Diseño y Análisis de Algoritmos
Complejidad

Dr. Jesús Ariel Carrasco Ochoa
ariel@inaoep.mx
Oficina 8311

Contenido

- Definición
- Clases de complejidad

Complejidad Computacional

Estudio de los recursos necesarios (tiempo y espacio) para resolver un problema.

Análisis de complejidad de un algoritmo es la estimación de los recursos requeridos por el algoritmo (tiempo y espacio) para resolver un problema.

Clases de Complejidad

Los problemas se dividen en clases de complejidad dependiendo del tiempo y espacio necesarios para resolverlos.

Hay clases de tiempo y clases de espacio.

Problemas en una misma clase requieren un tiempo o espacio acotado por la clase.

Clases de Complejidad

Las clases de complejidad más conocidas son P y NP.

P es la clase de problemas que pueden ser resueltos en tiempo polinomial, es decir existe un algoritmo determinista de complejidad polinomial que los resuelve.

Clases de Complejidad

NP es la clase de problemas para los cuales se puede decidir si una solución es correcta o no en tiempo polinomial, es decir existe un algoritmo no-determinista de complejidad polinomial que los resuelve.

1) Seleccionar una posible solución *X*
2) Decidir si *X* es una solución correcta

   Sí → Fin, la solución es *X*

   No → ir al paso 1

Clases de Complejidad

Teorema: $P \subseteq NP$

Demostración:

Sea $P_1 \in P$ un problema cualquiera en P, entonces existe un algoritmo polinomial A_1 que lo resuelve.

Consideremos el siguiente algoritmo no determinista:

Clases de Complejidad

1) Seleccionar una posible solución X
2) Aplicar A_1 para resolver P_1 obteniendo X_1
3) Decidir si $X=X_1$

   Sí → Fin, la solución es X

   No → ir al paso 1

Este algoritmo es un algoritmo no determinista polinomial que resuelve a P_1 por lo tanto $P_1 \in NP$, con lo cual podemos concluir que $P \subseteq NP$ ■

Clases de Complejidad

Conjetura: $P \neq NP$

Clases de Complejidad

TiempoD($f(n)$)

Clase de problemas para los cuales existe un algoritmo determinista con tiempo acotado por $f(n)$.

EspacioD($f(n)$)

Clase de problemas para los cuales existe un algoritmo determinista con espacio acotado por $f(n)$.

Clases de Complejidad

TiempoN($f(n)$)

Clase de problemas para los cuales existe un algoritmo no determinista con tiempo acotado por $f(n)$.

EspacioN($f(n)$)

Clase de problemas para los cuales existe un algoritmo no determinista con espacio acotado por $f(n)$.

Propiedades de las Clases de Complejidad

Si $P_1 \in \text{TiempoD}(f(n)) \rightarrow P_1 \in \text{EspacioD}(f(n))$

Si $P_1 \in \text{EspacioD}(f(n))$ y $f(n) \geq log(n)$ entonces $\exists c \in \Re$ (que depende de P_1) tal que $P_1 \in \text{TiempoD}(c^{f(n)})$

Propiedades de las Clases de Complejidad

Para cualquier función totalmente recursiva $f(n)$ entonces $\exists$ P_1 tal que $P_1 \in \text{TiempoD}(f(n))$

Para cualquier función totalmente recursiva $f(n)$ entonces $\exists$ P_1 tal que $P_1 \in \text{EspacioD}(f(n))$

Propiedades de las Clases de Complejidad

Si $\lim_{n\to\infty}\left(\frac{g(n)}{f(n)}\right) = 0$ entonces:

$\text{TiempoD}(g(n)) \subset \text{TiempoD}(f(n))$

$\text{EspacioD}(g(n)) \subset \text{EspacioD}(f(n))$

$\forall(\varepsilon>0, r>0)$:

$\text{TiempoD}(n^r) \subset \text{TiempoD}(n^{r+\varepsilon})$

$\text{EspacioD}(n^r) \subset \text{EspacioD}(n^{r+\varepsilon})$

Definición de P y NP

$$\mathrm{P} = \mathrm{U}_{i \geq 1} \, \mathrm{TiempoD}(n^i)$$

$$\mathrm{NP} = \mathrm{U}_{i \geq 1} \, \mathrm{TiempoN}(n^i)$$

Problemas NP-difíciles

Un problema X es NP-difícil si para todo $Y \in NP$:

1) Cualquier instancia Y1 de Y puede ser transformada en una instancia X1 de X en tiempo polinomial
2) Si se resuelve X1 obteniendo la solución SX1, SX1 puede transformarse en una solución SY1 de Y1 en tiempo polinomial.

Problemas NP-difíciles

Si un problema X es NP-difícil:

1) X es tanto o más difícil que cualquier problema en NP
2) Si X puede ser resuelto en tiempo polinomial entonces cualquier problema en NP puede ser resuelto en tiempo polinomial, es decir P=NP.

