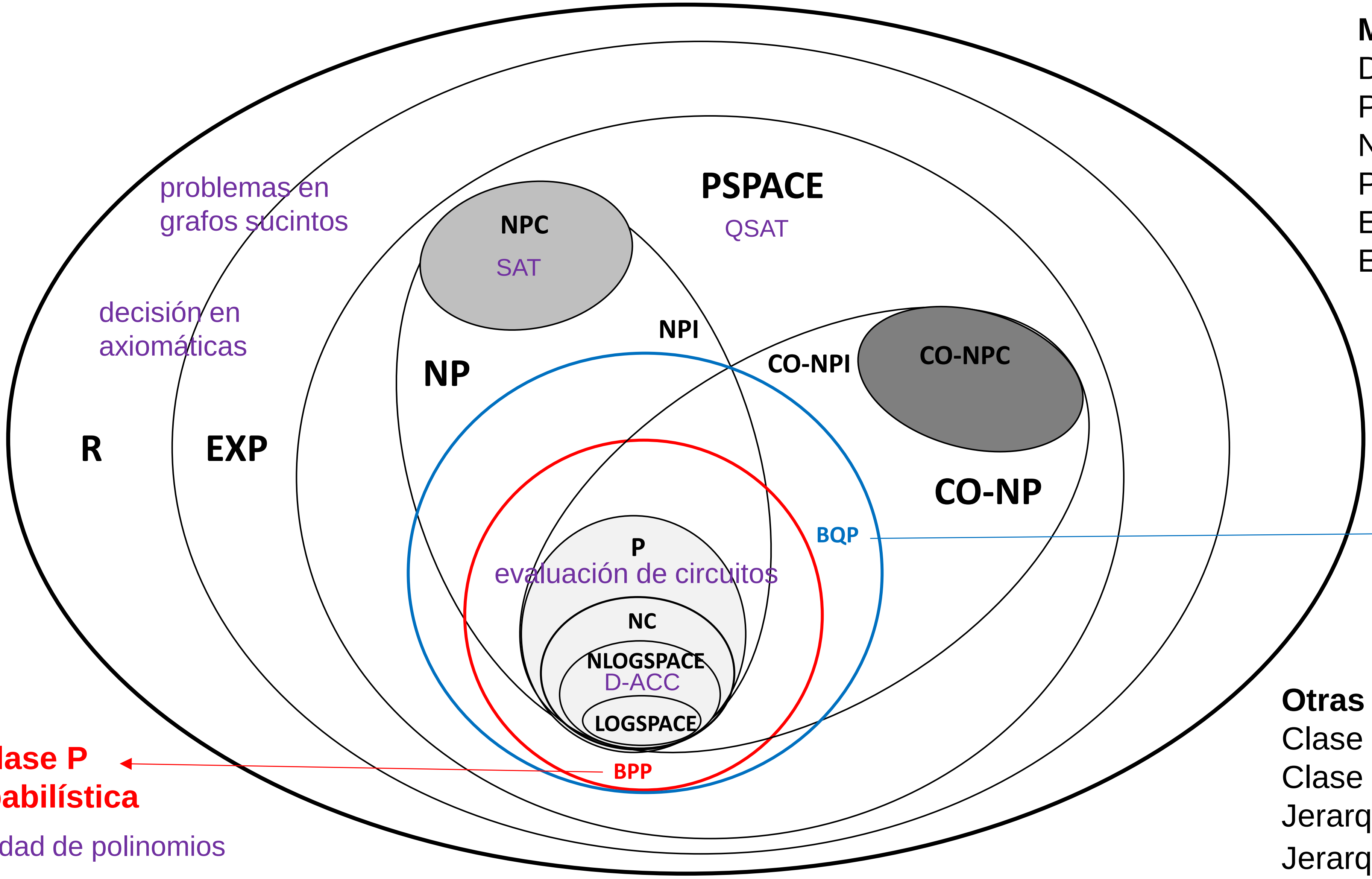
Clase teórica 9

Temas avanzados de complejidad computacional

(continuación)

Las jerarquías clásica y cuántica



Métodos de prueba
Diagonalización
P-completitud
NP-completitud
PSPACE-completitud
EXP-completitud
Etc.

La clase P cuántica

factorización

La clase P probabilística

identidad de polinomios

Otras clases y jerarquías
Clase #P
Clase IP
Jerarquía polinomial PH
Jerarquía PAS $\subset$ APX $\subset$ ONP

Profundizando en las MT probabilísticas

- Repaso:

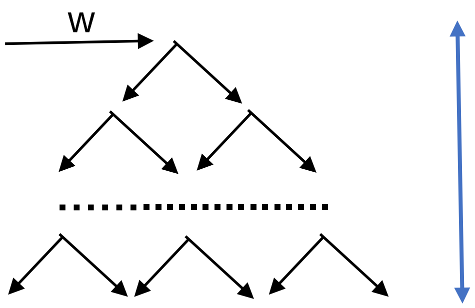
  - Una **MT probabilística (MTP),** en cada paso elige **aleatoriamente** una entre dos continuaciones, cada una con **probabilidad 1/2** ("tiro de moneda").

  - La clase P probabilística es **BPP** (*bounded probabilistic polynomial*):

  L ∈ BPP sii existe una MTP M con computaciones de tiempo poly(n) tal que, para toda cadena w:

    a) Si w ∈ L, entonces M acepta w en **al menos 2/3 de sus computaciones.**
    b) Si w ∉ L, entonces M rechaza w en **al menos 2/3 de sus computaciones.**

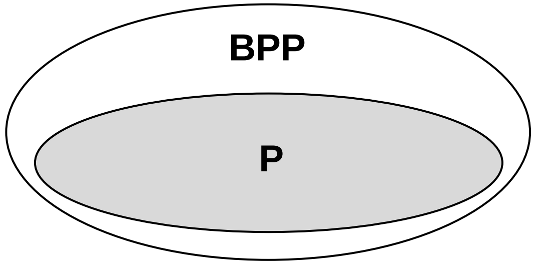
    (M tiene una **probabilidad de error ≤ 1/3**).



Computaciones de tiempo poly(|w|)

La MTP puede equivocarse con probabilidad ≤ 1/3

Pero ejecutando la MTP varias veces, la probabilidad de error se reduce



La conjetura más aceptada es que se cumple P = BPP!

(uso de generadores pseudoaleatorios)

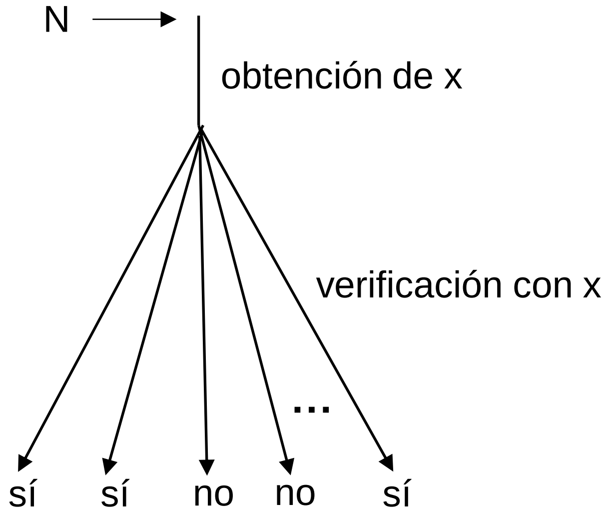
Ejemplo. Problema de composicionalidad COMP: “¿N es un número compuesto (no primo)?”

Se cumple que COMP = {N | N es un número compuesto} ∈ BPP.

- Existe una MTP M basada en lo siguiente: todo número compuesto impar N tiene **al menos (N – 1)/2 certificados de composicionalidad x en el intervalo |1, N – 1]**, en el sentido de que todo x permite verificar en tiempo poly(|N|) si N es compuesto.

- El esquema general de la MTP M es el siguiente. Dado un número N, M hace:

  1. Si N es par, acepta.
  2. Obtiene aleatoriamente un número x entre 1 y N – 1.
  3. Verifica si N es compuesto con la ayuda de x.



- Se cumple:
  - Si m es compuesto, **al menos la mitad de las computaciones de M aceptan**
  - Si m es primo, **todas las computaciones de M rechazan** (¿por qué?)

- De esta manera:
  - Si m es compuesto, **la probabilidad de error de M es ≤ 1/2.**
  - Si m es primo, **la probabilidad de error de M es 0** (¿por qué?)

- **COMP ∈ BPP:** ya ejecutando M dos veces la probabilidad de error de M es ≤ 1/3 (¿por qué?)

¿Y cuál es la probabilidad de error de M si se ejecuta p.ej. 10 veces? Si m es compuesto, es **≤ $1/2^{10}$**

- Otra clase probabilística, incluida en BPP, es **RP** (*randomized polynomial*):
  - L ∈ RP sii existe una MTP M con computaciones de tiempo poly(n) tal que, para toda cadena w:

    Si w ∈ L, entonces M acepta w en **al menos 1/2 de sus computaciones.** *** ver abajo (a)

    Si w ∉ L, entonces M rechaza w **en todas sus computaciones.** *** ver abajo (b)

    (M **nunca acepta mal,** y **rechaza mal con probabilidad ≤ 1/2**).

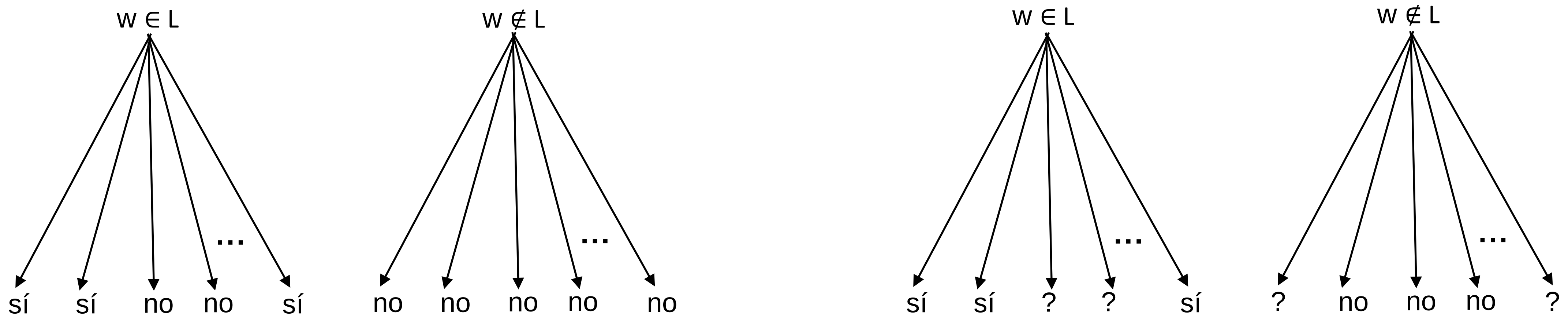
  Por ejemplo, **COMP ∈ RP**. También **PRIMOS = {N | N es un número primo} ∈ RP**.

- Y una tercera clase probabilística incluida en BPP es **ZPP** (*zero-error probabilistic polynomial*):
  - L ∈ ZPP sii existe una MTP M con computaciones de tiempo poly(n) tal que, para toda cadena w:

    Si w ∈ L, entonces M **acepta w con probabilidad ≥ 1/2 y rechaza con probabilidad 0.** *** ver abajo (c)

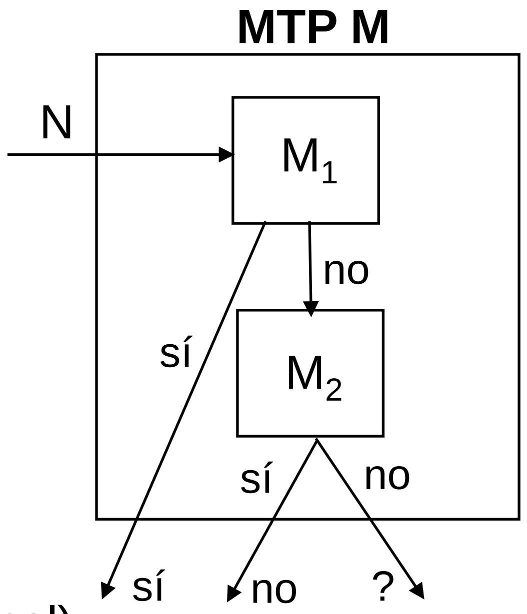
    Si w ∉ L, entonces M **rechaza con probabilidad ≥ 1/2 y acepta con probabilidad 0.** *** ver abajo (d)

    M **nunca se equivoca pero puede no responder nada** (tiene un tercer tipo de estado, "no sé" o "?").



caso (a) caso (b) caso (c) caso (d)

Ejemplo. **PRIMOS ∈ ZPP** (también COMP ∈ ZPP).



Sabemos que PRIMOS y COMP pertenecen a RP.

Así, existen una **MTP M_1 que decide PRIMOS** y una **MTP M_2 que decide COMP** tal que: tardan tiempo poly(n), nunca aceptan mal, y pueden rechazar mal con probabilidad $\leq 1/2$.

Vamos a construir una **MTP M que decida PRIMOS** y responda a la definición de ZPP:

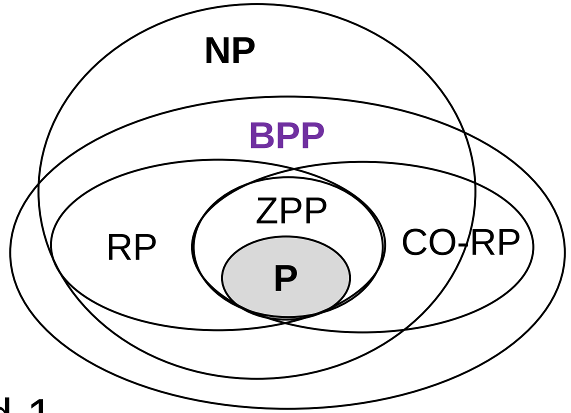
Sea la siguiente MTP M. Dado un número N, M hace:

1. Ejecuta M_1. Si acepta, **acepta** (N es primo porque M_1 decide PRIMOS y nunca acepta mal).
2. Ejecuta M_2. Si acepta, **rechaza** (N es compuesto porque M_2 decide COMP y nunca acepta mal).
3. Responde *no sé*.

- M tarda tiempo poly(n) porque M_1 y M_2 tardan tiempo poly(n)

- M nunca se equivoca (M₁ y M₂ nunca aceptan mal)

- La probabilidad de que M responda **no sé** (?) $\leq 1/2$ (¿por qué?)



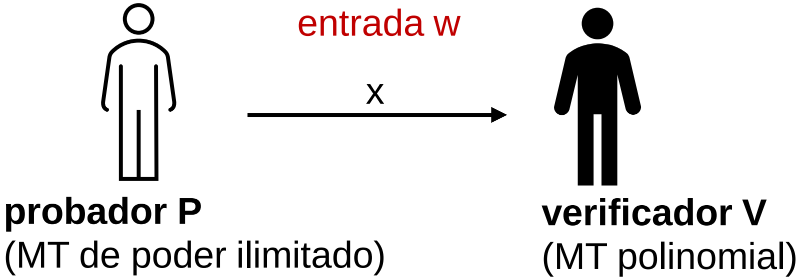
Si N es primo, M_1 responde **no** con probabilidad $\leq 1/2$ y M_2 responde **no** con probabilidad 1
Si N no es primo, M_1 responde **no** con probabilidad 1 y M_2 responde **no** con probabilidad $\leq 1/2$

Pero la conjetura es que **P = BPP**

- Ejecutando M por ejemplo 10 veces, la probabilidad de que M responda **no sé** baja a $\leq \mathbf{1/2^{10}}$

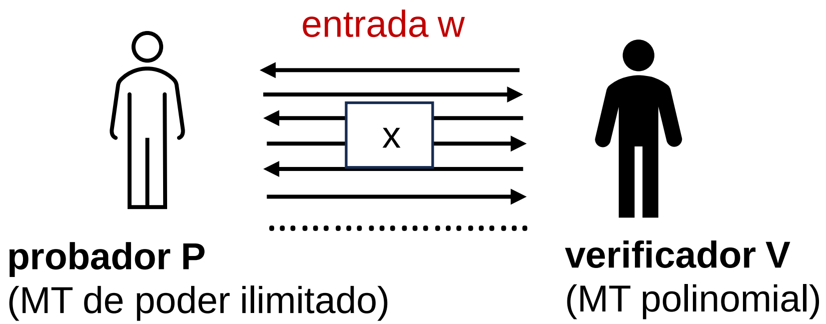
Profundizando en los sistemas interactivos

- **Vimos que un lenguaje L está en NP si cuenta con un verificador eficiente V (MT de tiempo polinomial):**



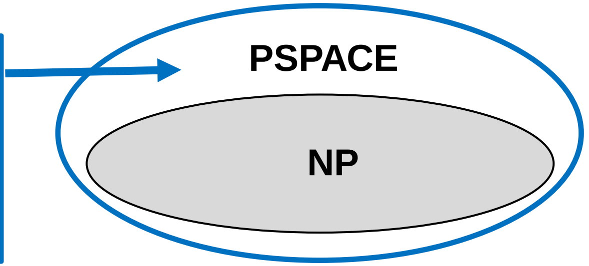
- Si w ∈ L, entonces existe un probador P (MT de poder ilimitado) que puede convencer a V de que w ∈ L, con la ayuda de un certificado x.
- Si w ∉ L, entonces no existe ningún probador P que pueda convencer a V de lo contrario, cualquiera sea el certificado x que utilice.

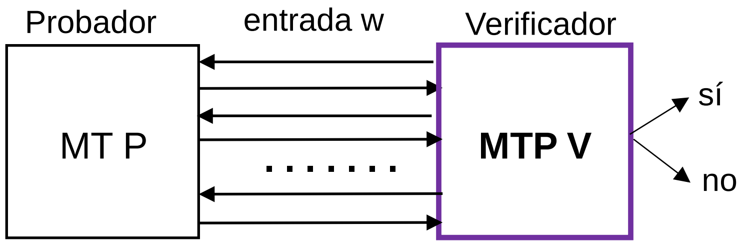
- **También vimos que un lenguaje L está en NP si puede ser verificado con un mecanismo más general, interactivo:**



- El verificador V y el probador P se intercambian poly(|w|) mensajes de tamaño poly(|w|) (V le envía preguntas a P y P le envía respuestas a V).
- En este caso, el certificado x incluye todos los intercambios entre P y V.

Si V es una **MT probabilística**, la clase de lenguajes decididos por un sistema interactivo es mucho más grande que NP, **¡alcanza a PSPACE!** Como en la clase BPP, se requiere que la probabilidad de error sea ≤ **1/3.**





En este contexto, la clase correspondiente se denomina **IP.**
La prueba de que **IP = PSPACE** se basa en la técnica de **algebraización.**

Ejemplo. Sistema interactivo probabilístico para decidir el complemento del problema del isomorfismo de grafos.

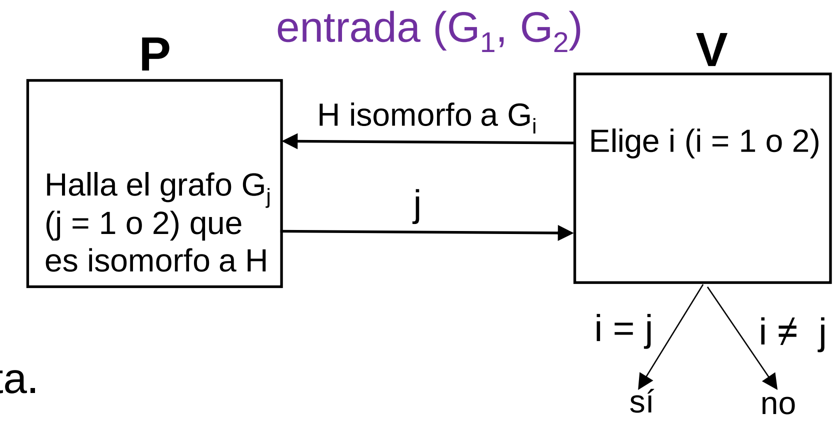
- Vimos que ISO = $\{(G_1, G_2) \mid G_1$ y G_2 son grafos isomorfos$\} \in$ NP.
  Los certificados sucintos son permutaciones de vértices (orden n).

- Y comentamos que **ISO^C = $\{(G_1, G_2) \mid G_1$ y G_2 no son grafos isomorfos$\}$ no estaría en NP**.
  Los certificados naturales no son sucintos: secuencias de todas las permutaciones posibles de vértices (orden n!).

- Así, como el alcance de un sistema (P, V) no probabilístico es NP, no decidiría ISO^C. **Pero sí lo hace si es probabilístico:**

Sea el siguiente sistema probabilístico (P, V). Dados dos grafos G_1 y G_2 con m vértices, (P, V) hace:

1. V elige un número i entre 1 y 2 y una permutación π de (1, …, m).
2. V obtiene el grafo H = π(G_i) (por lo tanto, H es isomorfo a G_i).
3. V le envía a P el grafo H.
4. P obtiene el número j entre 1 y 2 según H sea isomorfo a G_1 o a G_2.
5. P le envía a V el número j.
6. V acepta sii i = j.



<u>(P, V) tarda tiempo poly(|(G_1, G_2)|)</u>

- Las etapas 1, 2, 3 y 6 tardan tiempo poly(|(G_1, G_2)|). El tiempo de P no importa.
- Los mensajes intercambiados son 2.
- Los mensajes miden poly(|(G_1, G_2)|).

<u>(P, V) decide ISO^C con probabilidad de error ≤ 1/3</u>

- Si G_1 y G_2 **no son isomorfos**, V acepta con **probabilidad 1** (¿por qué?)
- Si G_1 y G_2 **son isomorfos**, V acepta con **probabilidad ≤ 1/2** (¿por qué?)
- Por lo tanto, con 2 ejecuciones de (P, V), la probabilidad de error ≤ 1/4
- **Esto vale con el probador P definido o cualquier otro**

Por lo tanto, en el marco de los sistemas interactivos probabilísticos, **las cadenas del lenguaje ISO^C cuentan con certificados sucintos.**

Profundizando en los algoritmos cuánticos

- **Repaso:** Un cúbit, como un bit, puede estar en los **estados básicos 0** o **1**, pero a diferencia del bit, puede estar también en un **estado de superposición de 0 y 1.** Los estados de un cúbit se expresan así:

Notación de Dirac		
|0>	|1>	α_0|0> + α_1|1>

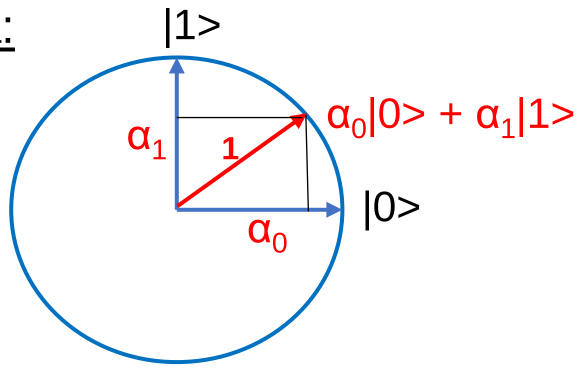
Notación vectorial		
(1, 0)	(0, 1)	(α_0, α_1)

Los coeficientes α_0 y α_1 son números **complejos**, conocidos como **amplitudes**, y cumplen **$|\alpha_0|^2 + |\alpha_1|^2 = 1$,** lo que se interpreta del siguiente modo:

- El cúbit está en un estado de superposición de 0 y 1.
- Al **medirlo** (leerlo), se obtiene el valor 0 con **probabilidad $|\alpha_0|^2$** o el valor 1 con **probabilidad $|\alpha_1|^2$.**
- Luego de la medición **se destruye la superposición** (el cúbit queda en el estado básico obtenido).

- Con amplitudes **reales** ya se manifiesta el poder cuántico (en lo que sigue utilizamos sólo números reales).

- Representación gráfica:



Con amplitudes reales, la representación gráfica habitual del estado de un cúbit es un **vector de tamaño 1** que rota alrededor de una circunferencia.

Como se refleja en el gráfico, los estados que puede adoptar un cúbit son **infinitos** (cualquier punto de la circunferencia).

- Los cubits se agrupan en **registros cuánticos.**

- El estado de un registro cuántico de m cubits puede alcanzar una superposición de **2^m estados básicos**.

- Por ejemplo, un registro de 2 cubits puede estar en un estado básico **|00>, |01>, |10> o |11>,**
  o en un estado de superposición $\boldsymbol{\alpha_{00}|00> + \alpha_{01}|01> + \alpha_{10}|10> + \alpha_{11}|11>}$**,**
  con $\boldsymbol{\alpha_{00}^2 + \alpha_{01}^2 + \alpha_{10}^2 + \alpha_{11}^2 = 1}$**.**

- En este último caso, el estado del registro se representa mediante el vector de amplitudes $\boldsymbol{(\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11})}$**,**
  y cuando el registro se mide pasa con probabilidad $\boldsymbol{\alpha_{xy}^2}$ al estado básico **|xy>,** siendo x e y = 0 o 1,
  quedando la amplitud correspondiente en 1 y las restantes en 0.

- Lo mismo sucede con los registros de 3 cubits, 4 cubits, etc.

- Por ejemplo, si un registro cuántico de 10 cubits en un momento dado está en el estado de superposición:

|0000000000> ⟶ con probabilidad α^2
|0000011010> ⟶ con probabilidad β^2
|0111000001> ⟶ con probabilidad γ^2
|0111110011> ⟶ con probabilidad δ^2

si al leerlo se obtiene |0111000001>, entonces queda **solamente** en este estado,
y por lo tanto su amplitud γ queda en **1** y las $2^{10} - 1$ amplitudes restantes quedan en **0**.

- El uso de **amplitudes negativas** marcarían la diferencia fundamental entre lo clásico y lo cuántico:

  - Un cúbit en el estado (1/√2)|0> + (1/√2)|1>, al medirse pasa a |0> con prob. 1/2, o bien a |1> con prob. 1/2.
  - Un cúbit en el estado (1/√2)|0> – (1/√2)|1>, al medirse se comporta de la misma manera (¿por qué?)

  Pero los estados son distintos (tiene **fase relativa** distinta): aplicándoles operaciones **los resultados difieren**.

- Las operaciones cuánticas se conocen como **puertas cuánticas,** que se aplican a **uno, dos o tres cubits**.

- Las puertas cuánticas se representan con **matrices**: aplicar una puerta cuántica sobre un registro consiste en multiplicar la matriz que la representa por el vector que representa el estado del registro.

- Por ejemplo, la **puerta de Hadamard (H)** se aplica sobre un cúbit y se representa por la siguiente matriz:

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$$

Aplicando H a **|0>** se pasa a **(1/√2)|0> + (1/√2)|1>:**

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

amplitudes de |0> y |1>

Aplicando H a **|1> se pasa a (1/√2)|0> – (1/√2)|1>:**

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

amplitudes de |0> y |1>

¿A qué operación de las MT probabilísticas recuerda H? A la elección aleatoria (“tiro de moneda”).

- Siguiendo con la puerta de Hadamard o H, vimos recién que aplicada a **|0>** se obtiene **(1/√2)|0> + (1/√2)|1>:**

$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

amplitudes de |0> y |1>

Si aplicamos nuevamente H, ahora al estado obtenido, se obtiene:

Ejercicio: comprobar que haciendo lo mismo pero a partir de |1> se vuelve a |1>

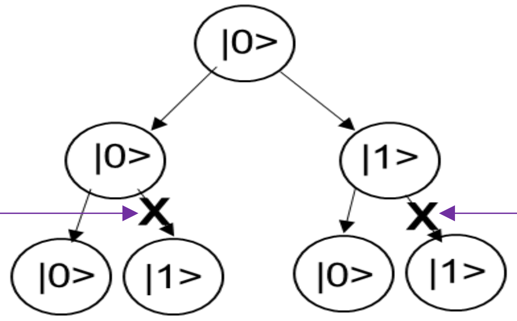
$$\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \cdot \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

amplitudes de |0> y |1>

es decir que **¡se vuelve al estado básico original |0>!**

- O sea que aplicando a un estado de superposición una puerta cuántica que pasa un estado a un estado de supersposición, se obtiene un estado básico. La explicación es la siguiente:

amplitud positiva de |1>
(ver slide anterior)

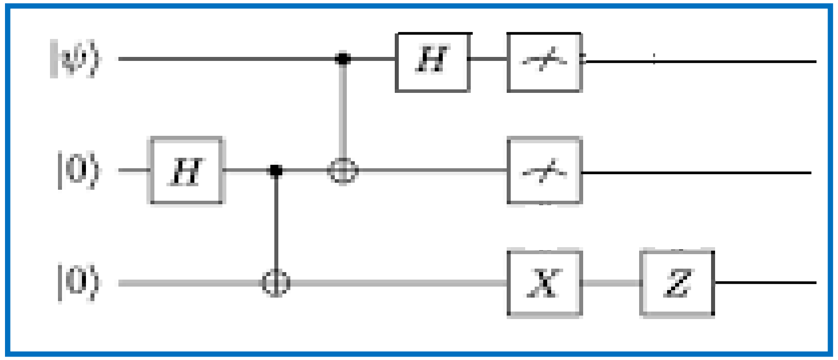


amplitud negativa de |1>
(ver slide anterior)

Desde (1/√2)|0> + (1/√2)|1> hay dos caminos que llevan a |0> y dos caminos que llevan a |1>. Pero mientras que en los primeros las amplitudes son positivas (interferencia positiva), en los segundos una amplitud es positiva y la otra es negativa (interferencia negativa). Así, en la medición se obtiene |0> con probabilidad 1.

- Una **computación cuántica** es una secuencia de aplicaciones de puertas cuánticas. El esquema estándar es:
    1. **Inicio.** Preparación de la cadena de entrada w (estado cuántico inicial del sistema).
    2. **Evoluciones.** Secuencia de aplicaciones de puertas cuánticas.
    3. **Medición.** Lectura del estado cuántico obtenido.

- El modelo computacional habitual es el **circuito cuántico**. P.ej., sea el siguiente fragmento de circuito:
    - La entrada w es un cúbit en estado |ψ>, seguido de dos cubits en estado |0>.
    - En el circuito, primero se aplica H al segundo cúbit, luego CNOT (NOT controlado) al segundo y tercer cúbit, luego CNOT al primer y segundo cúbit, luego H al primer cúbit, luego se miden el primer y segundo cúbit y se aplica X al tercer cúbit, y finalmente se aplica Z al tercer cúbit.



- En la computación cuántica, todas las puertas son **reversibles**: aplicándolas dos veces seguidas, la segunda aplicación cancela el efecto de la primera (se vuelve al estado original).

- El **tiempo de ejecución** de un circuito cuántico se establece por el número de sus puertas cuánticas.

- Los lenguajes decidibles por circuitos cuánticos en tiempo poly(n), y con probabilidad de error $\leq 1/3$, forman la clase **BQP** (*Bounded error, Quantum, Polynomial time).*

- Como en el caso de las MTP, iterando varias ejecuciones un circuito cuántico la probabilidad de error se puede decrementar significativamente.

- Se cumple **P $\subseteq$ BQP,** porque las MT de tiempo poly(n) se pueden simular con circuitos booleanos de tamaño poly(n), que a su vez se pueden simular con circuitos cuánticos de tamaño poly(n).

- También se cumple **BPP $\subseteq$ BQP,** porque los "tiros de moneda" de las MTP (y circuitos booleanos probabilísticos) se pueden simular en los circuitos cuánticos con aplicaciones de la puerta de Hadamard.

- **No se cumpliría BPP = BQP**: hasta el momento no se ha encontrado ninguna MTP que factorice en tiempo poly(n) (el algoritmo cuántico de Shor factoriza en tiempo poly(n)).

- Otra inclusión que se prueba es **BQP $\subseteq$ PSPACE** (recurriendo a un procedimiento recursivo y la reutilización de espacio).

- Y con respecto a **BQP vs NP**, la conjetura aceptada es que **son incomparables**.

  - En particular, <u>BQP no incluiría a los lenguajes NP-completos</u>. Por ejemplo, el algoritmo cuántico de Grover acelera los algoritmos de búsqueda clásicos de fuerza bruta sólo un tiempo cuadrático.

  - Por otro lado, habrían lenguajes de BQP no pertenecientes a la jerarquía polinomial PH (la cual incluye a NP).

Idea general del algoritmo cuántico de búsqueda (algoritmo de Grover)

- **Planteo**: dada una función $f : \{0, 1\}^n \longrightarrow \{0, 1\}$, se quiere encontrar algún $x \in \{0, 1\}^n$ tal que $f(x) = 1$.

- **Ejemplo con SAT:** dadas las $N = 2^n$ asignaciones $\mathcal{A}$ posibles a una fórmula booleana φ de n variables, se quiere encontrar alguna $\mathcal{A}$ que satisfaga φ. En este caso, x es una asignación $\mathcal{A}$, y f(x) = 1 sii $\mathcal{A}$ satisface φ.

- **Complejidad temporal clásica vs cuántica:** si hay N posibles soluciones, en el modelo clásico el tiempo requerido es O(N), mientras que en el modelo cuántico el tiempo baja a $O(\sqrt{N})$ - aceleración cuadrática -.

Esquema general del algoritmo cuántico

1. Se parte de n cubits, cada uno en el estado básico |0>.
2. Se aplica H a los n cubits, obteniéndose:

$$\mathbf{(1/\sqrt{2^n})|0\ldots00\rangle + (1/\sqrt{2^n})\,|0\ldots01\rangle + \ldots + (1/\sqrt{2^n})\,|1\ldots10\rangle + (1/\sqrt{2^n})\,|1\ldots11\rangle}$$

   es decir, **se superponen las $N = 2^n$ posibles soluciones |x>, cada una con la misma amplitud α_x**

3. Se itera K veces, con $K = O(\sqrt{N})$:
   Se aplican puertas cuánticas específicas que:
   amplifican las amplitudes de los |x> tales que **f(x) = 1,**
   decrementan las amplitudes de los |x> tales que **f(x) = 0**.
4. Finalmente se mide el registro. Se comprueba que se obtiene una solución x con **muy alta probabilidad.**

Representación gráfica del algoritmo cuántico de Grover

Supongamos que sólo $x_0 = 00\ldots00$ es solución, es decir, sólo $f(x_0) = 1$.

Iteración 0

$|x_0> = |00\ldots00>$ → amplitud $\alpha_0 = 1/\sqrt{2^n}$
$|x_1> = |00\ldots01>$ → amplitud $\alpha_1 = 1/\sqrt{2^n}$
$|x_2> = |00\ldots10>$ → amplitud $\alpha_2 = 1/\sqrt{2^n}$
………..……… ……………………………..
$|x_{N-2}> = |11\ldots10>$ → amplitud $\alpha_{N-2} = 1/\sqrt{2^n}$
$|x_{N-1}> = |11\ldots11>$ → amplitud $\alpha_{N-1} = 1/\sqrt{2^n}$

Aplicación de puertas H

Se superponen todas las posibles soluciones, todas con la misma probabilidad de medición.

Iteración 1

$|x_0> = |00\ldots00>$ → amplitud α_0 > anterior
$|x_1> = |00\ldots01>$ → amplitud α_1 < anterior
$|x_2> = |00\ldots10>$ → amplitud α_2 < anterior
………..……… ……………………………..
$|x_{N-2}> = |11\ldots10>$ → amplitud α_{N-2} < anterior
$|x_{N-1}> = |11\ldots11>$ → amplitud α_{N-1} < anterior

Aplicación de la inversión sobre la media

Se evalúan **simultáneamente** todos los estados, amplificándose la amplitud de $|x_0>$ y decrementándose las amplitudes de los estados restantes.

Iteración 2

$|x_0> = |00\ldots00>$ → amplitud α_0 > anterior
$|x_1> = |00\ldots01>$ → amplitud α_1 < anterior
$|x_2> = |00\ldots10>$ → amplitud α_2 < anterior
………..……… ……………………………..
$|x_{N-2}> = |11\ldots10>$ → amplitud α_{N-2} < anterior
$|x_{N-1}> = |11\ldots11>$ → amplitud α_{N-1} < anterior

Aplicación de la inversión sobre la media

Se evalúan **simultáneamente** todos los estados, amplificándose la amplitud de $|x_0>$ y decrementándose las amplitudes de los estados restantes.

Y así siguiendo hasta la iteración K, luego de la cual se mide el registro y **se obtiene x_0 con alta probabilidad.**

- **Ejemplo de aplicación del algoritmo cuántico de Grover (se describen las puertas cuánticas).**

- Sea una lista de $N = 2^6 = 64$ posibles soluciones x_i,
  y supongamos que sólo x_0 = |000000> cumple la condición $f(x_0) = 1$.

- Primero se construye la superposición **1/8 |000000>** + 1/8 |000001> + … +1/8 |111110> + 1/8 |111111>.
  Es decir, $\alpha_0 = \alpha_1 = \ldots = \alpha_{63} = 1/8$.

- Luego:
  a. Se cambia α_0 = 1/8 por –1/8 = –0,125.
  b. Se calcula el promedio P con las nuevas amplitudes: (–1/8 + 63*1/8) / 64 = 0,12109.
  c. Se modifican todas las amplitudes α_i por $(2P - \alpha_i)$, obteniéndose:

  0,367817 |000000> + 0,117187 |000001> + … + 0,117187 |111110> + 0,117187 |111111>

- Repitiendo el proceso 6 veces, se llega a:

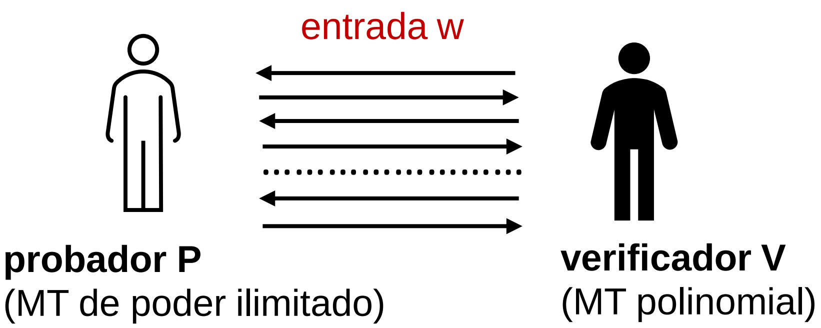
  0,998291 |000000> – 0,00736174 |000001> – … – 0,00736174 |111110> – 0,00736174 |111111>

- Finalmente, midiendo el estado, la probabilidad de encontrar x_0 = |000000> es $\mathbf{0{,}998291^2}$.

Nota: si se sigue iterando no se mejora sino que se empeora el resultado.
P.ej., después de 10 iteraciones, la amplitud de x_0 = |000000> es 0,487922.

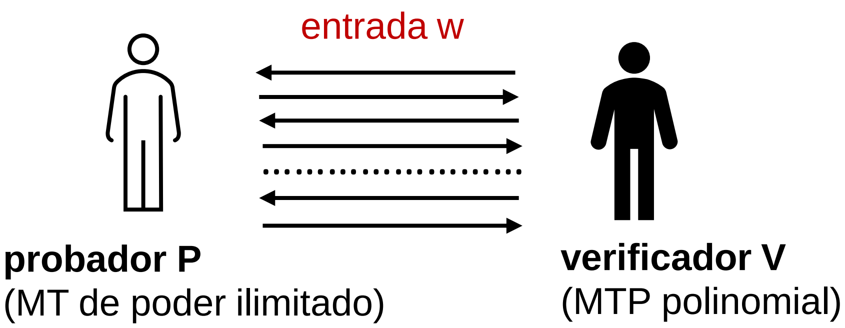
Anexo

Algo más sobre la clase NP y los sistemas de pruebas (1)



Sistema interactivo (P, V)

- El verificador V es una MT polinomial. Intercambia con el probador P poly(|w|) mensajes de tamaño poly(|w|) - prueba interactiva -.
- La clase de lenguajes aceptados por este modelo es **NP**.



Sistema interactivo probabilístico (P, V)

- El verificador V es una MTP polinomial. Intercambia con el probador P poly(|w|) mensajes de tamaño poly(|w|) - prueba interactiva probabilística -.
- La clase de lenguajes aceptados por este modelo es **PSPACE**.

- Un caso particular de prueba interactiva probabilística, de sumo interés para la criptografía, es la **prueba de conocimiento cero**, con la que V no aprende nada de lo que le envía P (sólo le alcanza para aceptar o rechazar w).

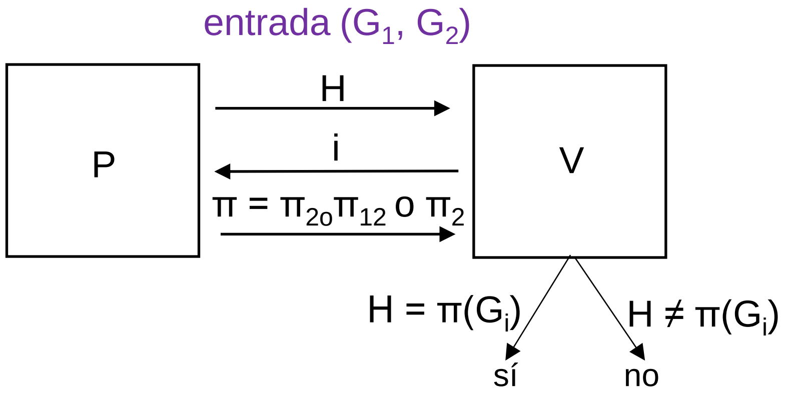
Con ciertas asunciones, se prueba que todo lenguaje L de NP cuenta con una prueba de conocimiento cero.

Ejemplo. Prueba de conocimiento cero para el lenguaje ISO (¿Qué prueba de conocimiento NO cero conoce?)

ISO = {(G_1, G_2) | G_1 y G_2 son grafos isomorfos}

Sea el siguiente sistema interactivo probabilístico (P, V). Dados dos grafos G_1 y G_2 con m vértices, (P, V) hace:

1. P obtiene dos permutaciones de vértices: π_{12}, que si G_1 y G_2 son isomorfos cumple $\pi_{12}(G_1) = G_2$, y π_2; y el grafo $H = \pi_2(G_2)$.
2. P le envía a V el grafo H.
3. V elige un número i entre 1 y 2.
4. V le envía a P el número i.
5. Si i = 1, P le envía a V la permutación $\pi = \pi_2 \circ \pi_{12}$, con $\pi_2 \circ \pi_{12}(G) = \pi_2(\pi_{12}(G))$.
   Si i = 2, P le envía a V la permutación $\pi = \pi_2$.
6. V acepta sii $H = \pi(G_i)$.



- Claramente. (P, V) tarda tiempo polinomial.

- Además:

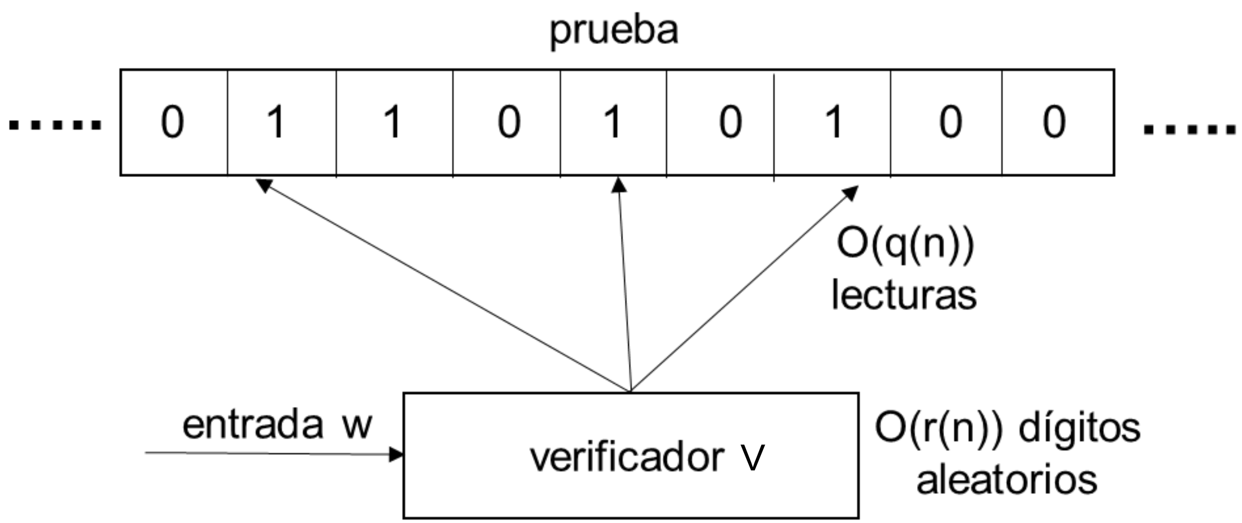
  Si G_1 y G_2 son isomorfos, V acepta con probabilidad 1, porque elija i = 1 o 2, en (6) siempre compara dos grafos iguales:
  cuando i = 1, compara $\pi_2(G_2)$ con $\pi_2(\pi_{12}(G_1))$, siendo $\pi_{12}(G_1) = G_2$
  cuando i = 2, compara $\pi_2(G_2)$ con $\pi_2(G_2)$

  Si G_1 y G_2 no son isomorfos, V acepta con probabilidad $\leq$ 1/2 (con el probador P definido o con cualquier otro - esto último se puede probar -). Con la estrategia del ejemplo:
  si V elige 1, compara $\pi_2(G_2)$ con $\pi_2(\pi_{12}(G_1))$, que son grafos distintos
  si V elige 2, compara $\pi_2(G_2)$ con $\pi_2(G_2)$, que son grafos iguales.

  Con dos ejecuciones de (P, V) se alcanza la cota requerida $\leq$ 1/3.

Con lo que le envía P, V **no aprende nada** acerca del isomorfismo existente entre los grafos G_1 y G_2.

Algo más sobre la clase NP y los sistemas de pruebas (2)



<u>Verificador V para un lenguaje L</u>
Si w ∈ L, existe una prueba tal que la probabilidad de que V acepte w es 1.
Si w ∉ L, para toda prueba, la probabilidad de que V acepte w es a lo sumo 1/2.

Prueba chequeable probabilísticamente (PCP). Un verificador V recibe de un probador P una prueba de una cadena w, que V debe aceptar o rechazar. V es una MTP polinomial, y P es una MT de poder ilimitado. Además, dadas dos funciones r(n) y q(n):

- V puede utilizar O(r(n)) dígitos aleatorios.
- V puede hacer O(q(n)) consultas sobre la prueba (lecturas independientes de un solo dígito, 1 o 0).

Se prueba (**Teorema PCP**) que todo lenguaje de NP puede ser decidido por un sistema de estas características, en el que V utiliza sólo **O($\log_2$n) dígitos aleatorios** y efectúa sólo **O(1) consultas** (**muy contraintuitivo**).

Este teorema se usa para probar que varios problemas de optimización no tienen aproximaciones polinomiales.