Trabajo Práctico Nro 7 - La jerarquía espacio-temporal y Misceláneos

Ejercicio 1

Responder breve y claramente:

1. ¿Por qué en la complejidad espacial se utilizan MT con una cinta de entrada de sólo lectura?

En la **complejidad espacial** se utilizan MT con una **cinta de entrada de sólo lectura** para almacenar una entrada w de tamaño n, y no para trabajar debido a que sino, la MT ya ocuparía un espacio de al menos $O(n)$ celdas. Si la **cinta de entrada** fuera también una **cinta de trabajo** y contara para la medición del espacio ocupado $S(n)$, todas las máquinas de Turing que procesaran una entrada tendrían una complejidad espacial mínima lineal (proporcional al tamaño de la entrada), lo que impediría distinguir problemas que requieren menos espacio de cómputo que el necesario para simplemente leer o almacenar la entrada.

2. ¿Por qué si una MT tarda tiempo $poly(n)$ entonces ocupa espacio $poly(n)$, y si ocupa espacio $poly(n)$ puede llegar a tardar tiempo $exp(n)$?

¿Por qué si una MT tarda tiempo $poly(n)$ entonces ocupa espacio $poly(n)$?

Una MT que se ejecuta en tiempo $poly(n)$, es decir, hace $T(n)$ pasos, en el peor caso, *solo puede acceder y, por lo tanto, ocupar a lo sumo T(n) celdas en sus cintas de trabajo.* Esto se debe a que, en cada paso, el cabezal de una cinta solo puede moverse una posición hacia la izquierda o la derecha, o quedarse quieto. Por lo tanto, después de $T(n)$ pasos, el cabezal de cada cinta no puede estar a más de $T(n)$ posiciones de su posición inicial. *El espacio total ocupado en todas las cintas de trabajo es, como máximo, la suma de las celdas visitadas en cada cinta, lo cual está acotado por el número total de pasos.*

¿Por qué si una MT ocupa espacio $poly(n)$ puede llegar a tardar tiempo $exp(n)$?

Una MT que ocupa espacio $S(n)$ puede tardar mucho más tiempo que $S(n)$. Esto se debe a que *la máquina puede realizar muchos pasos dentro del espacio acotado $S(n)$ sin repetir una configuración.* El número máximo de pasos que una MT puede ejecutar sin entrar en un bucle infinito está limitado por la cantidad total de configuraciones distintas que puede alcanzar en un espacio $S(n)$. Viendo el ejemplo que se presenta en la teoría, si tenemos una MT M con:

- Una cinta de entrada de sólo lectura.
- Una cinta de trabajo.

- $|Q|$ estados.
- $|\Gamma|$ símbolos.
  Podemos calcular la cantidad máxima de pasos, si no loopea como:
  $(n+2) \cdot S(n) \cdot |Q| \cdot |\Gamma|^{S(n)} = c^{S(n)} = exp(S(n))\ pasos$. *Si el espacio $S(n)$ que ocupa la MT es polinomial $poly(n)$, el tiempo máximo de ejecución puede ser exponencial en n $exp(poly(n))$.*

3. ¿Por qué los lenguajes de la clase $LOGSPACE$ son tratables?