Problemas NP-completos

Un problema X es NP-completo si:

1) X es NP-difícil
2) $X \in NP$.

Problemas NP-completos

Si un problema X es NP-completo:

1) X es tanto o más difícil que cualquier otro problema en NP
2) Si X puede ser resuelto en tiempo polinomial entonces cualquier otro problema en NP puede ser resuelto en tiempo polinomial, es decir P=NP.

Problemas NP-completos

Existen los problemas N-Completos ?

Problemas NP-completos

Existen los problemas N-Completos ?

Cómo decidir si un problema X es NP-completo ?

- Demostrando que X cumple la definición de NP-completo
- Demostrando que $X \in NP$ y que un problema Y que se sepa que es NP-completo puede transformarse en X en tiempo polinomial y que la solución de X se puede transformar en la de Y en tiempo polinomial

Problemas NP-completos

Existen los problemas N-Completos ?

SAT es un problema NP-completo

Dada una fórmula Booleana de *n* variables decidir si existe una combinación de valores para sus variables que la satisfaga (evalúe a 1)

Problemas NP-completos

SAT es un problema NP-completo

Dada una combinación de valores para las variables de una fórmula booleana se puede verificar en tiempo polinomial si se satisface o no.

- A partir de esto se puede construir un algoritmo no determinístico polinomial para SAT

Problemas NP-completos

SAT es un problema NP-completo

- Si un problema X esta en NP existe un algoritmo AX que puede verificar si una solución candidata es correcta en tiempo polinomial.
- La solución candidata puede codificarse en binario y entonces el algoritmo AX puede escribirse como una fórmula Booleana que solo se satisface si la solución candidata es correcta.

Problemas NP-completos

CNFSAT es un problema NP-completo

Dada una fórmula Booleana en Forma Normal Conjuntiva (CNF) de *n* variables decidir si existe una combinación de valores para sus variables que la satisfaga.

- Forma Normal Conjuntiva (CNF) significa que es un conjunción de disyunciones de variables independientes

CNFSAT está en NP porque SAT está en NP y una fórmula Booleana en CNF sigue siendo una fórmula Booleana.

Problemas NP-completos

CNFSAT es un problema NP-completo

Se sabe que cualquier fórmula Booleana se puede transformar en otra equivalente que esté en CNF.

1. Aplicar las leyes de D'Morgan hasta que todas las negaciones apliquen sólo a variable individuales
2. Aplicar distributividad hasta tener una CNF

Problemas NP-completos

CNFSAT es un problema NP-completo

Esta transformación se puede hacer en tiempo polinomial

La solución de CNFSAT no tiene que transformarse pues ya es una solución de SAT

Por lo tanto CNFSAT es NP-completo

Problemas NP-completos

3-CNFSAT es un problema NP-completo

Dada una fórmula Booleana en Forma Normal Conjuntiva (CNF) de *n* variables con exactamente 3 variables en cada disyunción, decidir si existe una combinación de valores para sus variables que la satisfaga.

3-CNFSAT está en NP porque SAT está en NP y una fórmula Booleana en 3-CNF sigue siendo una fórmula Booleana.

Problemas NP-completos

3-CNFSAT es un problema NP-completo

Una fórmula Booleana en CNF se puede transformar en otra equivalente que esté en 3-CNF.

1. Seleccionar una disyunción que no tenga exactamente 3 variables $(x_{i_1} \vee x_{i_2} \ldots \vee x_{i_k})$ y sustituirlo por:

   $(x_{i_1} \vee x_{i_2} \vee z_1)(\overline{z_1} \vee x_{i_3} \vee z_2)\ldots(\overline{z_{K-3}} \vee x_{i_{K-1}} \vee z_k)$

2. Repetir hasta que todas las disyunciones tengan exactamente 3 variables

Problemas NP-completos

3-CNFSAT es un problema NP-completo

Esta transformación se puede hacer en tiempo polinomial

La solución de 3-CNFSAT se transforma en una de CNFSAT eliminado todas las variables agregadas y considerando solo las originales

Por lo tanto 3-CNFSAT es NP-completo

Problemas NP-completos

2-CNFSAT es un problema NP-completo ?

Problemas NP-completos

2-CNFSAT es un problema NP-completo ?

Se sabe que 2-CNFSAT está en P pues existe al menos un algoritmo determinista polinomial que lo resuelve.

EspacioP y EspacioNP

$$\text{EspacioP} = \bigcup_{i \geq 1} \text{EspacioD}(n^i)$$

$$\text{EspacioNP} = \bigcup_{i \geq 1} \text{EspacioN}(n^i)$$

Sobre P y NP

$$EspacioD(\log(n)) \subseteq P \subseteq NP \subseteq EspacioP$$

Al menos una contención es propia dado que:

$$EspacioD(\log(n)) \subset EspacioP$$

Diseño y Análisis de Algoritmos

Sumas

Dr. Jesús Ariel Carrasco Ochoa
ariel@inaoep.mx
Oficina 8311