IN34A - Optimización

Complejidad

Leonardo López H.

lelopez@ing.uchile.cl

Primavera 2008

Contenidos

Problemas y Procedimientos de solución
- Problemas de optimización v/s problemas de decisión
- Métodos, Algoritmos y Heurísticas

Complejidad de Algoritmos y Problemas
- Consideraciones
- Máquina de Turing
- Complejidad de un Algoritmo
- Complejidad de un Problema

Referencias

Contenidos

Problemas y Procedimientos de solución
- Problemas de optimización v/s problemas de decisión
- Métodos, Algoritmos y Heurísticas

Complejidad de Algoritmos y Problemas
- Consideraciones
- Máquina de Turing
- Complejidad de un Algoritmo
- Complejidad de un Problema

Referencias

Problema

Problema

Un problema es un conjunto de instancias al cual corresponde un conjunto de soluciones a través de una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).

- Consideraremos dos tipos de problemas: problemas de *optimización* y problemas de *decisión*.

Problema de optimización

$$\begin{array}{ll} \text{mín} & f(x) \\ s.a. & x \in S \subseteq \mathbb{R}^n \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \text{mín} & f(x) \\ s.a. & g_i(x) \leq 0 \quad i = 1, ..., m \end{array}$$

- Donde $x = (x_1, ..., x_n) \in \mathbb{R}^n$ es la *variable de decisión*.
- $f : \mathbb{R}^n \to \mathbb{R}$ es la *función objetivo* y mide la calidad de las decisiones. Es una medida de efectividad.
- $S \subseteq \mathbb{R}^n$ representa el conjunto de decisiones factibles que forman las restricciones del problema.

Se busca, entre todas las decisiones factibles, la decisión $x^* \in S$ que determine el mejor valor de la función objetivo. x^* es la *solución óptima* del problema.

Problema de decisión I

Problema de Decisión

Un problema de decisión es un problema en donde las respuestas posibles son SI ó NO. Si existe solución al problema, la respuesta a la pregunta del problema es SI y en caso contrario NO..

- En general, un problema de decisión puede re-escribirse como un problema de optimización y viceversa.

Ejemplos:

1. Consideremos el problema del vendedor viajero.
   - **Problema de optimización:** dados el conjunto de ciudades, el conjunto caminos entre ciudades y los costos de utilizar cada camino, determinar un tour de costo mínimo.

Problema de decisión II

- **Problema de decisión:** dados el conjunto de ciudades, el conjunto caminos entre ciudades, los costos de utilizar cada camino y un entero no negativo k, ¿existe un tour de costo menor o igual que k?

2. Consideremos el problema de programación de trabajos. Se tiene un conjunto de n trabajos que deben ser procesados por m máquinas en forma secuencial de modo tal que el tiempo de procesamiento sea mínimo. Es decir, cada trabajo debe ser procesado primero por la máquina 1, luego por la 2, etc.

   - **Problema de optimización:** dados los tiempos de procesamiento de cada uno de los n trabajos en cada una de las m máquinas, determinar una secuencia para procesarlos que minimice el tiempo total de procesamiento.

Problema de decisión III

- **Problema de decisión:** dados los tiempos de procesamiento de cada uno de los n trabajos en cada una de las m máquinas y un entero no negativo k, ¿existe una secuencia tal que el tiempo total de procesamiento sea menor o igual que k?.

Instancia de un problema

- Una instancia de un problema se obtiene especificando valores particulares para todos sus parámetros.
- Consideremos el problema de invertir una matriz de m filas y m columnas.
  - Una instancia de este problema se obtiene especificando valores particulares para los parámetros m y cada coeficiente a_{ij} de la matriz.
- Para determinar el *tamaño* de una instancia, tipicamente se identifica una función de uno o más parámetros que permita tener una idea de la cantidad de memoria necesaria para representar el problema en su totalidad, para la instancia dada, en un computador.
  - Para el problema de invertir una matriz de m filas y m columnas, el tamaño de sus instancias queda bien representado por $m \cdot m$.

Métodos, Algoritmos y Heurísticas

Método

Secuencia de pasos, que puede ser finita o infinita, cuyo objetivo es determinar la solución de un problema.

Algoritmo

Secuencia finita de pasos que garantiza el encontrar una solución de un problema para una instancia dada.

