INF155 - Informática Teórica

Ayudantía #9

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

En el capítulo anterior

(a la misma hora y en este mismo canal)

Lenguajes computables y CE

- Un lenguaje es *computable* si existe una TM que siempre se detiene a decir sí o no.
- Un lenguaje es *computacionalmente enumerable* (CE) si existe alguna TM que lo acepte.
- Un lenguaje es no CE si no existe ninguna MT que lo acepte.

Propiedades

- $\mathscr{L}$ es computable $\Rightarrow$ $\overline{\mathscr{L}}$ es computable
- $\mathscr{L}$ y $\overline{\mathscr{L}}$ son CE $\Rightarrow$ ambos son computables
- $\mathscr{L}$ es CE no computable $\Rightarrow$ $\overline{\mathscr{L}}$ no es CE
- $\mathscr{L}$ no es CE $\Rightarrow$ $\overline{\mathscr{L}}$ no es computable

Algunos lenguajes icónicos

- $\mathscr{L}_d$ el lenguaje diagonal es no CE.
- $\mathscr{L}_U$ el lenguaje universal es CE no computable.
- $\mathscr{L}_{ne}$ es CE no computable.
- $\mathscr{L}_e$ es no CE.

Teorema de Rice

Toda propiedad no trivial de los lenguajes CE es no decidible, no computable.

Una **propiedad** de los lenguajes CE es simplemente un conjunto de los lenguajes CE. Si la propiedad es **no trivial** significa que hay lenguajes que tienen dicha propiedad y otros que no.

Recursos

Previamente, nos enfocamos sólo en aquellos problemas que podemos resolver computacionalmente. Ahora, además de considerar los problemas en sí, nos interesaremos en lo que podemos hacer con los recursos que tenemos a disposición (los cuales son **escasos**).
Particularmente, nos centraremos en el recurso **tiempo**.

Propiedades de clausura de los polinomios

Los polinomios son cerrados respecto a la composición, es decir que si $p(n)$ y $q(n)$ son polinomios, entonces $p \circ q(n) = p(q(n))$ también es un polinomio.

Complejidad

Se habla de la *complejidad* de un algoritmo como la función que acota el uso de recursos en función del tamaño del problema. Decimos que un problema tiene *solución eficiente* si el tiempo está acotado por un polinomio.
Para una TM no determinista, consideramos el tiempo de ejecución como el número de pasos de la computación más corta (el camino más corto, y siempre elige el camino más corto) que lleva a aceptar.


P=NP
e^πi = -1

P

Un problema $\mathscr{L}$ está en P (TM determinista, tiempo polinomial) si hay una TM determinista que acepta el lenguaje $\mathscr{L}$, determinando si $\sigma \in \mathscr{L}$ en un número de pasos acotado por un polinomio en $|\sigma|$.

En otras palabras, se dice que un problema está en P si hay un algoritmo determinista que lo resuelve en tiempo polinomial.

Ejemplos: multiplicación, algoritmos de ordenamiento, máximo común divisor …

NP

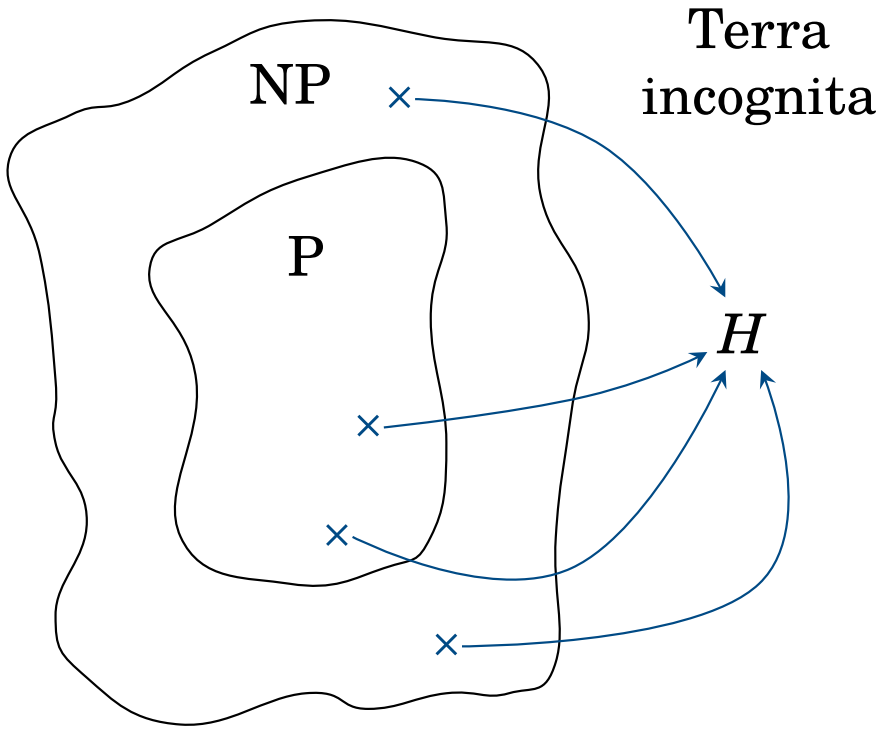
Un problema $\mathscr{L}$ está en NP (TM no determinista, tiempo polinomial) si hay una TM no determinista que determina si $\sigma \in \mathscr{L}$ en un número de pasos acotado por un polinomio en $|\sigma|$. Si no hay una computación que acepta en esos pasos, sabemos que no acepta. Son problemas computables.

