Diseño de Algoritmos
Sesión 08

Profesores:
Tomás Lara Valdovinos – t.lara@uandresbello.edu
Jessica Meza-Jaque – je.meza@uandresbello.edu

OBJETIVOS DE LA SESIÓN

- Conocer los conceptos de clases de problema P y NP
- Relacionar estas clases en cuanto a sus características
- Analizar ejemplos de problemas clasificados en estas clases

CONTENIDOS DE LA SESIÓN

- Concepto de clase P
- Concepto de clase NP
- Muestra de ejemplos

Recordemos

- **Problemas de Búsqueda**
- **Problemas de Decisión**

Problemas de Búsqueda

- Dada una instancia encontrar una solución correspondiente.
- También se puede determinar que no existe solución o ésta no está definida (devolver λ o $\perp$).
- Corresponde a la noción de “Resolver un problema” en la vida real.

Problemas de Decisión

- Dada una instancia, determinar si la instancia se encuentra en un grupo especificado (la instancia cumple una condición).
- La solución puede ser ‘si’ o ‘no’

P versus NP

- Cotidianamente nuestra experiencia nos muestra que **resolver problemas** por lo general es más difícil que determinar si **su solución es correcta**.
- ¿Es esto coincidencia o es la representación de un hecho real en la vida?
- De esta manera podemos ver informalmente es sentido de la pregunta P vs NP

Clase de problemas P

- La clase de **problemas P** representa los **Problemas de decisión** que pueden ser **Eficientemente** resueltos.
- Definimos como **Eficiente** como una solución que puede ser expresada como un polinomio en función del **Tamaño de su entrada** (importante relación para la exposición presentada desde ahora).

- **P** es por **Polynomial-time**

Clase de problemas P - Ejemplos

- Ordenar un conjunto de números
  - Solución O(n*logn) usando el algoritmo de ordenamiento Quicksort

- Encontrar un número en un arreglo
  - Solución O(n) usando búsqueda secuencial
  - Solución O(logn) usando búsqueda binaria

- Encontrar la ruta más corta entre 2 nodos de un grafo
  - Solución O(V + E) Usando el algoritmo Depth-First Search (Búsqueda en profundidad)

Clase de Problemas NP

- La clase de **Problemas NP** representa los problemas que pueden ser resueltos en tiempo polinómico en una **máquina de Turing (MT) no-determinista**.
- También podemos definirlo como el grupo de problemas de decisión cuya **solución puede ser verificada** en tiempo polinómico.

- NP es por **Non-deterministic Polynomial-time**

Clase de Problemas NP

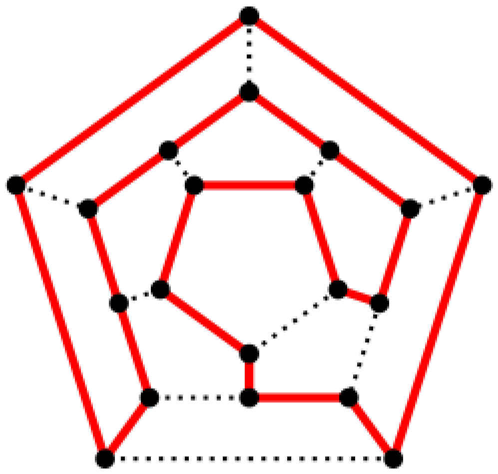
- Una MT no determinista es una máquina que utilizando una misma entrada podría entregar **muchos posibles** resultados sin poder determinarse el “camino” para lograrlo.
- Podemos llamarlo también **Algoritmo No determinista**

Clase de Problemas NP - Ejemplos

- Todos los problemas en P pertenecen a NP
  - Una MT no-determinista puede funcionar como una MT determinista.
  - Podemos determinar que cualquier solución de un problema P es válida.

Clase de problemas NP - ejemplo

- Determinar el ciclo hamiltoniano en un grafo, esto implica encontrar una ruta que pase a través de todos los nodos del grafo y vuelva al inicio.
    - Se puede resolver con heurísticas.
    - La solución es fácilmente verificable O(n) si podemos identificar que todos los nodos están marcados en la ruta elegida.



Intratabilidad en la computación

- Decimos que un problema es intratable cuando:
  - Solucionarlo requiere más que un tiempo polinomial en función del tamaño de su entrada.
  - Problemas que no tienen una solución (asumiendo hasta el día de hoy que existen).