Heurística

Procedimientos que en un tiempo razonable dan una buena aproximación a la solución del problema.

Contenidos

Problemas y Procedimientos de solución
- Problemas de optimización v/s problemas de decisión
- Métodos, Algoritmos y Heurísticas

Complejidad de Algoritmos y Problemas
- Consideraciones
- Máquina de Turing
- Complejidad de un Algoritmo
- Complejidad de un Problema

Referencias

Consideraciones I

- La dificultad de un problema está relacionada, por una parte, con la estructura del problema, y por otra, con el tamaño de la instancia.
  - Incluso si fijamos el tamaño de la instancia, todavía pueden haber diferencias en el tiempo requerido para la ejecución de un algoritmo. Por ejemplo, es más difícil ordenar la lista [3, 5, 2, 9, 1] que la lista [1, 3, 5, 6, 7], ya que la segunda ya está ordenada.
- La teoría de complejidad se centra en torno a problemas de decisión.
  - Como los problemas de optimización pueden escribirse como problemas de decisión (y viceversa), es posible usar la teoría de complejidad también en problemas de optimización.

Consideraciones II

- Se puede establecer el tiempo que tomará resolver un problema para una instancia dada, con un algoritmo o método particular, en función del tamaño de la instancia.
  - Para métodos secuenciales, aquellos que solo ejecutan una instrucción en cada instante del tiempo, se puede sustituir la medida del tiempo por una medida equivalente, como el número de operaciones elementales (sumas, multiplicaciones, operaciones lógicas básicas, etc).
- Es necesaria una forma de poder evaluar los algoritmos de forma independiente a la maquina sobre la que se ejecutan. Alan Turing planteo un *modelo conceptual* de computador en el cual ejecutar algoritmos.

Máquina de Turing Determinística

- La máquina de Turing Determinística es un modelo conceptual que ejecuta instrucciones básicas de forma secuencial basándose en reglas bien definidas.
  - Aproxima el comportamiento de un computador.
- En este modelo conceptual es posible ejecutar algoritmos y evaluar sus tiempos de ejecución, contando el número de instrucciones básicas que la máquina realiza hasta encontrar una solución de un **problema de decisión**.

Complejidad de un Algoritmo I

La **complejidad de un algoritmo** $f(n)$ está determinada por el tiempo máximo que requiere el algoritmo para determinar la solución de una instancia de tamaño n del problema.

- Es decir, se establece una cota superior de la complejidad de un algoritmo para un tamaño de instancia dado, puesto que existen muchas instancias del mismo tamaño y no todas requieren el mismo tiempo de ejecución. Este análisis corresponde al **análisis del peor caso**.
- Alternativamente podríamos analizar $f(n)$ bajo otras definiciones: en el caso promedio, en el mejor caso, etc.

Complejidad de un Algoritmo II

- Lo que realmente nos importa es como se comporta $f(n)$ para grandes valores de n. Es decir, nos interesa estudiar el comportamiento asintótico de las complejidades ($n \to \infty$).

Notación $\mathcal{O}$ (cota superior asintótica)

Dado $f, g : \mathbb{R}_+ \to \mathbb{R}$, decimos que $f(n) = \mathcal{O}(g(n)) \Leftrightarrow \exists k \in \mathbb{R}_+$, $n_0 \in \mathbb{N}$ tal que $f(n) \leq kg(n)\ \forall n \geq n_0$.

- $f(n) = \mathcal{O}(g(n))$ significa que f no crece más rápido que g, permitiendo definir un límite superior al crecimiento de la función f.

Complejidad de un Algoritmo III

Decimos que la complejidad $f(n)$ de un algoritmo es $\mathcal{O}(g(n))$, es decir del "orden de $g(n)$", si $f(n) = \mathcal{O}(g(n))$, donde g es una relación entre el tiempo y el tamaño n de la instancia.

- Por ejemplo, si un algoritmo que ordena una lista de n elementos tiene una complejidad del orden de $\mathcal{O}(n^2)$, significa que existe un número $k \in \mathbb{R}$ tal que el algoritmo requiera a lo más kn^2 unidades de tiempo para obtener una solución.