Una definición más práctica es la siguiente: Un problema $\mathscr{L}$ está en NP si y solo si para cada instancia $\sigma \in \mathscr{L}$ hay un *certificado* c tal que una TM determinista con entrada σ y c verifica que $\sigma \in \mathscr{P}$ en tiempo polinomial en $|\sigma|$.

NP-Hard

Un problema se llama NP-duro (NP-*hard*) si todos los problemas en NP se reducen polinomialmente a él (como H en la figura)

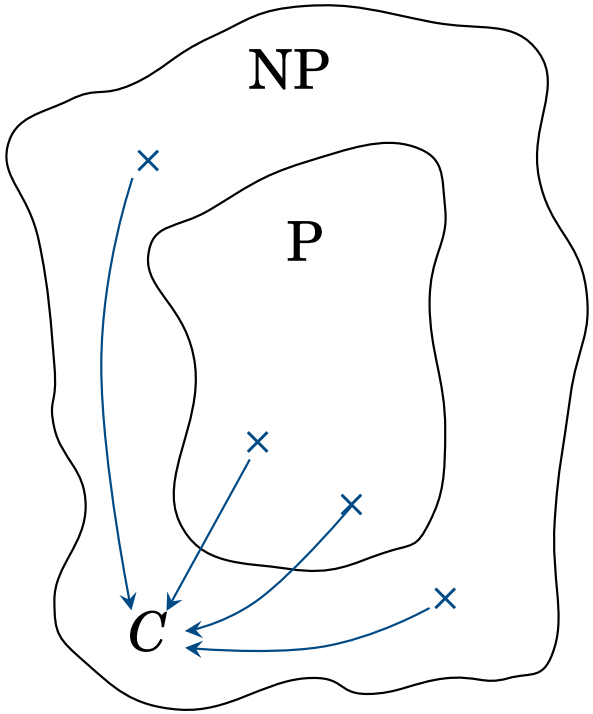
Ejemplo: Halting Problem es NP-Hard, pero $\notin NP$



NP-Completo

Un problema es NP-completo si es NP-duro y está en NP. Un ejemplo es el problema C en la figura.

Notar que si se halla un problema NP-Completo C con solución determinista polinomial, automáticamente $P = NP$.



Ejemplos de Problemas NP-Completos

- Timetabling Problem
- Travelling Salesman Problem
- Knapsack Problem
- Minesweeper Consistency Problem
- Vehicle Routing Problem
- etc.

Reducción en tiempo polinomial

Se dice que P *se reduce polinomialmente a* Q (se anota $P \leq_p Q$) si hay un algoritmo determinista que traduce toda instancia σ de P en una instancia de Q con la misma respuesta en tiempo polinomial.

Además, si $A \leq_p B$ y $B \in$ NP, entonces $A \in$ NP. De la misma forma, si $A \leq_p B$, y $B \in$ P, entonces $A \in$ P.