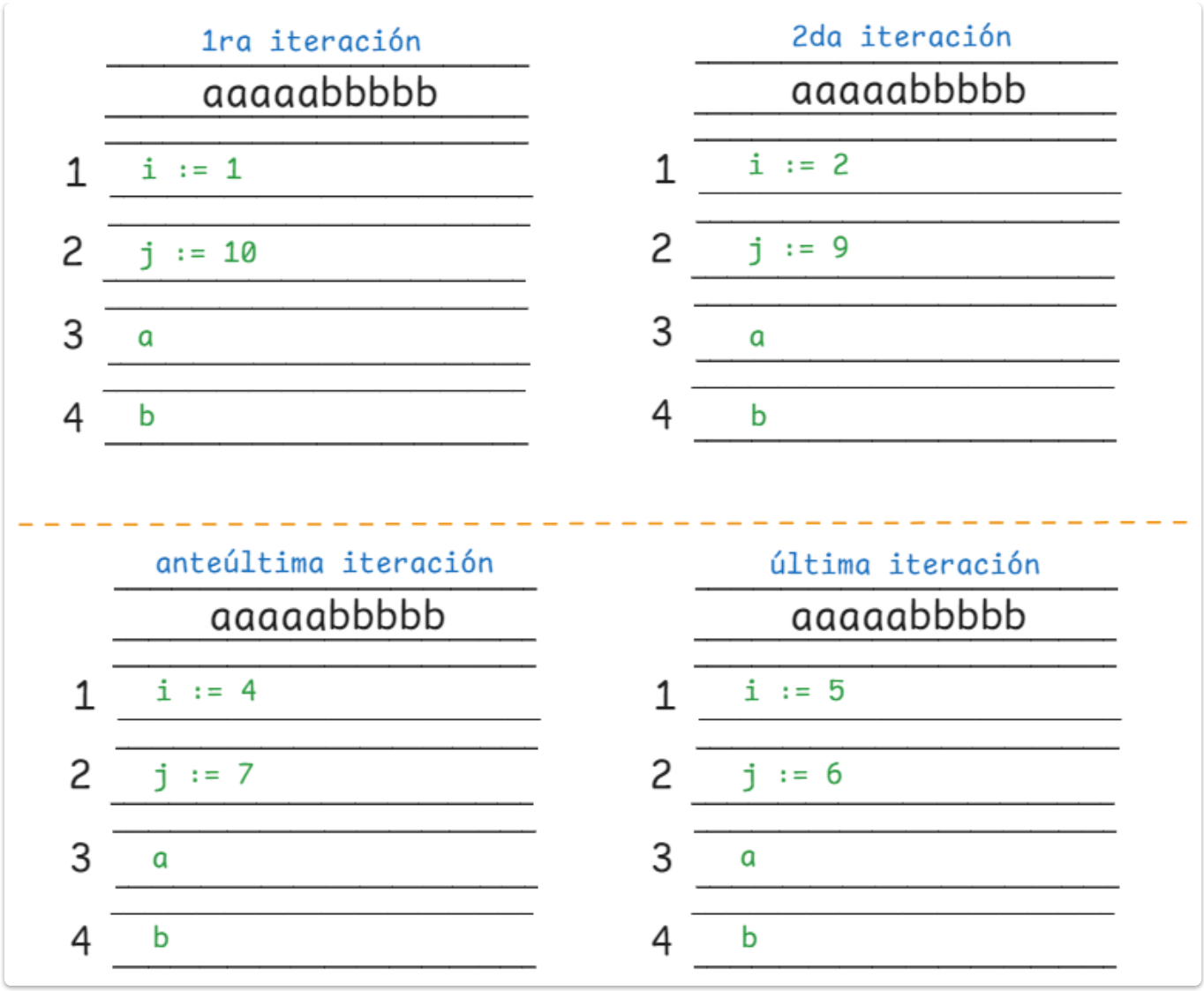
La clase $LOGSPACE$ agrupa los lenguajes que pueden ser decididos en espacio $O(log_2(n))$. Si una MT decide un lenguaje en espacio $S(n) = O(log_2(n))$, entonces el tiempo máximo que puede tardar está acotado por: $T(n) = O(c^{S(n)})$. Si $S(n) = log_2(n)$, entonces: $T(n) = O(c^{log_2(n)})$. Usando la propiedad $a^{log_x(b)} = b^{log_x(a)}$, **esto se convierte en:** $T(n) = O(n^{log_2(c)})$.

