Análisis y Complejidad de Algoritmos

Complejidad Computacional

Arturo Díaz Pérez

- Lenguajes formales
- Gramáticas formales
- Jerarquía de Chomski
- Teoría de la complejidad
  - Una desigualdad computacional
- Computabilidad

Tópicos

- ☞ Teoría de autómatas
- ☞ Teoría de computabilidad
- ☞ Teoría de complejidad
- ☞ ¿Como evadir la intratabilidad?

Lenguajes Formales

- Alfabeto
  - $\Sigma = \{ a_1, a_2, ..., a_n \}$, un conjunto finito de símbolos
- Cadenas de símbolos
  - $\Sigma^1 = \Sigma$
  - $\Sigma^2 = \Sigma \times \Sigma$, $s_1 s_2 = (s_1, s_2)$
  - $\Sigma^i = \Sigma^{i-1} \times \Sigma$, $s_1 s_2 ... s_i = (s_1, s_2, ..., s_i)$
  - λ, palabra vacía es tal que, $\lambda s = s \lambda = s$, $\Sigma^0 = \{ \lambda \}$

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$$

Lenguajes Formales

- Cerradura de Kleene
  - Diccionario
  - Cerradura Positiva $\Sigma^+ = \Sigma^* - \{\lambda\}$
- Lenguaje
  - Un lenguaje es $L \subseteq \Sigma^*$
  - Los autómatas finitos, autómatas de pila y máquinas de Turing son caracterizados por su capacidad para reconocer lenguajes

Gramáticas Formales

- Una gramática formal es una estructura
  - ← G = < ***N***, ***T***, **P**, *S*>, donde
    - ***N*** es un conjunto finito de símbolos llamados ***no terminales***
    - ***T*** es un conjunto finito de símbolos llamados ***terminales***
      - $V = \boldsymbol{N} \cup \boldsymbol{T}$, vocabulario
    - $S \in \boldsymbol{N}$, es el ***símbolo inicial***
    - $P \subset V^* \times V^*$, es un conjunto de ***reglas sintácticas*** o ***producciones*** de la forma
      - $(\alpha, \beta) \in P \Rightarrow \alpha \rightarrow \beta$, "$\alpha$ *deriva* β"

Gramáticas Formales

- G = < ***N***, ***T***, **P**, *S*>
  - ← Derivaciones
    - $(\alpha, \beta) \in P$ $\sigma, \tau \in V^*$, tales que, $\sigma = \sigma_1 \alpha \sigma_2$ y $\tau = \tau_1 \beta \tau_2$, entonces, aplicando la producción se dice que "τ *se sigue de* σ", $\sigma \rightarrow \tau$
    - "τ *se deriva de* σ", $\sigma \rightarrow^* \tau$, si $\exists\ \sigma_0, \sigma_1, \ldots, \sigma_n, \in V^*$, tales que, $\sigma_0 = \sigma$ y $\sigma_n = \tau$, $\forall\ i < n$, "σ_{i+1} *se sigue de* σ_i"

Gramáticas Formales

- El ***lenguaje generado*** por *G* es
  - $L(G) = \{\tau \in \boldsymbol{N}^* \mid \tau$ se deriva de S, $S \rightarrow^* \tau\}$
  - El conjunto de palabras con símbolos terminales que se derivan en *G* del símbolo inicial *S*

Jerarquía de Chomski

- Sea G = < ***N***, ***T***, **P**, *S*> una gramática
  - G es de ***tipo 0 o irrestricta*** si sus producciones son de la forma
    - $\alpha \rightarrow \beta$, donde $\alpha \in N^+$ y $\beta \in V^*$
  - G es de ***tipo 1 o sensible al contexto*** si sus producciones son de la forma
    - $\alpha A \gamma \rightarrow \alpha \beta \gamma$, donde α, β y $\gamma \in V^*$ y $A \in N$
  - G es de ***tipo 2 o libre del contexto*** si sus producciones son de la forma
    - $A \rightarrow \alpha$, donde $\alpha \in V^*$ y $A \in N$
  - G es de ***tipo 3 o regular*** si sus producciones son de la forma
    - $A \rightarrow \alpha B$, o
    - $A \rightarrow \alpha$ donde $\alpha \in T^*$ y $A,B \in N$

Jerarquía de Chomski

- El lenguaje se dice ser del mismo tipo que la gramática que lo genere
  - ← LR ⊂ LLB ⊂ LSB ⊂ LI

Lenguajes Regulares

- El lenguaje reconocido por un autómata finito A
  - ← $A = < \Sigma, Q, \delta, q_0, F >$, es
  - ← $L(A) = \{ s \in \Sigma^* \mid \delta^*(s, q_0) \in F \}$, donde
    - δ^* es la función de transición extendida, $\delta^* : \Sigma^* \times Q \rightarrow Q$
      - $\delta^*(q, a) = \delta (q, a), \ a \in \Sigma \}$
      - $\delta^*(q, as) = \delta^*(\delta (q, a), s), \ a \in \Sigma, s \in \Sigma^* \}$

- El lenguaje reconocido por un autómata finito se conoce como un ***lenguaje regular***