- Ejemplos:
  - Problemas cuya única solución encontrada hasta ahora es de tiempo exponencial (solo pueden ser resueltos por fuerza bruta).
  - Halting Problem (No tiene solución).

Formalicemos la definición de P y NP

- Podemos definir las clases de problemas P y NP en términos de **problemas de Búsqueda** y **Problemas de decisión**.

En función de “Problemas de búsqueda”

La clase P como un problema de búsqueda natural

- Decimos que P es la agrupación de problemas de búsqueda cuya solución pude ser encontrada (cuando existe) eficientemente.

- R $\subseteq$ {0, 1}* x {0, 1}* se puede resolver eficientemente si:
  - Existe un algoritmo en tiempo polinómico **A** dado que para cada entrada **x** tenemos que:
    - Si $R(x) = \{y : (x, y) \in R\}$ no es vacío, entonces A(x) resuelve el problema de búsqueda R
    - Y de otra forma, $A(x) = \perp$ (indicando que x no tiene solución)

La clase P como un problema de búsqueda natural

- Detonaremos por *PF (Polynomial-time Find)* la clase de problemas de búsqueda que son eficientemente resolubles.
- Esto es:
  - R ∈ PF si R tiene una solución que podemos encontrar eficientemente.
  - Y existe un algoritmo que en tiempo polinómico resuelve el problema de R.

La clase P como un problema de búsqueda natural

- Otras consideraciones:
  - Si para la instancia x, no existe una solución (por ejemplo $R(x) = \emptyset$) entonces claramente el algoritmo **A** no resuelve R (*$A(x) \notin R(x)$*).
  - En este caso se requiere que $A(x) = \perp$.
  - Por lo tanto A **SIEMPRE** entrega una respuesta correcta, en este caso indicando que x no tiene una solución.

En función de “Problemas de búsqueda”

La clase NP como un problema de búsqueda natural

- Los problemas de búsqueda naturales tienen la propiedad que las soluciones válidas (para ellos) pueden ser **eficientemente** reconocibles.
- Esto es:
  - Dada una instancia **x** para el problema **R** y una solución candidata **y**, se puede identificar cuando **y** es una solución válida o no para **x** con respecto al problema **R.**
  - $\mathbf{y} \in \mathbf{R(x)}$ o $\mathbf{y} \notin \mathbf{R(x)}$

La clase NP como un problema de búsqueda natural

- El problema de búsqueda **R**, dado que:
  - R $\subseteq$ {0, 1}* x {0, 1}*
- Tiene una solución verificable si existe un algoritmo **A**, dado que:
  - Para cada **x** e **y, A(x, y) = 1** si y solo si (x, y) ∈ R.
  - A(x, y) = 0 de lo contrario.
  - En palabras simples: A comprueba si y es una solución válida para x en R

La clase NP como un problema de búsqueda natural

- Denotaremos por *PC* (Polynomial-time Check) a la clase de problemas de búsqueda cuya solución puede ser **verificable** eficientemente.

- La clase *PC* es el dominio para el estudio de cuales problemas están en *PF* ya que la habilidad para reconocer una solución valida es un prerrequisito para discutir acerca de la complejidad de encontrar una solución.

P vs NP en términos de Problemas de búsqueda

- ¿Qué pasaría si cada problea en PC estuviera también en PF?

- Si PC $\subseteq$ PF se consideraría que cuando la solución de las instancias pueden ser **eficientemente verificables** (que sean correctas), también es el caso de que esas soluciones pueden ser **encontradas eficientemente**.

- En palabras simples, todos los problemas razonables en PC son fáciles de resolver.

P vs NP en términos de Problemas de búsqueda

- Por otra parte, si PC \ PF no es vacío, entonces existen problemas razonables que son difíciles de resolver.
- Esto conforma la intuición básica de que algunos problemas son fáciles de resolver mientras que otros son difíciles de resolver.