Ejercicio 2

Describir la idea general de una MT M que decida el lenguaje $L = \{a^n b^n \mid n \geq 1\}$ en espacio logarítmico. *Ayuda: basarse en el ejemplo mostrado en clase.*

Una máquina de Turing M que decide el lenguaje $L = \{a^n b^n \mid n \geq 1\}$ en espacio logarítmico podría actuar de la siguiente forma:

1. Hace $i := 1$ en la cinta 1.
2. Hace $j := n$ en la cinta 2. Si j es impar, rechaza.
3. Copia el símbolo i de w en la cinta 3.
4. Copia el símbolo j de w en la cinta 4.
5. Si $i = (j-1)$: Si los símbolos son distintos, acepta, si no, rechaza.
   Si $i \neq j$: Si los símbolos son iguales, rechaza.
6. Hace $i := i + 1$ en la cinta 1.
7. Hace $j := j - 1$ en la cinta 2.
8. Vuelve al paso 3.
   La MT M ocupa el espacio de los contadores i y j, que en binario miden $O(log_2(n))$, más 2 celdas para alojar cada vez a los símbolos comparados (espacio constante). Por lo tanto, $L \in SPACE(log_2(n))$.



Ejercicio 3

Describir la idea general de una MT M que decida el lenguaje SAT en espacio polinomial.
Ayuda: la generación y la evaluación de una asignación de valores de verdad se pueden efectuar en tiempo polinomial.

Una máquina de Turing M que decida el lenguaje SAT puede ocupar espacio polinomial si usamos una estrategia que reúse el espacio de las cintas y no barra las 2^n $(\text{siendo n la cantidad de variables de } \phi)$ posibles asignaciones.

Si ϕ tiene n variables, existen 2^n posibles asignaciones, la máquina en vez de almacenarlas todas las puede simular una por una, llevando un contador para saber cuál es la asignación que estoy procesando:

1. Establecemos un contador en 1 en una cinta.

2. *Generamos* una asignación A para las n variables de ϕ en otra cinta, esto se puede efectuar en *tiempo polinomial* y, por lo tanto, ocupando *espacio polinomial*.
3. *Verificamos* si la fórmula ϕ es verdadera para la asignación A, esto se puede efectuar en *tiempo polinomial* y, por lo tanto, ocupando *espacio polinomial*.
   - Si es satisfactoria, la máquina se detiene y acepta la fórmula ϕ.
   - Si no es satisfactoria y el contador no es 2^n, se aumenta el contador y se reutiliza el espacio ocupado en el punto 2, volviendo a ejecutar desde ese mismo punto.
   - Si no es satisfactoria y el contador es 2^n, la máquina se detiene y rechaza.

Si bien exploramos una cantidad exponencial de asignaciones posibles, podemos hacer que la MT M ocupe espacio polinomial. Usa un contador que en binario mide $O(log_2(2^n))$ que, por propiedades de los logaritmos, termina siendo $O(n)$, más una cinta que mide $O(|\phi|)$. Por lo tanto, $SAT \in PSPACE$.

Ejercicio 4

Justificar por qué el lenguaje $QSAT$ no pertenecería a P ni a NP. *Ayuda: ¿Qué forma tienen los certificados asociados a $QSAT$?*

¿Por qué no pertenecería a P?

La clase P contiene los lenguajes que pueden *ser decididos por una máquina de Turing determinística en tiempo polinomial*. En la teoría se nos muestra que $QSAT$ es $PSPACE - completo$ (dentro de los lenguajes decidibles en espacio polinomial, está entre los más difíciles), por lo tanto, cualquier lenguaje en $PSPACE$ puede ser reducido a $QSAT$ en tiempo polinomial. *Si nos basamos en la conjetura de que $P \subset PSPACE$*, si $QSAT$, que es $PSPACE - completo$, perteneciera a P, esto implicaría que $P = PSAPCE$, algo que ya *nos da un indicio de que no pertenecería a P.* Otra cosa que nos hace ver que *$QSAT$ no pertenecería a P* es el hecho de que *determinar este lenguaje conlleva explorar todas las posibles alternativas para una fórmula ϕ por "fuerza bruta".* Esto es exponencial en el peor caso.