Complejidad de un Algoritmo IV

Algunas propiedades de la Notación $\mathcal{O}$

1. $f(n) = \mathcal{O}(g(n)) \wedge g(n) = \mathcal{O}(h(n)) \Rightarrow f(n) = \mathcal{O}(h(n))$
2. $f(n) = \mathcal{O}(g(n)) \Rightarrow f(n) + g(n) = \mathcal{O}(g(n))$
3. $n^p = \mathcal{O}(q^n)\ \forall p > 0, q > 1$

- La propiedad 2 nos permite concluir que: $\mathcal{O}(an^2 + bn + c) = \mathcal{O}(an^2 + bn) = \mathcal{O}(an^2) = \mathcal{O}(n^2)$.
- La propiedad 3 nos dice que todo polinomio crece asintóticamente más lentamente que cualquier expresión exponencial.
- La propiedad 2 y 3 nos permite concluir que: $\mathcal{O}(2^n + n^k) = \mathcal{O}(2^n)$.

Complejidad de un Algoritmo V

- En general, se utiliza la menor cota superior asintótica. Por ejemplo, es cierto que $2n = \mathcal{O}(n^2)$ pero es preferible decir que $2n = \mathcal{O}(n)$.

Si un algoritmo tiene una complejidad $f(n) = \mathcal{O}(g(n))$ se dice:

- **Algoritmo Polinomial:** si $g(n)$ es un polinomio en n. Estos algoritmos son considerados eficientes.
- **Algoritmo Exponencial:** si $f(n)$ no puede ser acotado por un polinomio. Estos algoritmos se consideran ineficientes.

Complejidad de un Algoritmo VI

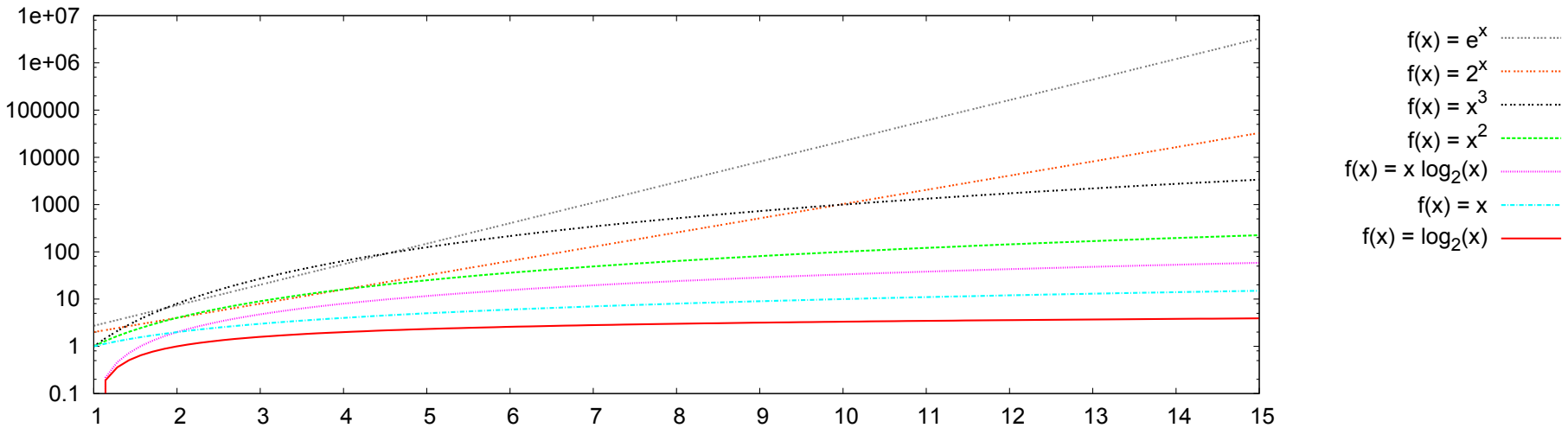


Figura: Crecimiento de algunas funciones

Complejidad de un Algoritmo VII

$g(n) \setminus n$	10	30	50	60
n	0,00001 [seg]	0,00003 [seg]	0,00005 [seg]	0,00006 [seg]
n^2	0,001 [seg]	0,027 [seg]	0,125 [seg]	0,216 [seg]
2^n	0,001 [seg]	17,9 [min]	35,7 [años]	366 [siglos]
3^n	0,059 [seg]	6,5 [años]	2×10^8 [siglos]	$1,3 \times 10^{13}$ [siglos]

Cuadro: Tiempos de ejecución para distintos $g(n)$ y n

Complejidad de un Problema I

La **complejidad de un problema** está dada por la complejidad del **mejor algoritmo** de entre todos los algoritmos que resuelven el problema.