P vs NP en términos de Problemas de búsqueda

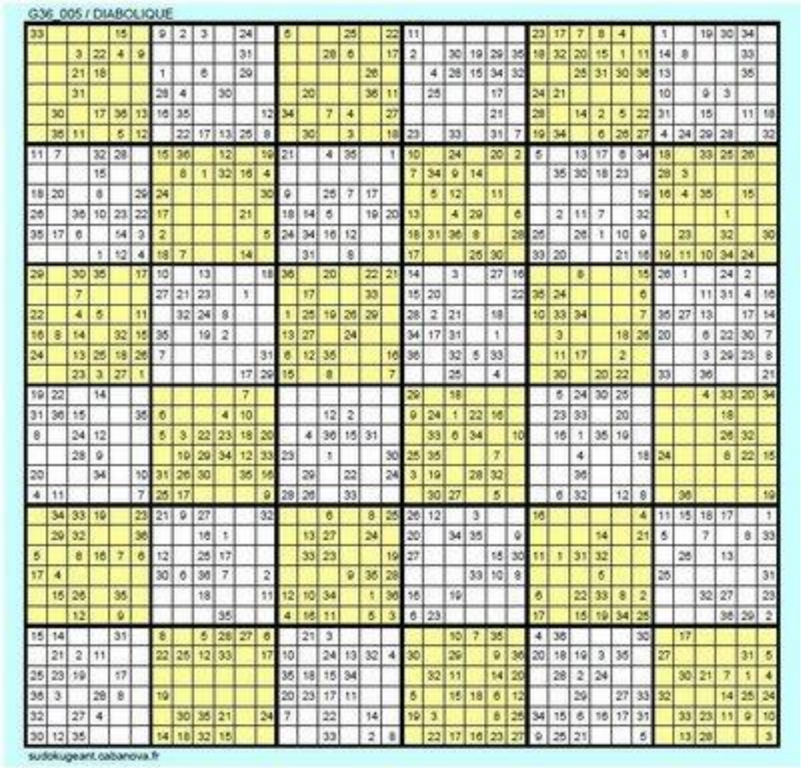
- Ejemplo:

- Considere varios juegos de puzzle, como rompecabezas, laberintos, sudoku, etc.

Rompecabezas 4 piezas

Rompecabezas 1000 piezas

Sudoku 6x6



Sudoku 3x3

P vs NP en términos de Problemas de búsqueda

- En cada uno de esos puzzles, verificar que la solución es correcta es muy fácil, mientras que encontrar una solución a veces es extremadamente difícil.

En función de “Problemas de decisión”

La clase P como un problema de decisión natural

- Desde este punto de vista, la clase P está compuesta por todos los problemas de decisión que pueden ser resueltos de manera eficiente.

- Un problema de decisión $S \subseteq \{0, 1\}^*$ puede ser resuelto eficientemente si:
  - Existe un algoritmo **A** computable en tiempo polinomial, dado que:
    - Para cada x, A(x) = **1** si y solo si **x ∈ S**
- Denotaremos como **P** a este tipo de problemas

En función de “Problemas de decisión”

La clase NP como un problema de decisión natural

La definición más popular indica que:

- Un problema de decisión está en NP si una Máquina de Turing no-determinista entrega una solución que defina la membresía de una instancia en un grupo.

- Pero podemos ver esta definición de una manera menos ficticia, sin cambiar el significado de la siguiente manera…

La Clase NP y los sistemas de prueba NP

- Decimos que, la membresía de una instancia en un set puede ser determinada a través de una prueba adecuada.
- Entonces, definimos NP como la clase de problemas de cisión que tiene **sistemas de pruebas eficientemente verificables.**

- En términos generales, decimos que un set **S** tiene un sistema de pruebas si las instancias en **S** tienen pruebas válidas de membresía, mientras que las instancias que no están en **S** no tienen pruebas válidas.

P vs NP

- Si pudiéramos demostrar que todos los problemas en NP pueden ser resueltos de manera eficiente, esto es:
  - NP $\subseteq$ P
- Y por lo tanto la **correctitud** de su solución es verificable en tiempo polinómico para cada problema.
- Entonces podemos decir que P = NP

P = NP

- Actualmente no ha sido posible verificar esta igualdad.
- Actualmente se está suponiendo que P ≠ NP
- En sesiones posteriores veremos de qué manera se podría comprobar esta igualdad, y cual es su contexto histórico.

CHECK - OBJETIVOS DE LA SESIÓN

- Conocer los conceptos de clases de problema P y NP
- Relacionar estas clases en cuanto a sus características
- Analizar ejemplos de problemas clasificados en estas clases

CHECK

Diseño de Algoritmos
Sesión 08

Profesores:
Tomás Lara Valdovinos – t.lara@uandresbello.edu
Jessica Meza-Jaque – je.meza@uandresbello.edu