¿Por qué no pertenecería a NP?

Para que $QSAT$ pertenezca a NP debe tener un *certificado sucinto verificable por una máquina de Turing en tiempo polinomial* pero en el caso de este lenguaje, un certificado para verificar si una fórmula totalmente cuantificada es verdadera *no es sucinto*. Por ejemplo, para una fórmula $\phi = \forall x \exists y \exists z : x \wedge y \wedge z$, un certificado sería un *árbol de valores de verdad*:

- Este árbol debe detallar, para cada posible valor del cuantificador universal ($\forall$), el valor que elegiría el cuantificador existencial ($\exists$) para hacer la fórmula verdadera. *El tamaño de este*

árbol puede ser exponencial con respecto al tamaño de la fórmula, por lo tato, *no es sucinto*.

Ejercicio 5

Probar que $NP \subseteq PSPACE$. *Ayuda: Si L pertenece a NP, entonces existe una MT M_1 capaz de verificar en tiempo $poly(|w|)$ si una cadena w pertenece a L, con la ayuda de un certificado x de tamaño $poly(|w|)$. De esta manera, existe también una MT M_2 que decide L en espacio $poly(|w|)$, sin la ayuda de ningún certificado.*

Para probar que $NP \subseteq PSPACE$, debemos demostrar que **cualquier lenguaje verificable en tiempo polinomial con certificado sucinto también puede decidirse usando solo espacio polinomial**, aunque no tengamos el certificado.

SI tenemos un lenguaje L que pertenece a NP, entonces existe:

- Una máquina de Turing M que lo verifica en *tiempo polinomial*, por lo tanto, *espacio polinomial*.
- Un *certificado sucinto* x que tiene *tamaño polinomial con respecto a la entrada* w, $|x| \leq poly(|w|)$.

Nosotros tenemos que construir una máquina de Turing M' que decida si $w \in L$ sin conocer el certificado x. Esto lo podemos hacer **recorriendo todos los certificados posibles de longitud $\leq poly(|w|)$.**

- La máquina M' irá generando todos los posibles certificados de longitud $\leq poly(|w|)$, que podrían llegar a ser $k^{poly(|w|)}$, *probándolos uno por uno, sin almacenarlos todos a la vez* (parecido a lo que hicimos con SAT antes).
- *Para cada posible x generado vamos a simular $M(w, x)$*, si encontramos un x tal que $M(w, x)$ acepta, entonces *aceptamos*. Si terminamos de probar todos los x posibles y ninguno sirve, entonces *rechazamos*.

M' ocupa una cinta con un contador para saber por cuál certificado vamos, ya vimos que un contador binario ocuparía tamaño $O(log_2(k^{poly(|w|)}))$ que nos terminaría quedando por propiedad $O(poly(|w|) \cdot log_2(k)) \rightarrow O(poly(|w|)$, más una cinta para almacenar los certificados generados que, como máximo, ocupan $O(|w|)$. Entonces, **el total de espacio es polinomial**, aunque el número de posibles certificados sea exponencial, **lo que demuestra que $NP \subseteq PSPACE$**.

Ejercicio 6

Supongamos que existe una MT M de tiempo polinomial que, dado un grafo G, devuelve un circuito de Hamilton de G si existe o responde no si no existe. Describir la idea general de una MT $M´$ que, utilizando M, decida en tiempo polinomial si un grafo G tiene un circuito de Hamilton. *Ayuda: basarse en el ejemplo mostrado en clase con $FSAT$ y SAT.*

Para probar que la máquina de Turing M' **decide en tiempo polinomial** si un grafo G tiene un circuito de Hamilton podemos basarnos en la relación general entre un problema de búsqueda FP y su lenguaje asociado (problema de decisión) P, que nos dice que **si un problema de búsqueda FP puede resolverse en tiempo polinomial, también puede decidirse en tiempo polinomial el lenguaje P asociado.**