- Al analizar la complejidad de un algoritmo particular para un problema se obtiene sólo una cota superior para la complejidad del problema.
- Se puede determinar una cota inferior de la complejidad de un problema probando matemáticamente que cualquier posible algoritmo deberá tener, por lo menos, la complejidad establecida por la cota inferior.

Complejidad de un Problema II

- Por lo tanto, la complejidad de un problema se establece sólo cuando ambas cotas coinciden.
- Actualmente existe una gran cantidad de problemas para los cuales las cotas son distintas, es decir, se desconoce su complejidad a pesar que existen algoritmos que los resuelven.

Complejidad de un Problema III

Existen distintas clases de problemas:

Clase $\mathcal{P}$

Problemas en los cuales existe un algoritmo polinomial que los resuelve (problemas fáciles).

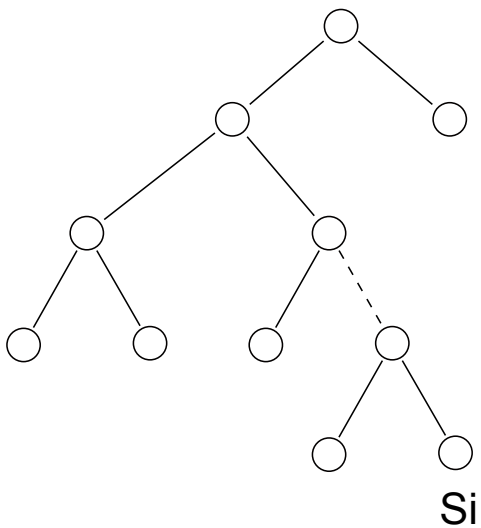
Clase $\mathcal{NP}$ (non-deterministic polinomial)

Problemas en los cuales existe un algoritmo no determinístico que lo resuelve en tiempo polinomial.

La clase $\mathcal{NP}$ se define sobre una **Máquina de Turing No Determinística**, la que se caracteriza por ser capaz de ejecutar muchas instrucciones simples en forma simultánea.

Complejidad de un Problema IV

- Supongamos una máquina que tiene la propiedad de dividirse en varias copias idénticas de sí misma cada vez que se ve enfrentada a una alternativa, y que las copias continúan trabajando en paralelo. Representemos este proceso como un árbol de búsqueda:



Complejidad de un Problema V

- Si una de las copias responde afirmativamente, está resuelto el **problema de decisión**.
- Si el tiempo máximo que se requiere para recorrer una rama está acotado polinomialmente, el problema está en $\mathcal{NP}$.
- Si alguien me da una solución para una instancia de mi problema cuya respuesta es SI y yo puedo verificar en tiempo polinomial que esa solución es correcta, el problema está en $\mathcal{NP}$.
  - Estos problemas tienen un *certificado* de la respuesta SI, que puede ser verificado en tiempo polinomial.
  - Un algoritmo que en la máquina no determinística requiere tiempo polinomial, se puede ejecutar en la máquina determinística en tiempo exponencial.

Complejidad de un Problema VI

- Lo anterior indica que los problemas $\mathcal{NP}$ son más difíciles que los de la clase $\mathcal{P}$. Por lo tanto, la clase $\mathcal{NP}$ contiene los problemas de la clase $\mathcal{P}$: $\mathcal{P} \subseteq \mathcal{NP}$.
- Por ejemplo: El problema del vendedor viajero está en $\mathcal{NP}$.

- Entre los problemas $\mathcal{NP}$ existe una subclase formada por los problemas difíciles aún no resueltos eficientemente, los problemas $\mathcal{NP}$-completos.

Clase $\mathcal{NP}$-completo

Un problema A de la clase $\mathcal{NP}$ es $\mathcal{NP}$-**completo** si cualquier problema en $\mathcal{NP}$ se puede transformar polinomialmente en A.

