DEPT. INFORMÁTICA E INGENIERÍA DE SISTEMAS
UNIVERSIDAD DE ZARAGOZA

NP-completos

Modelos Abstractos de Cálculo

Elvira Mayordomo

Una aplicación práctica

- Supón que tu jefe te pide que escribas un algoritmo eficiente para un problema extremadamente importante para tu empresa.
- Después de horas de romperte la cabeza sólo se te ocurre un algoritmo de “fuerza bruta”, que analizándolo ves que cuesta tiempo exponencial.
- Te encuentras en una situación muy embarazosa:

“No puedo encontrar un algoritmo eficiente, me temo que no estoy a la altura”

Una aplicación práctica

- Te gustaría poder decirle al jefe:

"No puedo encontrar un algoritmo eficiente, ¡porque no existe!"

- Para la mayoría de los problemas, es **muy difícil demostrar que son intratables**, porque la mayoría de los problemas interesantes que no se saben resolver son **NP-completos**.
  - Los NP-completos parecen intratables
  - Nadie ha sabido demostrar que los NP-completos son intratables

Una aplicación práctica

- Después de aprobar M.A.C. puedes ser capaz de demostrar que el problema de tu jefe es NP-completo y decirle:

"No puedo encontrar un algoritmo eficiente, pero tampoco pueden un montón de científicos famosos"

- O todavía mejor:

"Si pudiera diseñar un algoritmo eficiente para este problema, ¡no estaría trabajando para usted! Me habría ganado el premio de un millón de dólares que da el Instituto Clay."

Una aplicación práctica

- Después de la segunda respuesta, tu jefe abandonará la búsqueda.
- Pero la necesidad de una solución no desaparecerá.
- Seguramente al demostrar que es NP-completo has aprendido mucho y ahora puedes:
  - Olvidar lo de intentar encontrar un algoritmo en tiempo polinómico para el problema.
  - Buscar un algoritmo eficiente para un problema diferente relacionado con el original.
  - O bien intentar usar el algoritmo exponencial a ver qué tal funciona con las entradas que te interesan.

P versus NP

- Ya hemos visto las clases de problemas P y NP:
  - En P están los problemas que se pueden resolver en tiempo polinómico.
  - En NP están los problemas que se pueden **comprobar** en tiempo polinómico.

P versus NP

- Sabemos que P $\subseteq$ NP.
- ¿P=NP? Esta es una de las preguntas abiertas más importantes en informática.
- Ver en

  http://www.claymath.org/prizeproblems/index.htm

  cómo ganar **un millón de dólares** resolviéndola.

P versus NP

- ¿Cómo la resolverías?
- Para intentar P $\neq$ NP:

  Demostrar que hay un problema que está en NP pero no en P.
- Para intentar P=NP:

  Demostrar que todos los problemas de NP se pueden resolver en tiempo polinómico, así que NP $\subseteq$ P

P versus NP

Con los NP-completos, esto se simplifica.

- Para intentar P=NP:

  Demostrar que hay un problema NP-completo que se puede resolver en tiempo polinómico.

P versus NP

- P ≠ NP es **equivalente** a “existe un problema NP-completo que no está en P”

- P = NP es **equivalente** a “existe un problema NP-completo que está en P”

Lista de NP-completos

- Existe una larga lista de NP-completos (ver Garey Johnson).
- Añadir un problema a la vista quiere decir que el problema es tan difícil como cualquiera de los que ya están (puedes decirle al jefe *"No puedo encontrar un algoritmo eficiente, pero tampoco pueden un montón de científicos famosos"*)

Lista de NP-completos

- Para añadir un problema C a la lista hay que:
  1) Demostrar que C está en NP
  2) Hacer una reducción $B \leq^{p} C$ desde un problema B de la lista.

Referencias

- GAREY, M. y JOHNSON. D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman. 1978.
- Premio del instituto Clay:

  www.claymath.org/prizeproblems/pvsnp.htm

  www.claymath.org/prizeproblems/milliondollarminesweeper.htm