Nosotros tenemos una máquina de Turing determinística M que resuelve el **problema de búsqueda** de encontrar un CH en un grafo G en **tiempo polinomial**. Ahora queremos construir una máquina de Turing determinística M' que **decida** si G tiene un CH en **tiempo polinomial**. La estrategia para lograr esto es que **la máquina M' use como subrutina a la máquina M.**

La máquina M' para un grafo de entrada G:

1. Ejecuta $M(G)$.
2. Si $M(G)$ devuelve un *circuito de Hamilton*, entonces *aceptar* (la respuesta es *sí*).
   Si $M(G)$ responde *"no"*, entonces *rechazar* (la respuesta es *no*).
   Como asumimos que M corre en **tiempo polinomial**, y M' simplemente **llama a M** y analiza la salida, M' también corre en **tiempo polinomial**.

Esto prueba que si el **problema de búsqueda** (encontrar un circuito de Hamilton) está en FP, entonces el **problema de decisión** (saber si existe) está en P.

Ejercicio 7

Sea la MTP M que definimos en clase para decidir probabilísticamente el lenguaje $MAYSAT = \{\varphi \mid \varphi \text{ es una fórmula booleana satisfactible por más de la mitad de las posibles asignacion}$. Hemos indicado que para toda φ, si $\varphi \in MAYSAT$ entonces M la acepta con probabilidad $> \frac{1}{2}$, y si $\varphi \notin MAYSAT$ entonces M la rechaza con probabilidad $\geq \frac{1}{2}$. Precisando más la primera probabilidad: asumiendo que M tarda $p(n)$, si $\varphi \in MAYSAT$ entonces M la acepta con probabilidad $\geq \frac{1}{2} + (\frac{1}{2})^{p(n)}$. Explicar por qué. *Ayuda: en tiempo $p(n)$, M puede producir $2^{p(n)}$ computaciones posibles, y entonces, ¿Cuántas son de aceptación como mínimo si $\varphi \in MAYSAT$?*

Si tenemos una MTP que se ejecuta en tiempo polinomial $p(n)$ para una entrada de tamaño n, esto significa que cualquier computación (cualquier camino en el árbol binario de computaciones) tiene una longitud de a lo sumo $p(n)$ pasos, a su vez, la MTP puede producir $2^{p(n)}$ computaciones, esto se debe a que en cada uno de los $p(n)$ pasos, la máquina puede tomar una de dos decisiones aleatorias, generando $2 \times 2 \times \cdots \times 2$ ($p(n)$ veces) caminos distintos de longitud $p(n)$.

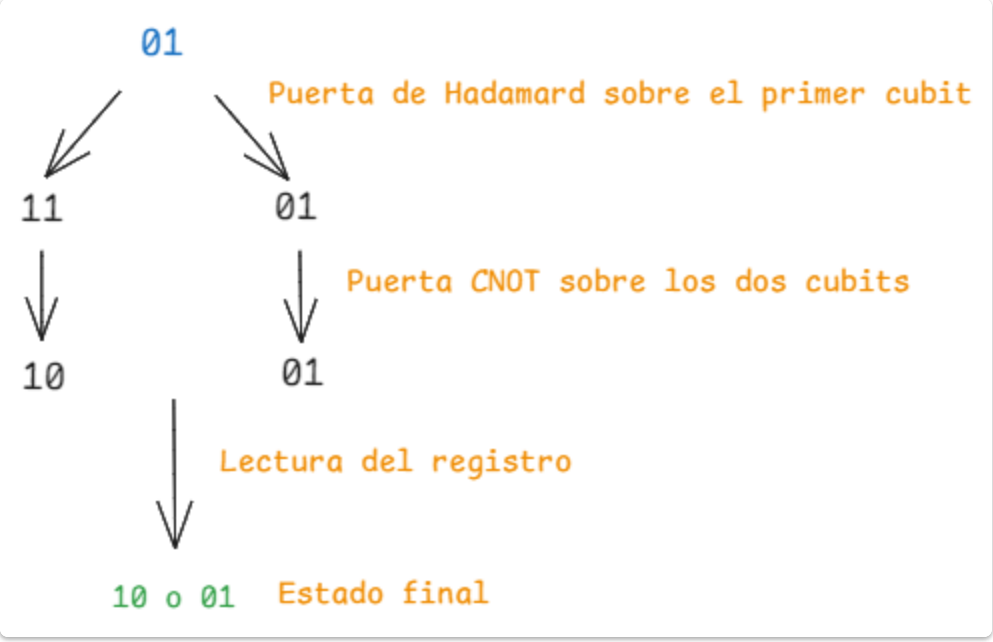
La probabilidad de que una MTP M acepte una cadena w es la relación entre la cantidad de computaciones en las que M acepta w y la cantidad total de computaciones. En este caso, la cantidad total considerada es $2^{p(n)}$.