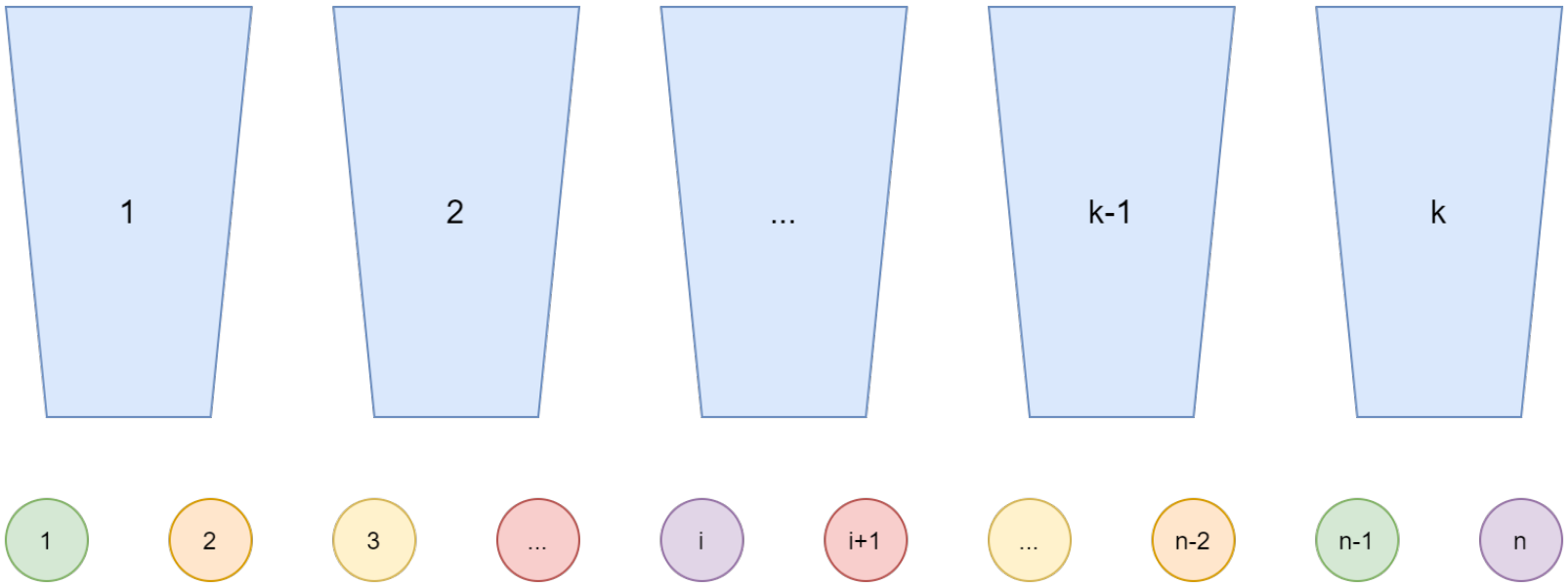
¿Cómo demostrar que un problema es NP-Completo?

Para demostrar que un problema es NP-Completo, debemos:

- **Demostrar que está en NP:** Por la definición alternativa de NP, una manera es describir un certificado con el cual se pueda demostrar en tiempo polinomial que la instancia cumple.
- **Demostrar que es NP-duro:** Esto es, demostrar que todos los problemas en NP pueden reducirse polinomialmente a él. Pero como todos los problemas en NP se pueden reducir a un problema que es NP-completo, y este está en NP, basta reducir polinomialmente un problema NP-completo cualquiera a nuestro problema.

Ejemplo

El problema BinPacking considera un conjunto de n ítems, donde el ítem i tiene un peso w_i asociado. Se cuenta, además, con contenedores (*bin*) idénticos, de capacidad V. La pregunta es si los ítems pueden almacenarse en k contenedores.



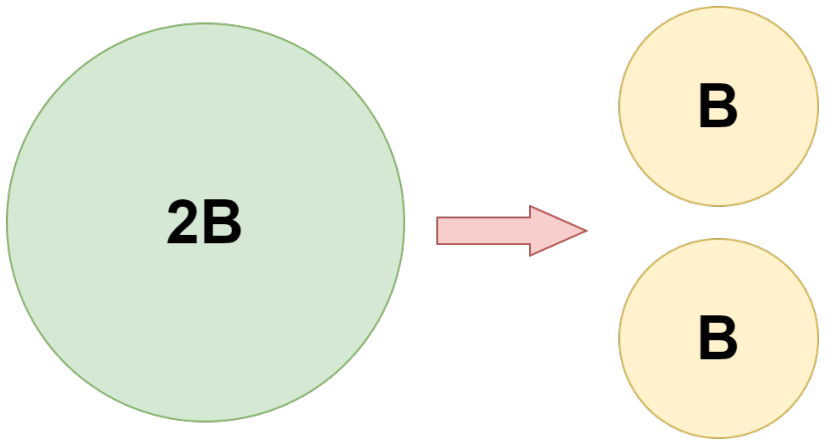
Ejemplo

Demostración: para demostrar su NP-completitud, debemos demostrar que el problema está en NP y que es NP-duro.