Complejidad de un Problema VII

- Esto significa que si encuentro un algoritmo polinomial para un problema $\mathcal{NP}$-completo, automáticamente se pueden resolver polinomialmente todos los problemas de la clase $\mathcal{NP}$.
- Una pregunta abierta en complejidad es:
  ¿$\mathcal{P} = \mathcal{NP}$ ó $\mathcal{P} \neq \mathcal{NP}$?

Complejidad de un Problema VIII

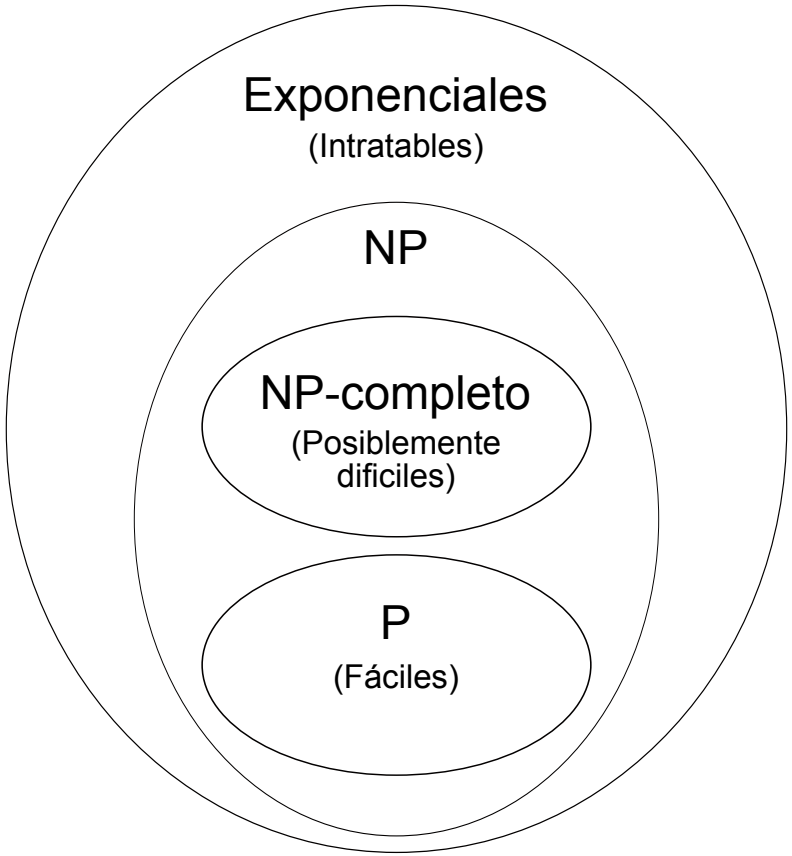


Figura: Clases de complejidad computacional

Complejidad de un Problema IX

Ejemplos:

1. Problemas $\mathcal{P}$:
   - Ordenar un arreglo.
   - Ruta más corta entre 2 puntos.
   - Calcular el determinante de una matriz.
   - Programación Lineal.
2. Problemas $\mathcal{NP}$-completo:
   - Problema del Vendedor Viajero.
   - Programación Lineal Entera.
3. Problemas Intratables:
   - Determinar todos los puntos enteros que satisfacen un sistema de desigualdades lineales.
   - Las Torres de Hanoi.

Complejidad de un Problema X

- Muchos problemas de programación matemática que se originan de aplicaciones reales resultan estar en la clase $\mathcal{NP}$ y por lo tanto son potencialmente difíciles de resolver. Esto ha dado lugar a mucha investigación en métodos heurísticos para estos problemas.
- En la práctica muchos algoritmos con mala complejidad teórica (por el peor caso) tienen un comportamiento promedio aceptable, por ello, actualmente existen otros enfoques que analizan complejidad promedio (usando algún modelo estocástico apropiado).

Contenidos

Problemas y Procedimientos de solución
- Problemas de optimización v/s problemas de decisión
- Métodos, Algoritmos y Heurísticas

Complejidad de Algoritmos y Problemas
- Consideraciones
- Máquina de Turing
- Complejidad de un Algoritmo
- Complejidad de un Problema

Referencias

Referencias I


Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001).
Introduction to Algorithms.
The MIT Press, 2 edition.


DII.
Apuntes in34a - optimización.


Varas, S., Ortiz, C., and Vera, J. (2000).
Optimización y modelos para la gestión.
Dolmen Ediciones.