$

Remarque : valable avec $x \to a$ pour $a \in \{+\infty,-\infty\}\cup IR$ ($g'(x)$ ne doit pas s'annuler)

Somme discrète / continue

Soient a,b $\in$ IN et f : [a,b] $\to$ IR, continue et croissante : $\sum_{i=a}^{b-1} f(i) \leq \int_a^b f(x)dx \leq \sum_{i=a+1}^{b} f(i)$

Soient a,b $\in$ IN et f : [a,b] $\to$ IR, continue et décroissante : $\sum_{i=a+1}^{b} f(i) \leq \int_a^b f(x)dx \leq \sum_{i=a}^{b-1} f(i)$

Complexité temporelle

Définition :

La complexité temporelle d'un algorithme A est l'application T_A : IN $\to IR^+$, qui à n $\to$ #opérations élémentaires effectuées par A pour donner une réponse aux données de taille n

Attention :

Taille : codage mémoire ;
Élémentaire : modèle de « machine » (RAM ; Tris ; …)
: différentes mesures de complexité

Définition : Soit $c_A(x)$ le coût temporel (spatial) de l'algorithme A sur la donnée x. La compléxité temporelle (spatiale) de A est alors :

- dans **le meilleurs des cas** est : $C_A(n) = \min_{x \in \text{Donnée}(A)} \{c_A(x) : |x| = n\}$
- dans **le pire des cas** est : $C_A(n) = \max_{x \in \text{Donnée}(A)} \{c_A(x) : |x| = n\}$
- en **moyenne** est : $C_A(n) = \sum_{x \in \text{Donnée}(A),\ |x| = n} p_n(x)*c_A(x)$

  avec $p_n(.)$ une mesure de probabilité sur les données de taille n

Tris par permutations

$x_1, \ldots\ldots\ldots\ldots, x_n,$ On débute en x_n Si $x_i > x_{i+1}$ alors on permute x_i et x_{i+1}

meilleurs des cas $C_A(n) = 0$;
meilleurs des cas $C_A(n) = n(n-1)/2$;
en moyenne $C_A(n) = n(n-1)/4$ **Attention**

Coût amorti Potentiel

	O_1		O_n
Structure :	S_1	…………….	S_n

Définition : Soit h une fonction « potentiel » telle que $h(x) \geq 0$, alors le coût amorti est $c_a(O_i) = c(O_i) + h(S_i) - h(S_{i-1})$

Proposition : Si $h(S_n) - h(S_0) \geq 0$ alors $C_a(n) = \Sigma^n_{i=1}\ c_a(O_i) \geq \Sigma^n_{i=1}\ c(O_i) = C(n)$

Borne inférieure

Optimalité

$\forall\ \Pi$ on a : $T_\Pi(.) \in \Omega(\max(n,s(.)))$, de plus $\forall$ A résolvant Π on a : $T_\Pi(.) \in O(T_A(.))$

Transformation

Π_1 se transforme en Π_2 en temps f(.) si

(1) $\phi_D : \text{Dom}(\Pi_1) \to \text{Dom}(\Pi_2)$: Calculable en $C_D(|x|)n$ et $|\phi_D(x)| \in O(|x|)$
(2) $\phi_S : \text{Sorties}(\Pi_2) \to \text{Sorties}(\Pi_1)$: Calculable en $C_S(|x|)$ et $|\Pi_2(x)| \in O(|x|)$
(3) $C_D(.) + C_S(.) \in O(f(.))$

Proposition :
Si Π_1 est transformable en Π_2 en temps f(.) alors (i) $T_{\Pi_1} \in O(T_{\Pi_2} + f)$, et (ii) $T_{\Pi_2} \in \Omega(T_{\Pi_1} - f)$

Tris par comparaisons

Test : $x_i \leq x_j$; Arbre binaire de décision : une exécution ≡ un chemin de r à une feuille

Lemme : A arborescence binaire avec n! feuilles alors $h(A) \geq \lceil n \times \ln(n) \rceil - n$

Corollaire : Tout algorithme de tris <u>par comparaisons</u> est en $\Omega(n \times \lg(n))$

Deuxième Dichotomie

la classe P les classes NP et PSpace

Définitions : Soit Π un problème de décision

(1) $\Pi \in P$ si $\exists$ algorithme <u>déterministe</u> Polynomial le résolvant

(2) $\Pi \in NP$ si $\exists$ algorithme <u>Non déterministe</u> Polynomial le résolvant

Tables de complexité temporelle

M.R. Garey et D.S. Johnson 1979

Computer and Intractability : A Guide to the Theory of NP-Completness

	10	20	30	40	50	60
n	.00001’’	.00002’’	.00003’’	.00004’’	.00005’’	.00006’’
n^2	.0001’’	.0004’’	.0009’’	.0016’’	.0025’’	.0036’’
n^3	.001’’	.008’’	.027’’	.064’’	.125’’	.216’’
n^5	.1’’	3.2’’	24.3’’	1.7’	5.2’	13’
2^n	.001’’	1’	17.9’	12.7j	35.7a	366s
3^n	.059’’	58’	6.5a	3855s	$2x10^8$s	$1.3x10^{13}$s

’’ : seconde ; ’ : minute ; j : jour ; a : année ; s : siècle

fonction	actuel	100x	1000x
n	N1	100xN1	1000xN1
n^2	N2	10xN2	31.6N2
n^3	N3	4.64xN3	10xN3
n^5	N4	2.5xN4	3.98xN4
2^n	N5	N5+6.64	N5+9.97
3^n	N6	N6+4.19	N6+6.29