- El problema claramente está en NP: un certificado es la selección de ítems que van en cada uno de los k contenedores. Para verificar, basta confirmar que estén todos los ítems del problema; y que la suma de los pesos de los ítem en el contenedor i no sobrepasen su capacidad.

Ejemplo

- Para demostrar que es NP-duro, reducimos un problema NP-completo conocido a él. Un camino simple es reducir PARTITION (un problema que se sabe NP-Completo) a BINPACKAGING. En este problema, se tiene un multiconjunto de números naturales que suman $2B$, y se pregunta si es que hay alguna manera de dividirlo en dos conjuntos que sumen B.



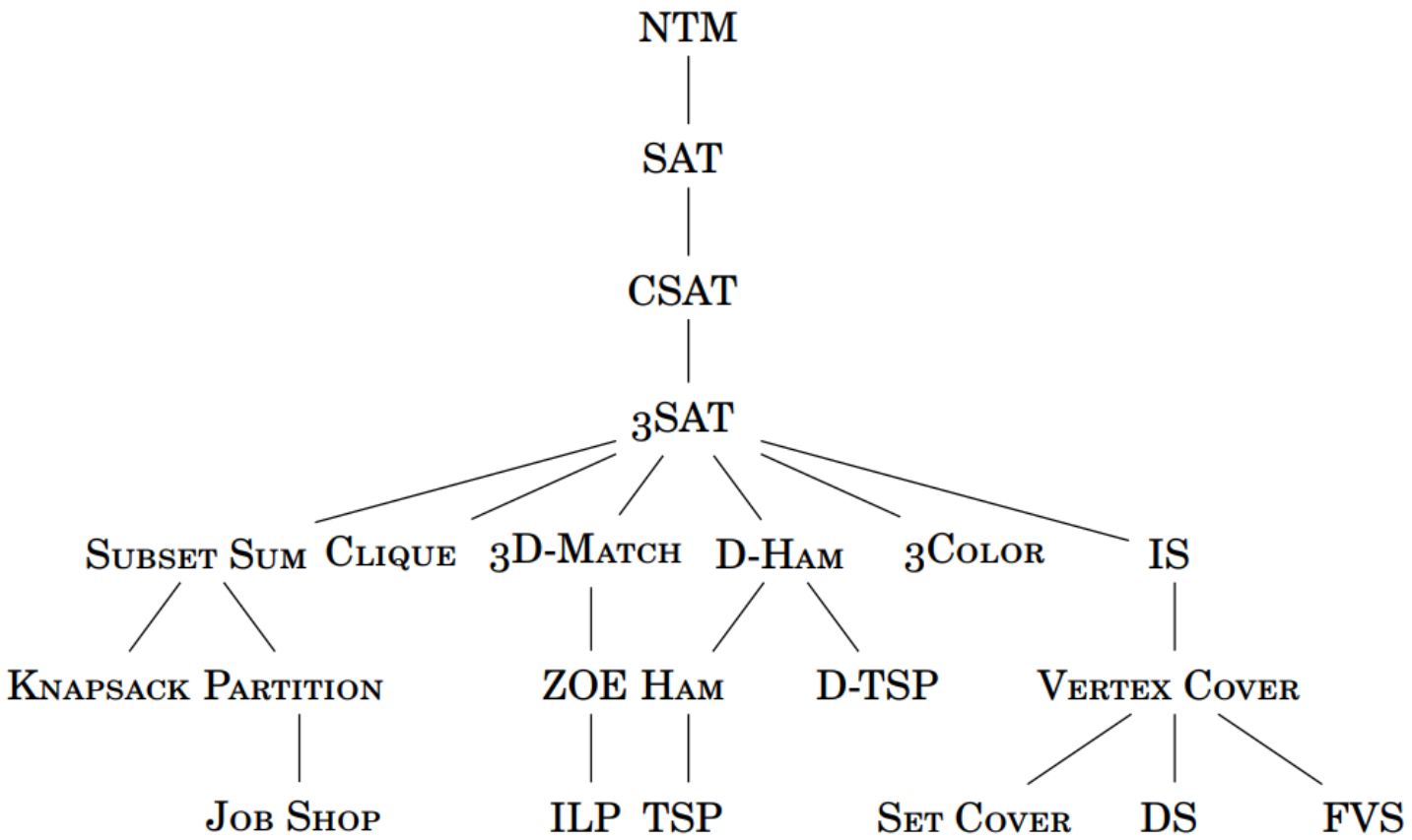
Ejemplo