Si $\varphi \in MAYSAT$ entonces la probabilidad de que M acepte φ siguiendo el razonamiento explicado sería: $\frac{2^{p(n)-1}+1}{2^{p(n)}} = \frac{2^{p(n)-1}}{2^{p(n)}} + \frac{1}{2^{p(n)}} = 2^{-1} + (\frac{1}{2})^{p(n)} = \frac{1}{2} + (\frac{1}{2})^{p(n)}$

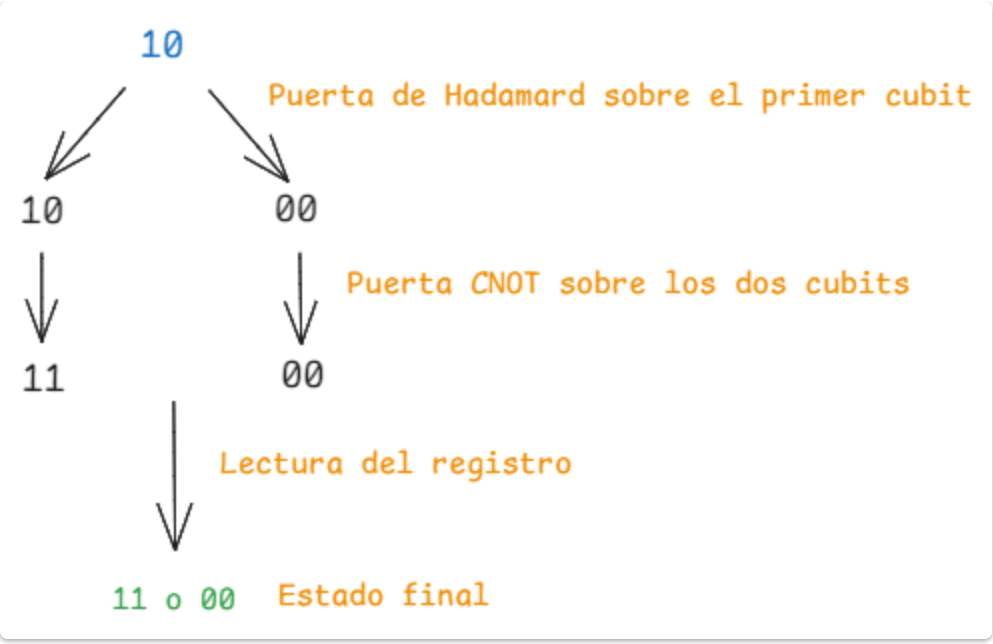
Ejercicio 8

Considerando el ejemplo de computación cuántica mostrado en clase, indicar los resultados posibles cuando en lugar de arrancar con el estado inicial 00, la computación arranca con:

1. El estado inicial 01.



2. El estado inicial 10.



3. El estado inicial 11.