La classe P et la classe PSpace

Soit A un algorithme :

- On pose $\mathbf{t_A(n)}$ = max du temps pris par A sur les données de taille n.
  $\mathbf{s_A(n)}$ = max de l'espace pris par A sur les données de taille n.
- Pour f : IR → IR, on pose alors
  Dtime(f(n)) = $\{L \subseteq \Sigma^*$: $\exists$ <u>algo déterministe</u> A reconnaissant L et tel que $t_A(n) \in O(f(n))\}$
  Dspace(f(n)) = $\{L \subseteq \Sigma^*$: $\exists$ <u>algo déterministe</u> A reconnaissant L et tel que $s_A(n) \in O(f(n))\}$

Définition :

La **classe P** est l'ensemble $\cup_{k\geq 0}$ Dtime(n^k).

La **classe PSpace** est l'ensemble $\cup_{k\geq 0}$ Dspace(n^k).

Non-déterminisme

Entrée : G=(X,E) un graphe non orienté

(i) Choisir $S \subseteq X$

(ii) Vérifier si S forme un stable

Entrée : $G=(X,E)$ et $k \in N$

(i) Choisir $S \subseteq X$

(ii) **Si** ($|S| \geq k$) et ($\forall e \in E$, $|e \cap S| \leq 1$) **Alors** Arrêt(oui) **Sinon** Arrêt(non)

Définition : Un algorithme A non déterministe reconnaît le langage L lorsque :

- si $x \in L$, alors $\exists$ suite de « vœux » faite par A qui génère la réponse « oui »
- si $x \notin L$, alors A répond toujours non

On pose alors

$\mathbf{t_A(n)} = \max_{\omega : |\omega|=n} \min_{\text{suite de vœux } y}$ temps pris par A sur la donnée ω avec la suite de vœux y.

NTime(f(n))$=\{L \subseteq \Sigma^* : \exists$ algo non déterministe A reconnaissant L et tel que $t_A(n) \in O(f(n))\}$

Définition : La **classe NP** est l'ensemble $\cup_{k \geq 0}$ Ntime(n^k)

1ère phase : Oracle ; **2ème phase :** calcul déterministe

Définition : $L \in NP \Leftrightarrow \exists\ P(.,.) \in P$ et $\exists\ c,k \in IN$ tq $\forall x \in \Sigma^*$: $(x \in L) \Leftrightarrow (\exists\ y,\ |y| \leq c.|x|^k$ et $P(x,y))$

Entrée : $x \in \Sigma^*$:

(i) Choisir y ;

(ii) **Si** $P(x,y)$ **Alors** Arrêt(oui) **Sinon** Arrêt(non)

La classe NP

Définition : $L \in NP \Leftrightarrow (x \in L \Leftrightarrow \exists^P y,\ P(x,y))$

Propriété : $P \subseteq NP$; $P \subseteq Pspace$; $NP \subseteq Pspace$

Problème NP-Complet

Nom : CNF-SAT
Données : V ensemble de variables booléennes, C ensemble de clauses sur V
Question : Existe-t-il un assignement sur V rendant C vrai ?

Définition : Π est **NP-Complet** si $\Pi \in NP$ et si la résolution de Π par un algorithme déterministe polynomial entraîne P = NP.

Théorème (Cook 1971) SAT est NP-Complet

Nom : Exact-3-SAT
Données : V ensemble de variables booléennes, C ensemble de clauses ayant toutes 3 littéraux.
Question : Existe-t-il un assignement sur V rendant C vrai ?

Définition :
Le problème Π_1 se réduit, **au sens de Karp**, au problème Π_2, ce que l'on note par $\Pi_1 <_{Karp} \Pi_2$, si
(1) $\exists\ \phi$: Données(Π_1) $\rightarrow$ Données(Π_2) application polynomiale, et
(2) $x \in L_{\Pi 1} \Leftrightarrow \phi(x) \in L_{\Pi 2}$

Définition : $\Pi \in$ **NP-Complet** si (i) $\Pi \in$ NP, et (ii) $\forall\ \Pi_1 \in$ NP, on a $\Pi_1 <_{Karp} \Pi$

Définition : Π admet un **certificat polynomial**, si $\forall\ I \in D(\Pi)$, $(I \in L_\Pi) \Leftrightarrow (\exists^P y, A(I,y)=\text{Vrai})$: avec A(a,b) algorithme polynomial en #a, #b

Proposition :
Un problème de décison $\Pi \in$ NP-Complet si
(1) Π admet un certificat polynomial, et
(2) $\exists\ \Pi_1 \in$ NP-Complet tel que $\Pi_1 <_{Karp} \Pi$

Théorème Exact-3-SAT est NP-Complet

Question P=NP ou P≠NP : Frontière « ténue »

Nom	:	**Plus long chemin**
Données	:	$G=(X,E,v)$, $v : E \rightarrow \mathbb{N}$, $k \in \mathbb{N}$ et $a,b \in X$
Question	:	Existe-t-il une ab-chaîne de poids $\geq k$?

NP-Complet même lorsque v est une fonction constante

Nom	:	**Plus court chemin**
Données	:	$G=(X,E,v)$, $v : E \rightarrow \mathbb{N}$, $k \in \mathbb{N}$ et $a,b \in X$
Question	:	Existe-t-il une ab-chaîne de poids $\leq k$?

Polynomial : Dijkstra $O(|E|+|X|\lg|X|)$ *Attention* *NP-Complet si* $v : E \rightarrow Z$ *et* $k \in Z$