Damos el mismo conjunto como pesos de los ítems, indicamos B como la capacidad de los contenedores, y preguntamos si se pueden almacenar en 2 contenedores. Si hay una partición, bastan dos contenedores; si bastan dos contenedores, los ítems en ellos definen una partición. Esto se puede realizar en tiempo polinomial.

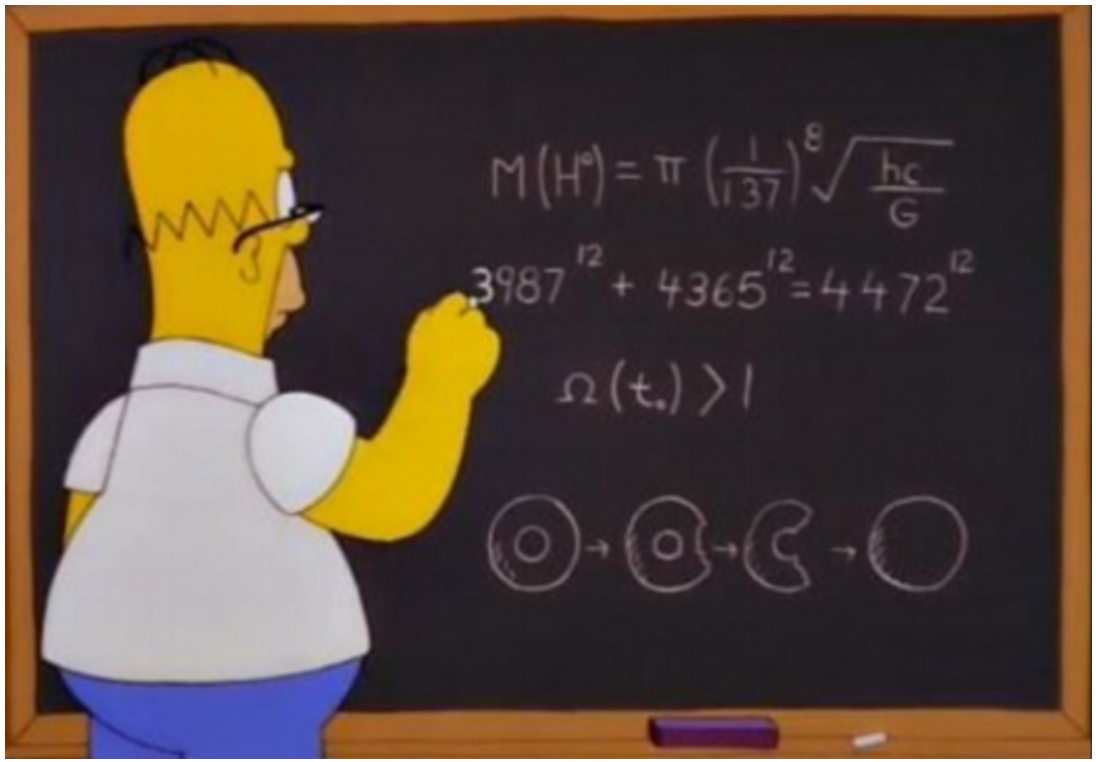
Como el problema está en NP y es NP-duro, es NP-completo.

□

Reducciones del curso



Ejercicios


$M(H^0) = \pi \left(\frac{1}{137}\right)^8 \sqrt{\frac{hc}{G}}$
$3987^{12} + 4365^{12} = 4472^{12}$
$\Omega(t_0) > 1$

Ejercicio 1

Demuestre que no es decidible si una máquina de Turing acepta al menos un palíndromo (una palabra que se lee igual al revés y al derecho, como la oración ¿Son mulas o cívicos alumnos?, donde estamos obviando la distinción entre letras acentuadas o no y omitimos espacios). Sea preciso.