Lenguajes Regulares

- Un autómata finito también puede calcular una función
  - Sea $M = < \Sigma, \Psi, Q, \delta, \lambda >$ una máquina finita con salida
  - M calcula la función de T pasos
    - $f^{(T)}_M : Q \times \Sigma^T \rightarrow Q \times \Psi^T$,
    - $f(q_0, w_1, w_2, ..., w_T) = (y_1, y_2, ..., y_T)$

Lenguajes Libres del Contexto

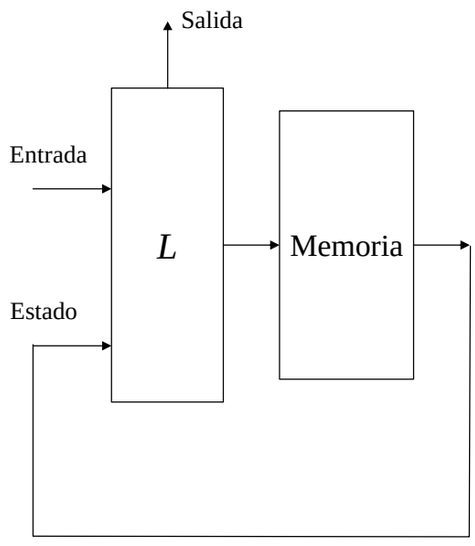
- El lenguaje reconocido por un autómata de pila AP
  - AP = $< \Sigma, \Gamma, Q, \delta, q_0, Z_0, F >$, es
  - $L(A) = \{ s \in \Sigma^* \mid \delta^*(q_0, s, Z_0) = (\tau, q_f)\ q_f \in F \}$, donde
    - δ^* es la función de transición extendida, $\delta^* : Q \times \Sigma^* \times \Gamma \rightarrow Q \times \Gamma$
  - El lenguaje reconocido por un autómata de pila no determinista se conoce como un ***lenguaje libre del contexto***

Lenguajes Libres del Contexto

- Ejemplo
  - G = < ***N***, ***T***, **P**, *S*>
    - ***N*** = { *S* }, ***T*** = {a, b}
    - **P** = { $S \rightarrow aSb$, $S \rightarrow ab$ }
  - Una derivación

$$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaSbbb \rightarrow aaaabbbb$$

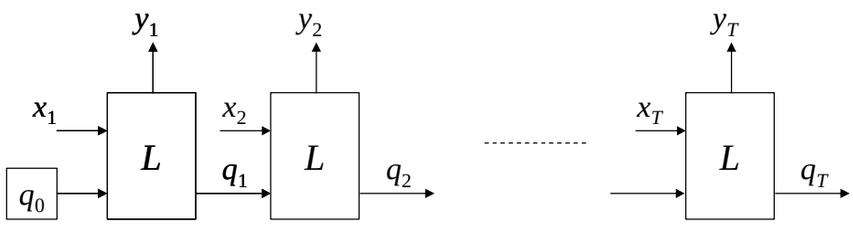
Máquinas de Estados Finitos



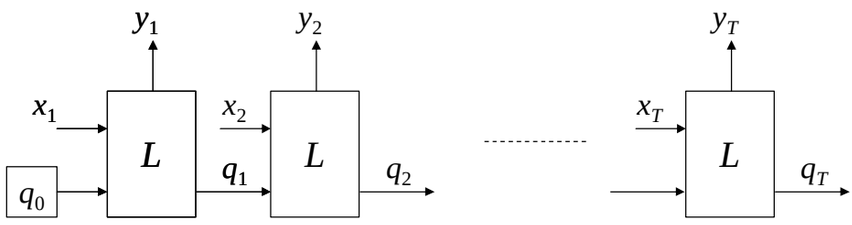
Una Desigualdad Computacional

- Sea f: $B^n \rightarrow B^m$, la función que calcula un autómata finito en T pasos
  - ← Su estado y T entradas externas contienen las n entradas Booleanas y sus T salidas contienen las m salidas Booleanas

- La función f podría ser calculada por el siguiente circuito



Una Desigualdad Computacional



Sea $C(L)$ el número de compuertas utilizadas para construir L

El circuito tiene $T \cdot C(L)$ compuertas

Ya que este circuito no es necesariamente el mejor para calcular f, entonces

$$C(f) \leq T \cdot C(L)$$

donde $C(f)$ es el tamaño del menor circuito para calcular f

$W = T \cdot C(L)$ es el trabajo computacional para calcular f

Una Desigualdad para RAM

- Considere una RAM con S bits de memoria
- Si S es grande, como lo es usualmente, $C(L)$ para la RAM es proporcional a S
- Por lo tanto,

$$C(f) \leq kS \cdot T$$

- ST, el producto espacio-tiempo, es una medida importante de complejidad de un problema
- Las funciones con un tamaño de circuito grande se pueden calcular por una RAM solo si tiene una gran capacidad de almacenamiento, o ejecuta muchos pasos o ambos.
- Existen resultados similares para máquinas de Turing

