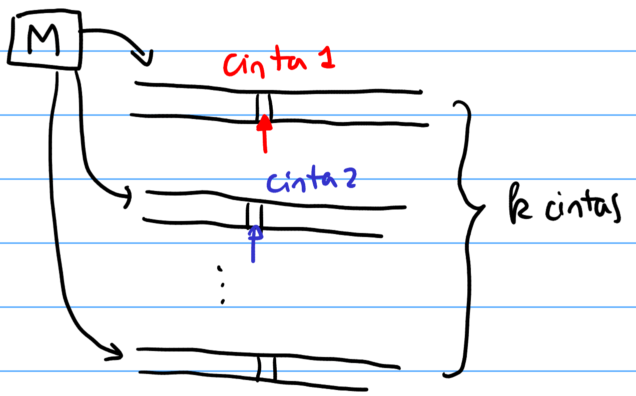
Recordar:

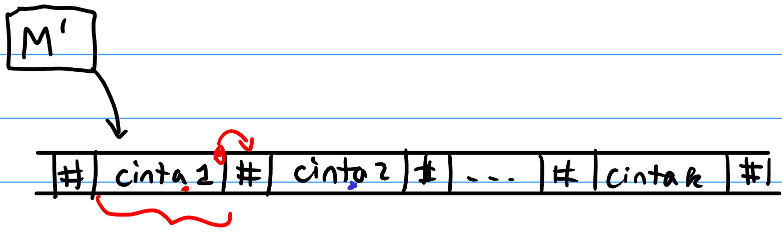
- Dada una M.T. M, que siempre termina, el $\boxed{\text{tiempo de ejecución}}$ (en peor caso) de M es una función $T: \mathbb{N} \to \mathbb{N}$ definida por:

$$T(n) = \max \{ \text{tiempo que tarda } M(w) \text{ en terminar} \mid w \in \Sigma^*, |w| = n \}.$$

Teorema. Si M es una M.T. multicinta con tiempo de ejecución $T(n)$, existe una M.T. M' equivalente a M de una sola cinta cuyo tiempo de ejecución es $O(T(n)^2)$.

Dem. (Idea).





- Simular un paso de la máquina M requiere tiempo proporcional al tamaño de la cinta entera de M'.
- ¿Qué tamaño tiene la cinta de M'?
  - Inicialmente tiene tamaño k.
  - Después de hacer un paso de simulación de la máquina M, la cinta de M' tiene tamaño a lo sumo $2k$.
  - Después de simular dos pasos de M, la cinta de M' tiene tamaño a lo sumo $3k$.

  $\vdots$

  - Después del i-ésimo paso tiene tamaño a lo sumo $(i+1)k$.

¿Cuál es el costo total de la simulación?

- 1er paso: Cuesta k
- 2do paso: cuesta a lo sumo $2k$
- 3er paso: cuesta a lo sumo $3k$

$\vdots$

- i-ésimo paso: cuesta a lo sumo $i \cdot k$

Dada una entrada w de tamaño $|w| = n$,
la máquina M requiere $T(n)$ pasos para terminar.
Por lo tanto M' requiere:

$$\sum_{i=1}^{T(n)} i \cdot k = k \sum_{i=1}^{T(n)} i = \frac{T(n) \cdot (T(n)+1)}{2} \cdot k \in O\left(T(n)^2\right)$$

Recordar.

- Una M.T. es una 7-upla

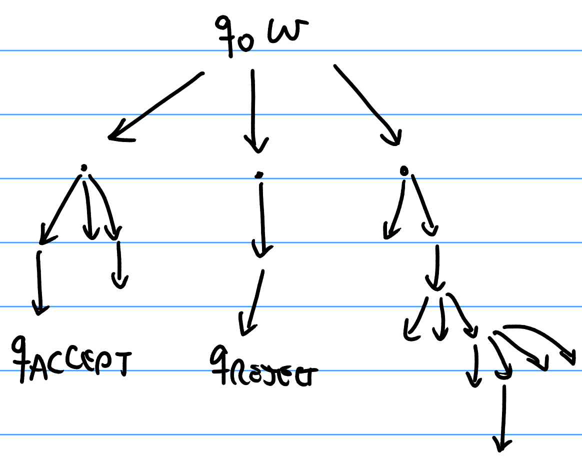
$$(\Sigma, \Gamma, Q, \delta, q_0, q_{ACCEPT}, q_{REJECT})$$