Solución Ejercicio 1

La propiedad de contener palíndromos es no trivial (un lenguaje computacionalmente enumerable que contiene un palíndromo es, por ejemplo, $\{a\}$; uno que no contiene palíndromos es $\{ab\}$). Luego, por el teorema de Rice esto no es decidible.

Ejercicio 2

El señor Benesuela es un estudiante de TALF muy aplicado. Al llegar al Teorema de Rice, realiza las siguientes afirmaciones:

1. Determinar si un lenguaje es regular es no CE.
2. Determinar si una TM puede moverse a la izquierda y a la derecha es computable.
3. Determinar si un lenguaje es finito es CE no computable.
4. Determinar si el lenguaje aceptado por una TM está vacío es indecidible, no computable.
5. Determinar si el lenguaje aceptado por una TM es finito es computable.
6. Determinar si cierta TM tiene al menos k estados, para un k fijo como parámetro, es indecidible.

¿Las afirmaciones son correctas o incorrectas? Justifique para cada una de ellas el porqué está bien (o mal).

Solución Ejercicio 2

1. **Correcto.** Es una propiedad no trivial, por teorema de Rice, es indecidible.
2. **Correcto.** Basta leer su código binario y notar sus movimientos: si aparece izquierda y derecha entonces acepta, en caso contrario rechaza. Como siempre se detiene, es computable.
3. **Incorrecto.** Es una propiedad no trivial, por teorema de Rice, es indecidible.
4. **Correcto.** Por teorema de Rice (ya que existe $\mathscr{L}_e$ y $\mathscr{L}_{ne}$).
5. **Incorrecto.** Solo algunos lenguajes son finitos, por lo que es una propiedad no trivial. Por teorema de Rice, es indecidible.
6. **Incorrecto.** El teorema de Rice sólo habla acerca de las propiedades de los lenguajes, NO de las TM's que los aceptan. Para este caso basta con leer el código de la TM y contar el número de estados. Si tiene una cantidad de estados menor a k, rechaza. En caso contrario, acepta. Es computable.

Ejercicio 3

La legendaria Malondra está construyendo una TM malvada con sonidos. Para esto, ha etiquetado los estados como “estado-maullido” o “estado-silbato” lo que implica que cada vez que la TM visita dicho tipo de estado emite ese sonido. Luego se pregunta: si le entrego una entrada σ a mi TM ¿seré capaz de saber si en algún momento maúlla? Inmediatamente, concluye que eso no es computable. ¿Por qué llega a dicha conclusión?

Solución Ejercicio 3

Podemos notar que el problema anterior se puede interpretar de la siguiente manera:

- "estado-maullido" $\Leftrightarrow$ estado de aceptación.
- "estado-silbato" $\Leftrightarrow$ estado de rechazo.

Por lo que preguntar si en algún momento acepta es el problema de reconocer si $\sigma \in \mathscr{L}(M_{\mathscr{L}})$ (note que es similar al complemento del lenguaje diagonal), lo cual es CE no computable.

Ejercicio 4

El problema Knapsack propone un conjunto de ítem i, con el ítem i de peso w_i y valor v_i. Contamos con una mochila (*knapsack*) de capacidad C, queremos saber si podemos cargar la mochila sin sobrepasar la capacidad de valor al menos V.
El problema ILP (*Integer Linear Program*) da un conjunto de desigualdades lineales en las variables x_j de la forma:

$$\sum_j a_{ij} x_j \leq b_i$$

con la condición adicional que todos los coeficientes a_{ij}, las cotas b_j y las variables x_j son enteras, y pregunta si pueden satisfacerse todas las desigualdades simultáneamente.

Ejercicio 4

1. Describa una reducción de Knapsack a ILP, y demuestre que es polinomial.
2. Solo sabiendo que ILP es NP-completo, ¿qué puede concluir sobre Knapsack?

Solución Ejercicio 4

Note que las preguntas son independientes entre sí, puede responderse una sin necesariamente responder la otra, solo asumiendo la respuesta.

1. Necesitamos expresar las restricciones de KNAPSACK en términos de ILP, desigualdades lineales con coeficientes, cotas y variables enteras.
   - **Cada ítem se incluye o excluye:** Asociamos una variable x_i al ítem i. Incluirlo es darle valor 1, excluirlo es darle valor 0. Cada ítem da lugar a dos restricciones:

     $$x_i \leq 1$$
     $$-x_i \leq 0$$

     Estas tienen la forma requerida.