Teoría de la Complejidad

- Tratable vs. Intratable
  - Un *problema* es una cuestión general en donde existe
    - descripción de parámetros
    - descripción de la solución
  - Un *algoritmo* es un procedimiento paso por paso
    - una secuencia
    - un programa de computadora
    - un objeto matemático
  - Se busca siempre el algoritmo *más* eficiente para resolver un problema
    - el más rápido (lo más común)
    - el que ocupa menos memoria (algunas veces)
    - la eficiencia se expresa como función del tamaño del problema

El Problema del Agente Viajero

- Ejemplo: El problema del agente viajero
- Parámetros
  - conjunto de ciudades
  - conjunto de longitudes de caminos entre ciudades
- Solución
  - El recorrido (tour) más corto a través de las ciudades

El Problema del Agente Viajero

- ¿Cuál es la medida apropiada del tamaño del problema?
  - *m* nodos (ciudades)
  - *m*(*m*+1)/2 distancias
  - Se debe usar una codificación del problema
    - alfabeto o símbolos
    - cadenas: *abcd*//10/5/9//6/9/3
- Medidas
  - *Tamaño del problema*: longitud de la codificación
  - *Complejidad en tiempo*: cuanto tiempo tarda el algoritmo en función del tamaño del problema

Complejidad en Tiempo

- ¿Qué es un problema tratable?
  - ← Un ***algoritmo de tiempo polinomial*** es aquel cuya complejidad en tiempo es $O(p(n))$ para algún polinomio $p(n)$
  - ← Un *algoritmo de tiempo exponencial* es aquel que no puede ser acotado por una función polinomial (Ej. $n^{\log n}$)
  - ← Un algoritmo polinomial es ***tratable***
  - ← Un algoritmo exponencial es ***intratable***

Complejidad en Tiempo

- ¿Qué es un problema tratable?

	10	20	30	40	50	60
n	.00001 seg.	.00002 seg.	.00003 seg.	.00004 seg.	.00005 seg.	.00006 seg.
n^2	.0001 seg.	.0004 seg.	.0009 seg.	.0016 seg.	.0025 seg.	.00036 seg.
n^3	.001 seg.	.008 seg.	.027 seg.	.064 seg.	.125 seg.	.216 seg.
n^5	.1 seg.	3.2 seg.	24.3 seg.	1.7 min	5.2 min	13.0 min
2^n	0.001 seg	1.0 seg	17.9 min	12.7 días	35.7 años	366 siglos
3^n	0.059 seg	58 min	6.5 años	3855 siglos	$2x10^8$ siglos	$1.3x10^{13}$ siglos

Efecto de la Aceleración

- Efecto de la aceleración o mejoramiento de procesadores
  - Considere el tamaño máximo del problema que se puede resolver en una hora

	Actual	100 veces más rápido	1000 veces más rápido
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
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

"Un algoritmo eficiente para el problema del agente viajero"

- Respuestas posibles:
  - "*Claro, en la tarde está listo !*"
    - Todos los algoritmos conocidos verifican todos los posibles caminos
      - la búsqueda exhaustiva es exponencial
  - "*Probaré que no existe tal algoritmo*"
    - Probar intratabilidad es difícil
      - Muchos problemas importantes no tienen algoritmos tratables conocidos pero tampoco se ha probado su intratabilidad

"Un algoritmo eficiente para el problema del agente viajero"

- ←"*No puedo encontrarlo, soy incapaz de hacerlo*"
  - ☞No adecuada si se quiere conservar el empleo
- ←"*El problema es NP-completo. No puedo encontrar un algoritmo eficiente pero tampoco lo ha podido encontrar ninguna de estas personas famosas ...*"
  - ☞El problema es igual de difícil que otros problemas los cuales personas inteligentes no han podido resolver eficientemente
- ←"*Se puede encontrar un algoritmo que se comporte bien en la mayoría de los casos*"
  - ☞Encontrar una solución aproximada,
  - ☞Utilizar aleatorización, aplicar paralelismo, etc.

Computabilidad

- ☞ En la primera mitad del siglo 20 algunos matemáticos tales como, Alan Turing, Kurt Gödel y Alonzo Church descubrieron que algunos problemas no pueden ser resueltos por las computadoras
  - ←Ejemplo: "*Es esta proposición matemática cierta o falsa*"
- ☞ Se requiere tener modelos teóricos para las computadoras
  - ←Los modelos teóricos ayudan a la construcción de computadoras reales

No Decidibilidad

- No decidibilidad: problemas que no pueden ser resueltos por las computadoras
  - Ejemplos:
    - ¿Este programa corre para siempre?
    - ¿Es este programa correcto?
    - ¿Son equivalentes estos dos programas?
    - ¿Es este programa óptimo?
    - ¿Tiene una ecuación con una o más variables y coeficientes enteros (5x + 15y = 12) una solución entera?
      - El décimo problema de Hillbert

Tópicos

- Teoría de autómatas
  - ¿Qué es una computadora?
- Teoría de computabilidad
  - ¿Qué pueden hacer las computadoras?
- Teoría de complejidad
  - ¿Qué hace que algunos problemas sean computacionalmente difíciles y otros fáciles?
- ¿Como evadir la intratabilidad?
  - Aproximación
  - Aleatorización
  - Paralelismo