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

- Una M.T. no determinística (M.T.N.) es una 7-upla

$$(\Sigma, \Gamma, Q, \delta, q_0, q_{ACCEPT}, q_{REJECT})$$

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$



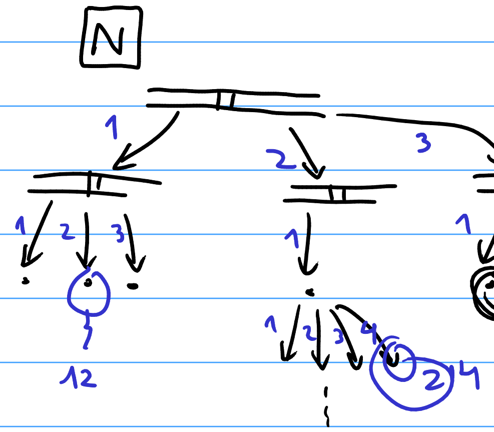
- Decimos que una M.T.N. <u>siempre termina</u> si todas las ramas terminan.

<u>Def</u>. Dada una M.T.N. M que siempre termina, decimos que su <u>tiempo de ejecución</u> (en peor caso) es la función $T : \mathbb{N} \longrightarrow \mathbb{N}$ definida así:

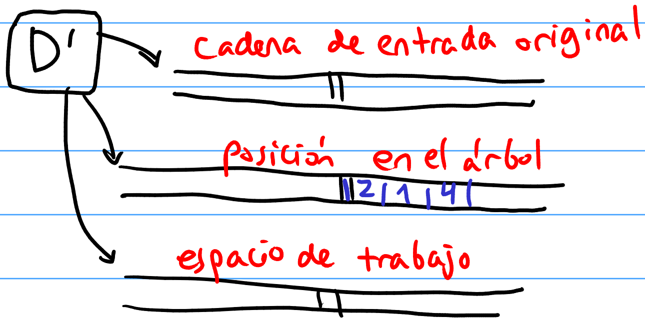
$$T(n) = \max \{ \text{altura del árbol de ejecución } M(w) \mid w \in \Sigma^*, |w| = n \}.$$

Teorema. Dada una M.T.N. N con tiempo de ejecución $T(n)$, existe una M.T. D equivalente a N con tiempo de ejecución $O(2^{T(n)})$.

Dem. (Idea).

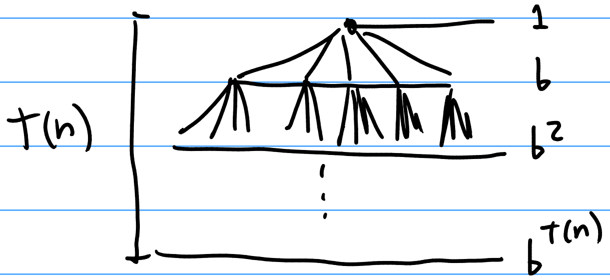


Construyamos primero una M.T. multicinta D' equivalente a N.



¿Cuánto cuesta simular N?

- Recordemos que N siempre termina y tiene tiempo de ejecución $T(n)$, el árbol de configuraciones es finito y tiene altura $T(n)$.
- Además, cada configuración tiene un cierto número de hijos. Como máximo, cada configuración tiene b hijos.
- El tamaño del árbol (cantidad de nodos)



es

$$\sum_{i=0}^{T(n)} b^i = \frac{b^{T(n)+1} - 1}{b-1} \leq \boxed{b^{T(n)+1}}.$$

- Además, simular la ejecución de una rama para llegar hasta un cierto nodo, va a costar $T(n)$ en el peor caso.

- Por lo tanto, el tiempo de ejecución de D' es a lo sumo:

$$T(n)\cdot b^{T(n)+1}$$

- Por el teorema anterior, existe una M.T. de una sola cinta D cuyo tiempo de ejecución es

$$O\left(\left(T(n)\cdot b^{T(n)+1}\right)^2\right)$$
$$= O\left(T(n)^2\cdot \left(b^{T(n)+1}\right)^2\right)$$
$$= O\left(T(n)^2\cdot b^{2T(n+2)}\right) \subseteq 2^{O(T(n))}.$$

Problemas tratables vs. intratables.

- Un problema es tratable si se puede resolver en tiempo polinomial, es decir: $O(n^k)$
- Si no, es intratable.

	n^3	2^n
$n=1000$	1.000.000.000	(astronómico)

- Los modelos de cómputo razonables se pueden simular uno a otro con costo extra polinomial.

Ej.

M.T. multicinta	$O(n^3)$	$O(n^k)$
M.T.	$O(n^6)$	$O(n^{2k})$
M.T.N.	$2^{O(n^6)}$	

- Ej.: Determinar si un número es primo por fuerza bruta toma tiempo exponencial.

$n \qquad O(n)$

$\boxed{1\,|\,1\,|\,0\,|\,1\,|\,0\,|\,1\,|\,0\,|\,1}$

t bits $\qquad O(2^t)$

$\parallel$

$\log_2(n)$

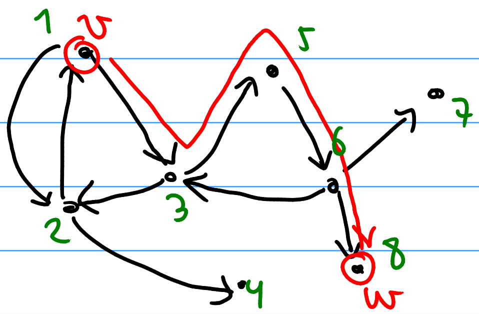
Def. Si $T: \mathbb{N} \to \mathbb{R}$,

$$\text{TIME}(T(n)) = \{ L \subseteq \Sigma^* \mid \exists \text{M.T. } M \text{ que decide } L \text{ y tiene tiempo de ejecución } O(T(n)) \}.$$

$$P = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$$

Ejemplo de ~~problema~~ lenguaje en la clase P:

$$\text{PATH} = \{ \langle G, v, w \rangle \mid G \text{ es un grafo dirigido, } v, w \text{ son vértices del grafo } G, \text{ y existe un camino dirigido que va desde } v \text{ hasta } w. \}$$



- Se podría resolver por fuerza bruta.

Enumerando todas las listas de vértices de hasta n vértices.

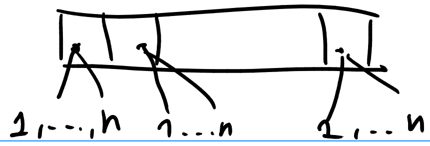
1, 3, 5

1, 3, 2, 1

1, 3, 5, 6, 7

1, 8, 4, 5

verificar si cada una de ellas es o no es un camino, y si el camino empieza en v y termina en w es porque $\langle G, v, w \rangle \in \text{PATH}$.



- Esto tendría complejidad exponencial, porque hay n^n listas de vértices posibles.

PATH $= \{ \langle G, v, w \rangle \mid$ G es un grafo dirigido, v, w son vértices del grafo G, y existe un camino dirigido que va desde v hasta w. $\}$

<u>Teorema</u>. PATH $\in$ P.

<u>Dem.</u> Usar Breadth-First-search (BFS).
Dado G, v, w.

```
visitados := {v}
cola := [v]
while not cola.vacía() {
    x := cola.desencolar()
    if x = w { return true }
    foreach arista x → y en el grafo G {
        if y ∉ visitados {
            cola.encolar(y)
            visitados := visitados ∪ {y}
        }
    }
}
return false
```

Itera a lo sumo n veces.

Itera a lo sumo n−1 veces.

Se puede hacer en tiempo polinomial

Si G tiene n vértices.

En total el costo del algoritmo es $O(n^3)$.

Por lo tanto PATH $\in$ TIME(n^3)
PATH $\in$ P.

• Recordemos que el máximo común divisor (gcd) entre dos números naturales $n, m \geq 0$ es el entero d más grande que divide a ambos n y m.

• Ej.

• $\gcd(\underset{2\cdot 2}{4}, \underset{2\cdot 5}{10}) = 2$

• $\gcd(\underset{3\cdot 5}{15}, \underset{2\cdot 7}{14}) = 1$

greatest common divisor

• Dos números n, m son coprimos si $\gcd(n, m) = 1$.

Def.

$$\text{RELPRIME} = \{ \langle n, m \rangle \mid \gcd(n, m) = 1 \}.$$

Ej. $\langle 10, 2 \rangle \notin \text{RELPRIME}$

$\langle 15, 14 \rangle \in \text{RELPRIME}$.

Observemos que podríamos resolver este problema por fuerza bruta.

15, 14 1 ✓ 2 × 3 × 4 × 5 × ... 14 ×

Y esto tendría complejidad exponencial, porque dados n, m el tiempo sería $O(\min(\underbrace{n}_{2^p}, \underbrace{m}_{2^q}))$.

n: p bits, m: q bits

$p \approx \log_2 n$ $q \approx \log_2 m$

Def. $\text{RELPRIME} = \{ \langle n, m \rangle \mid \gcd(n,m) = 1 \}$.

Teorema. $\text{RELPRIME} \in P$.

Dem. Algoritmo de Euclides para calcular gcd.

```
function gcd(n, m) {
    if m = 0 {
        return n
    } else {
        return gcd(m, n mod m)
    }
}
```

$0 \leq n \bmod m < m$

- Se puede ver fácilmente que si d divide a n y a m también divide a n y a $(n \bmod m)$ y viceversa.

- Para ver que tiene complejidad polinomial,

$\gcd(n,m) \longrightarrow \gcd(m, n \bmod m)$

- si $n < m$, $\gcd(m, n \bmod m) = \gcd(m, n)$.
- si $n = m$, $\gcd(m, \underbrace{n \bmod m}_{0}) = \gcd(m, 0) = m$.
- si $n > m$, hay dos casos:
  - (1) si $\frac{n}{2} \geq m$, $n \bmod m < m \leq \frac{n}{2}$.
  - (2) si $\frac{n}{2} < m$, $n \bmod m = n - m < n - \frac{n}{2} = \frac{n}{2}$.

$\frac{n}{2} < \overbrace{m < n < 2m}$

En resumen:

$$\gcd(n,m) \longrightarrow \gcd(m, n') \quad \underbrace{}_{n' = n \bmod m < \frac{n}{2}}$$

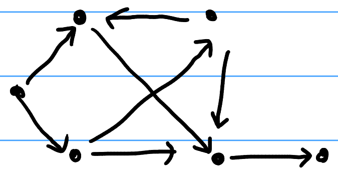
$$\longrightarrow \gcd(n', m') \quad \underbrace{}_{m' = m \bmod n' < \frac{m}{2}}$$

Como mucho, el tiempo de ejecución de $\gcd(\underbrace{n}_{p \text{ bits}}, \underbrace{m}_{q \text{ bits}})$ va a ser $O(\log_2 n + \log_2 m)$

$= O(p+q)$

Def. En un grafo dirigido G,
Un camino Hamiltoniano es
un camino dirigido que pasa
por todos los vértices de G exactamente una vez.

G :



Def.

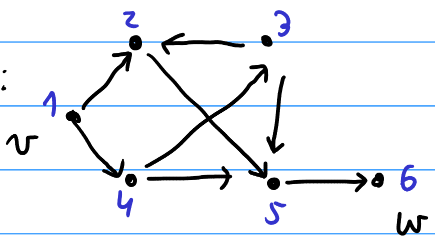
$$\text{HAMPATH} = \{ \langle G, v, w \rangle \mid G \text{ es un grafo dirigido y existe un camino Hamiltoniano que va de } v \text{ a } w \}.$$

- Se sabe cómo decidir este lenguaje en tiempo exponencial, haciendo fuerza bruta.

- No se sabe si es posible decidir HAMPATH en tiempo polinomial.

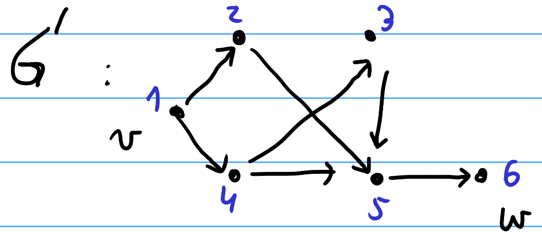
Sin embargo, sí es posible verificar una solución a HAMPATH en tiempo polinomial.

G :



$\langle G, v, w \rangle \in \text{HAMPATH}$ $\quad\quad$ $[1, 4, 3, 2, 5, 6]$

La verificación se puede hacer en tiempo polinomial.



$\langle G', 1, 6 \rangle \in \overline{HAMPATH}$

No es obvio que $\overline{HAMPATH}$ se pueda verificar en tiempo polinomial.

(No se sabe si es posible).

Otro ejemplo: "Primes is in P"

$$COMPUESTOS = \{ \langle n \rangle \mid n \in \mathbb{N}, \exists p, q \in \mathbb{N}, p, q > 1, n = p \cdot q \}$$

Este lenguaje se puede verificar en tiempo polinomial.

Si ya sé que $\langle n \rangle \in COMPUESTOS$.

Proveyéndole a alguien el número n y un factor p de n, puede verificar en tiempo polinomial que $\langle n \rangle \in COMPUESTOS$.

$\langle n \rangle \in \overline{COMPUESTOS}$

Def. Un <u>verificador</u> para un lenguaje $L \subseteq \Sigma^*$

es una M.T. V tal que:

$$L = \{ w \mid \exists c \in \Sigma^*. \, V(\langle w, c \rangle) = \text{acepta} \}$$

- Medimos el tiempo de ejecución de un verificador V en función del tamaño de w, ignorando el certificado.

- Un verificador V es de tiempo polinomial si su tiempo de ejecución es polinomial.

- Un lenguaje $L \subseteq \Sigma^*$ es polinomialmente verificable si tiene un verificador de tiempo polinomial.

<u>Ejemplo</u>.

HAMPATH es polinomialmente verificable.
Porque si existe un camino Hamiltoniano de v a w en un grafo G, la lista de vértices por los que pasa el camino se puede usar como certificado.

$$V(\langle G, v, w \rangle, \underbrace{\langle x_1, x_2, \ldots, x_n \rangle}_{\text{Certificado}})$$

Ver. si $x_1 = v$, $x_n = w$,

- Si no hay repetidos en $x_1, \ldots, x_n$.
- Si todos los vértices de G están en $x_1, \ldots, x_n$.
- Si todos los elementos de $x_1, \ldots, x_n$ son vértices de G.
- Si todo eso se cumple, aceptar.

  Si no, rechazar.

Ejemplo. Compuestos es polinomialmente verificable.

$V(\langle n \rangle, \langle p \rangle)$:

- Ver si p divide a n.
- Si lo divide, aceptar.
- Si no, rechazar.

Def. $NP = \{ L \subseteq \Sigma^* \mid L$ es polinomialmente verificable $\}$

HAMPATH $\in NP$ PATH $\in P \implies$ PATH $\in NP$

COMPUESTOS $\in NP$. RELPRIME $\in P \implies$ RELPRIME $\in NP$.

Obs. Si un lenguaje está en P, también está en NP.

$(P \subseteq NP)$.

Dem. Sup. que $L \in P$.

Es decir, existe una M.T. M que decide L.

podemos construir un verificador polinomial para L:

$V(w, c)$:

- Ignorar el certificado.
- Simula $M(w)$.

Es de tiempo polinomial y verifica el lenguaje L.

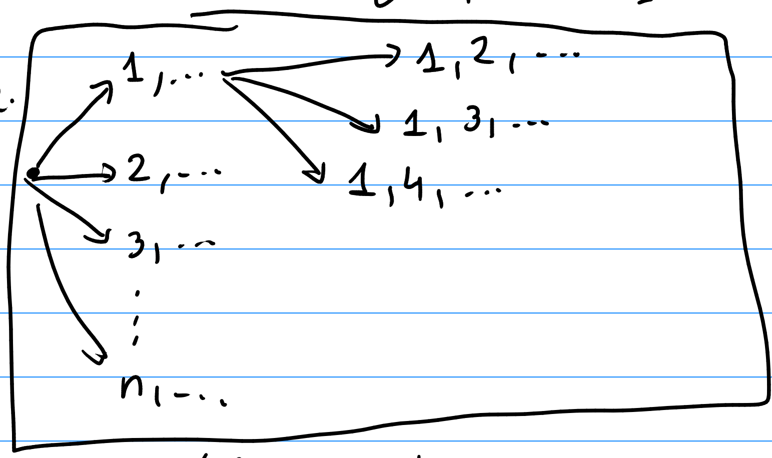
Ejemplo.

- El lenguaje HAMPATH se puede decidir con una M.T.N. en tiempo polinomial.

- Construimos una M.T.N. N para decidir HAMPATH:

1) Elegimos una lista de n vértices $[x_1, \dots, x_n]$ no determinísticamente.

n pasos no determinísticos.



Árbol de altura n

2) Chequeamos si la lista es un camino hamiltoniano. Si lo es, aceptar. Si no, rechazar.

Polinomial

Esta M.T.N. N decide Hampath en tiempo polinomial no determinístico.

Teorema. Un lenguaje $L \subseteq \Sigma^*$
está en NP
si y sólo si
existe una M.T.N. N que decide L en tiempo polinomial.

Dem.

($\Rightarrow$) Sup. que $L \in NP$, es decir, existe un verificador polinomial V para L.

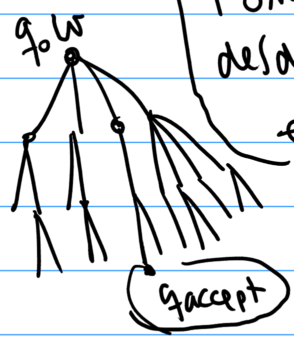
Sup. que V termina en n^k pasos.

Construimos una M.T.N. que decide L en tiempo polinomial así:

$N(w)$:

- Elegir no determinísticamente un certificado c de tamaño a lo sumo n^k.
- Simular $V(w,c)$.

($\Leftarrow$) Sup. que existe una M.T.N. N que decide L en tiempo polinomial.



Tomamos como certificado el camino que conduce desde la raíz hasta la configuración que esté en el estado q_{accept}.

Construimos el verificador V así:

$V(w,c)$:

- Simular $N(w)$. En cada transición, elegir la opción indicada por el certificado c. Polinomial
- Si se llegó a q_{accept}, aceptar. Si no, rechazar.

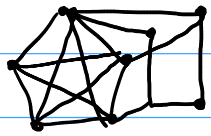
Def. alternativa de NP:

Si $T : \mathbb{N} \to \mathbb{R}$,

$$NTIME(T(n)) = \{ L \subseteq \Sigma^* \mid \text{existe una M.T.N. } N \text{ que decide } L \text{ en tiempo } O(T(n)) \}.$$

$$NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$$

Def. En un grafo no dirigido, una clique es un subgrafo tal que para todo par de vértices en la clique, hay una arista entre ellos.



Def.

$$\text{CLIQUE} = \{\langle G, k\rangle \mid G \text{ tiene una } \underbrace{\text{clique de tamaño } k}_{k\text{-clique.}}\}$$

Teorema. CLIQUE $\in NP$.

Dem. Veamos que existe un verificador polinomial para CLIQUE.

$V(\langle G, k\rangle, c)$:

- Chequear que c represente una lista de k vértices del grafo G tales que todo par de vértices en la lista son adyacentes en el grafo G.
  Se puede hacer en tiempo polinomial.
- Si esto vale, aceptar.
  Si no, rechazar.

¿CLIQUE $\in P$? No se sabe.

¿$\overline{\text{CLIQUE}} \in NP$? No se sabe.

Otro ejemplo: SUBSET-SUM.

Def.

$$\text{SUBSET-SUM} = \{ \langle \underbrace{X}_{\text{Cjto. de números naturales}}, \underbrace{n}_{\text{número natural}} \rangle \mid X = \{x_1, \dots, x_p\},\ \text{existe } \{y_1, \dots, y_q\} \subseteq X \text{ tal } \sum_{i=1}^{q} y_i = n \}.$$

Ejemplo: $\langle \{1, \underline{3}, \underline{7}, \underline{10}, 18\}, 20 \rangle \in$ SUBSET-SUM

$3 + 7 + 10 = 20$

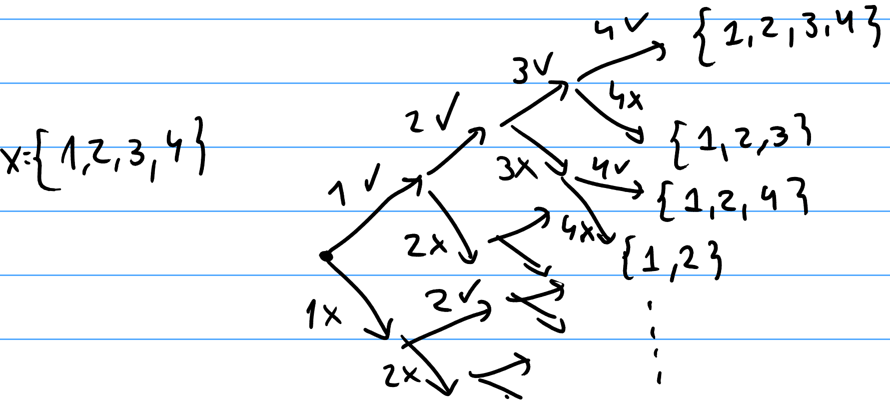
$\langle \{2, 8, 10\}, 11 \rangle \notin$ SUBSET-SUM

Teorema. SUBSET-SUM $\in$ NP.

Dem. Construyamos una M.T.N. N que decida SUBSET-SUM en tiempo polinomial.

$N(\langle X, n \rangle)$:

- Elegir no determinísticamente un subconjunto $Y \subseteq X$.

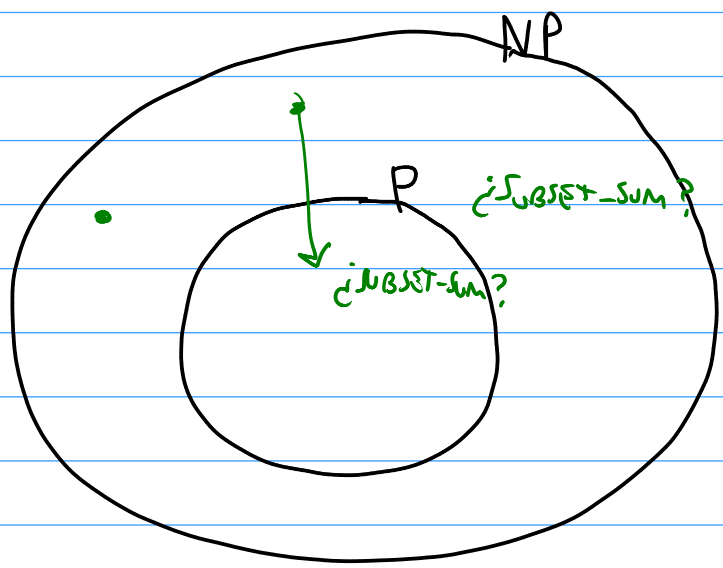


- Chequear si la suma de los elementos de Y es n.
- Si lo es, aceptar. Si no, rechazar.

¿SUBSET-SUM $\in$ P?   No se sabe.

¿$\overline{\text{SUBSET-SUM}}$ $\in$ NP?   No se sabe.

$P \subseteq NP$ ✓



¿$NP \overset{?}{\subseteq} P$?

$P \overset{?}{=} NP$