Solución Ejercicio 4

- **Capacidad de la mochila:** Da lugar a la restricción obvia:

$$\sum_i w_i x_i \leq C$$

  Es de la forma requerida.

- **Objetivo:** Buscamos valor de la mochila al menos V, que podemos describir como:

$$-\sum_i v_i x_i \leq -V$$

  Es de la forma requerida, usamos negativos ya que ILP exige desigualdades menor o igual.

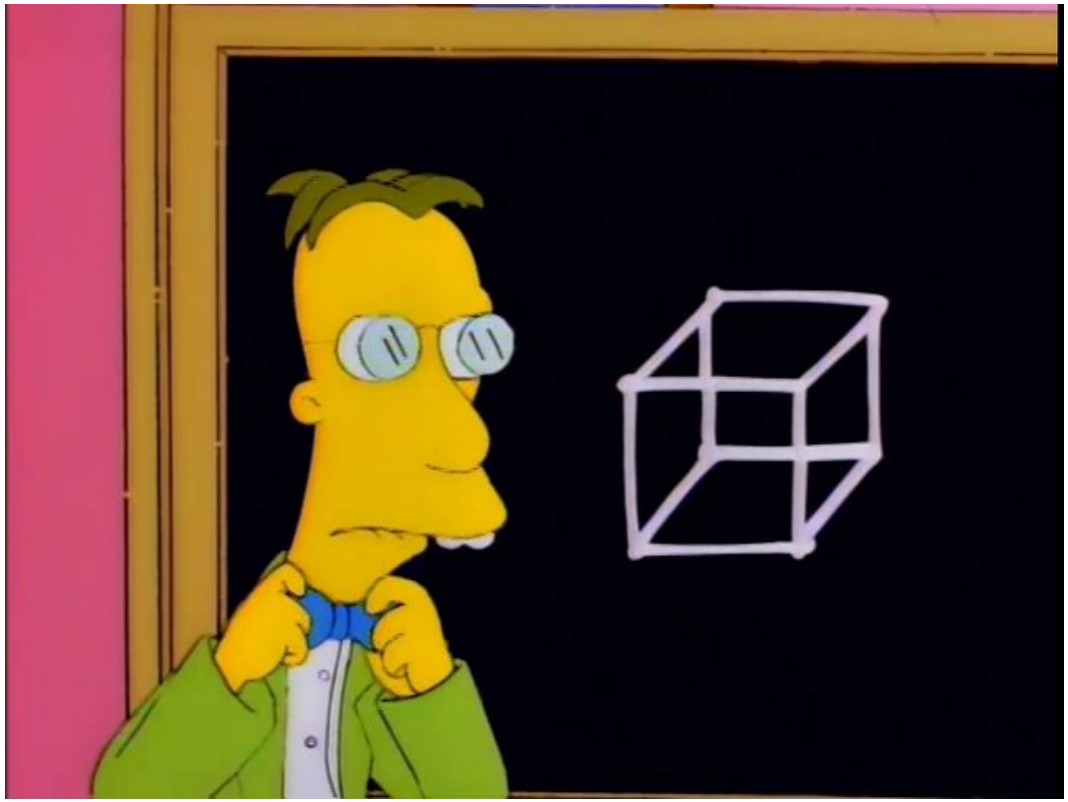
Solución Ejercicio 4

Es claro que el tamaño de las restricciones es proporcional al número de ítem, con lo que es proporcional al tamaño del problema. Escribir las desigualdades se puede hacer repasando el problema Knapsack una única vez, el proceso es polinomial. Queda claro de la derivación que si y solo si la instancia de Knapsack tiene respuesta afirmativa, la tiene la instancia de ILP así construida. Realmente es una reducción polinomial.

Solución Ejercicio 4

2. Solo permite concluir que KNAPSACK está en NP.
   En detalle, dada la pregunta ¿está x en KNAPSACK?, podemos usar la reducción polinomial a ILP para dar el $y$en la pregunta ¿está y en ILP?. Por ser polinomial la reducción, el tamaño de y está acotado por un polinomio en el tamaño de x. Como ILP está en NP, sabemos que hay una TM no determinista que resuelve la última pregunta en tiempo polinomial en el tamaño de y, que es polinomial en el tamaño de x; los tiempos son todos polinomiales.
   Hemos esbozado una máquina de Turing no determinista que acepta KNAPSACK.

¿Dudas?




LOONEY TUNES
"That's all Folks!"

INF155 - Informática Teórica

Ayudantía #9

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA