Universidad Nacional Autónoma de México
Facultad de Ciencias

Complejidad Computacional

Tarea 2

Bonilla Reyes Dafne - 319089660
García Ponce José Camilo - 319210536
Juárez Ubaldo Juan Aurelio - 421095568
Moreno Castro Fernanda - 319128727

Ejercicio 1

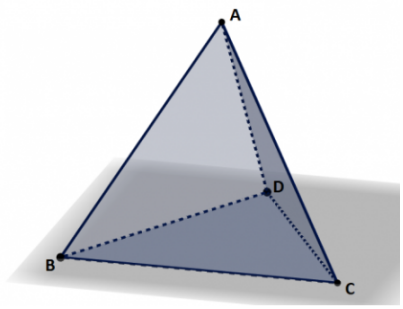
Considera los sólidos platónicos, y las gráficas consistentes de sus esquinas y aristas: tetraedro, cubo, octaedro e icosaedro. Para cada una de las gráficas anteriores, responde lo siguiente:

- ¿Tiene un ciclo Euleriano?
- ¿Tiene un ciclo Hamiltoniano?
- ¿Cuál es el clan más grande?
- ¿Cuál es la coloración mínima?

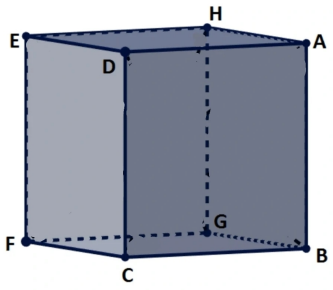
En cada caso justifica tu respuesta.

Se ejemplifica con las imágenes

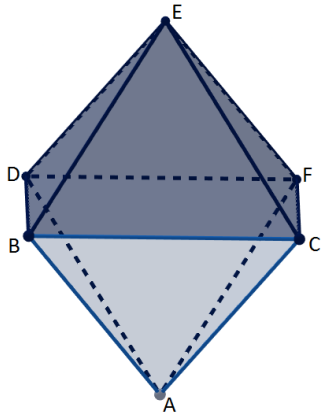
- **Tetraedro** : Un tetraedro es un grafo completo K4 con 4 vértices y 6 aristas.
  Ciclo Hamiltoniano: Se puede recorrer todos los vértices en un solo ciclo. $BACDB$
  Ciclo Euleriano: Cada vértice tiene grado 3 (impar), por lo que no tiene un ciclo Euleriano.
  Coloración: Es una 4 coloración, ya que cada vértice está conectado a todos los demás. $f(A) = 1, f(B) = 2, f(C) = 3, f(D) = 4$, no podemos tener una 3 coloración ya que son 4 vértices y todos están conectados entre si, entonces 3 colores no es suficiente.
  Clan: El tetraedro es un clan de tamaño 4, y es el máximo ya que no hay 5 vértices.



- **Cubo** : Un cubo es un grafo cúbico con 8 vértices y 12 aristas.
  Ciclo Hamiltoniano: Se puede recorrer todos los vértices en un solo ciclo. $FEHADCBGF$
  Ciclo Euleriano: Cada vértice tiene grado 3 (impar), por lo que no tiene un ciclo Euleriano.
  Coloración: Tiene una coloración mínima de 2. $f(A) = 1, f(H) = 2, f(B) = 2, f(G) = 1, f(D) = 2, f(C) = 1, f(E) = 1, f(F)$, claramente no puede tener una 1 coloración debido a que no es la gráfica trivial
  Clan: Tiene clanes de tamaño 2. AB, no tiene un clan de tamaño 3, debido a que podemos notar que existan 3 vértices adyacentes entre ellos (imagen de gráfica)

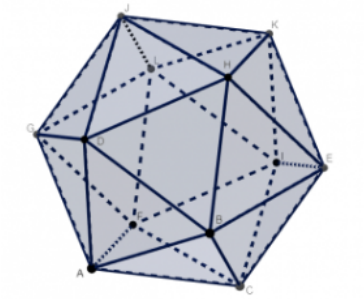


- **Octaedro** : Un octaedro es un grafo cúbico con 6 vértices y 12 aristas.
  Ciclo Hamiltoniano: Se puede recorrer todos los vértices en un solo ciclo. $DEFCABD$
  Ciclo Euleriano: Cada vértice tiene grado 4 (par), por lo que sí tiene un ciclo Euleriano. $DEFCEBCAFDABD$
  Coloración: Tiene una 3 coloración $f(A) = 1, f(E) = 1, f(D) = 2, f(C) = 2, f(B) = 3, f(F) = 1$, no puede tener una 2 coloración debido a que recordemos que .[el] numero de coloración de una gráfica es al menos su numero de clan"(Fuente en las referencias)
  Clan: Tiene clanes de tamaño 3. BEC, no tiene un clan de tamaño 4, debido a que podemos notar que existan 4 vértices adyacentes entre ellos (imagen de gráfica)

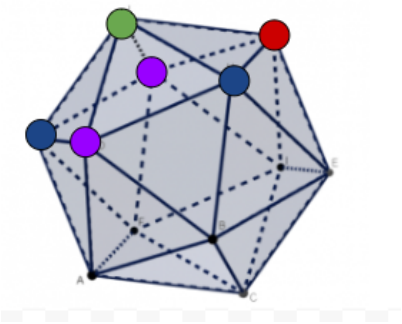


- **Icosaedro** : Un icosaedro es un grafo regular con 12 vértices y 30 aristas.
  Ciclo Hamiltoniano: Se puede recorrer todos los vértices en un solo ciclo. $JKIEBCADLFGJ$
  Ciclo Euleriano: Cada vértice tiene grado 5 (impar), por lo que no tiene un ciclo Euleriano.
  Coloración: $f(J) = 1, f(G) = 2, f(D) = 3, f(I) = 3, f(K) = 2, f(H) = 4, f(A) = 4, f(F) = 1, f(L) = 4, f(E) = 1, f(B) = 2, f(C) = 3$ es una 4 coloración, ya que como tenemos ciclos de longitud 5, entonces necesitamos 3 colores para colorearlos y como a cada vértice del ciclo son adyacentes

a otro vértice entonces necesitamos un color extra y por eso la 4 coloración
Clan: Tiene clanes de tamaño 3. ABD, no tiene un clan de tamaño 4, debido a que podemos notar que existan 4 vértices adyacentes entre ellos (imagen de gráfica)



El por qué es coloración 4 es porque como podemos ver en la siguiente imagen, vemos que el vértice rojo no puede ser color verde, ni azul, ni rosa. El vértice verde no puede ser azul y tampoco rosa.



Ejercicio 2

Responde lo siguiente:

- ¿Cuáles son las diferencias entre Problema de Decisión, Problema de Búsqueda y Problema de Optimización?

  - **Problema de Decisión:**
    - La pregunta central es si existe o no una solución que cumpla ciertos criterios.
    - La respuesta es binaria: "sí" o "no".
    - No se busca una solución específica, solo se verifica si alguna solución cumple los criterios dados.
  - **Problema de Búsqueda:**
    - Se trata de encontrar una solución específica dentro de un conjunto de posibles soluciones que cumplan ciertos criterios.
    - A diferencia del problema de decisión, no es suficiente responder "sí" o "no"; se requiere identificar la solución concreta que resuelve el problema.
  - **Problema de Optimización:**
    - Se enfoca en encontrar la mejor solución posible, que maximice o minimice un valor, respetando ciertas restricciones.
    - A diferencia de los problemas de decisión y búsqueda, aquí no solo se busca una solución válida, sino la mejor solución posible para el problema dado.

- Da un ejemplo de cada uno de los tres tipos de problemas anteriores, presentando el problema en su versión canónica. Muestra un ejemplar concreto de cada uno de los problemas.

  - **Decisión**
    Suma de conjuntos
    Ejemplar: un conjunto c de números y un entero x.
    Pregunta: ¿existe un subconjunto de c que sumados todos sus elementos den x?

    Ejemplar
    $c = \{10, 30, 70, 5, 25, 45, 90, 65, 100, 150\}$
    $x = 60$

  - **Búsqueda**
    Suma de conjuntos
    Ejemplar: un conjunto c de números y un entero x.
    Pregunta: ¿cuál es el subconjunto de c que sumados den x?

    Ejemplar
    $c = \{10, 30, 70, 5, 25, 45, 90, 65, 100, 150\}$
    $x = 60$

  - **Optimización**
    Suma de conjuntos
    Ejemplar: un conjunto c de números y un entero x.

Pregunta: ¿cuál es el subconjunto de c de menor tamaño, tal que sumados den x?

Ejemplar
$c = \{10, 30, 70, 5, 25, 45, 90, 65, 100, 150\}$
$x = 60$

- Menciona un par de ejemplos de problemas de decisión, cuyos ejemplares no involucren gráficas. Los ejemplos deben ser diferentes a los mencionados en clase.

  - **Suma de conjuntos (subset sum)**
    Ejemplar: un conjunto c de números y un entero x.
    Pregunta: ¿existe un subconjunto de c, tal que sumados den x?

  - **Sudoku**
    Ejemplar: una tabla parcialmente llena con números.
    Pregunta: ¿existe una forma de completar la tabla con números, tal que el tablero final sea una solución válida del sudoku ?

Los ejemplos deben ser diferentes a los mencionados en clase.

Ejercicio 3

Considera los siguientes dos problemas bastante diferentes con respecto a **CICLOS HAMILTONIANOS**. Uno es el problema de decisión, la pregunta de si existe o no uno de tales ciclos. El otro es el problema de búsqueda, en el que queremos de hecho encontrar el ciclo. Supón que existe un oráculo que nos dice, por el precio de una moneda por pregunta, si una gráfica tiene o no un ciclo hamiltoniano.

- **Demuestra que haciendo una serie de preguntas al oráculo, podemos resolver el problema de búsqueda utilizando solo una cantidad de monedas que crece polinomialmente como una función del número de vértices.**

  Si es posible encontrar el ciclo Hamiltoniano, veamos como:

  Demostración.

  Sea una gráfica $G = (V, E)$ con n vértices, primero le preguntamos al oráculo si G tiene un ciclo Hamiltoniano, si no tiene entonces terminamos la búsqueda (ya que no hay uno), pero si tiene entonces seguimos buscando.

  Ahora, para cada arista a de G, quitamos a a de la gráfica obteniendo una nueva gráfica G' y le preguntamos al oráculo si G' tiene un ciclo Hamiltoniano. Si no tiene un ciclo entonces agregamos de vuelta la arista a a la gráfica (esto debido a que como G' no tiene un ciclo Hamiltoniano entonces sabemos que a pertenece a algún ciclo Hamiltoniano de G) y seguimos revisando con las demás aristas, pero si el oráculo dijo que G' si tiene un ciclo Hamiltoniano entonces quitamos de forma permanente a la arista a de la gráfica (esto debido a que en G' hay al menos un ciclo Hamiltoniano que no contiene a a) y seguimos revisando las aristas restantes de la gráfica.

  Cuando terminemos de revisar todos las aristas nos va a quedar un ciclo Hamiltoniano, ya que solo se quedaron las aristas necesarias para el ciclo (todo porque eliminamos a los otros mediante las preguntas al oráculo), y así resolviendo el problema de búsqueda.

  Ahora observemos cuantas preguntas tomo esto. Notemos que cada pregunta al oráculo cuesta una moneda, por lo tanto, contemos cuantas preguntas se realizaron:

  1. Al inicio, fue una pregunta para saber si G tenía o no un ciclo Hamiltoniano.
  2. Luego, se realizó una pregunta para cada arista de G.

  Recordemos que una gráfica con m vértices tiene a lo más $\frac{m(m-1)}{2}$ aristas, por lo tanto, G tiene a lo más $\frac{n(n-1)}{2}$ aristas y con estos sabemos que se hicieron, a lo más, $\frac{n(n-1)}{2}$ $+1$ preguntas, una pregunta al inicio y $\frac{n(n-1)}{2}$ el preguntar por los vértices (una pregunta por vértice).

  Notemos que $\frac{n(n-1)}{2} = \frac{n^2}{2} + \frac{n}{2}$, entonces tomo (a lo más) $\frac{n^2}{2} + \frac{n}{2} + 1$ preguntas y esto tiene complejidad $O(n^2)$, ya que $n^2 \geq \frac{n^2}{2}$, $n^2 \geq \frac{n}{2}$, $n^2 \geq 1$ para $n \geq 1$, por lo tanto $3n^2 \geq \frac{n^2}{2} + \frac{n}{2} + 1$, entonces es $O(n^2)$ y $O(n^2)$ es polinomial en función al número de vértices n. □

- Concluye que podemos resolver el problema de decisión en tiempo polinomial, si y solo si podemos resolver el problema de búsqueda también en tiempo polinomial.

  Supongamos que podemos resolver el problema de decisión en tiempo polinomial, ahora veamos que podemos resolver el problema de búsqueda.

  Entonces para encontrar el ciclo vamos a realizar las acciones vistas en el ejercicio de arriba, solo que en vez de preguntar al oráculo hacemos la rutina o pasos para resolver el problema de decisión (esto debido a que el oráculo resolvía el problema de decisión y nosotros hicimos preguntas

para resolver el problema de búsqueda). Pero falta ver en cuanto tiempo resolvemos el problema de búsqueda.

Sabemos que en el ejercicio anterior nos tomó $\frac{n^2}{2} + \frac{n}{2} + 1$ preguntas, pero ahora no hacemos preguntas, hacemos pasos que toman tiempo polinomial, entonces nos va a tomar $\frac{n^2}{2} + \frac{n}{2} + 1 *$ un polinomio.

Aquí, notemos que $\frac{n^2}{2} + \frac{n}{2} + 1$ es un polinomio, entonces $\frac{n^2}{2} + \frac{n}{2} + 1 *$ un polinomio nos da otro polinomio (multiplicar dos polinomios es otro polinomio), por lo tanto, nos toma tiempo polinomial resolver búsqueda.

Ahora supongamos que podemos resolver el problema de búsqueda en tiempo polinomial, ahora veamos que podemos resolver el problema de decisión. Entonces, para saber si existe o no un ciclo solo realizamos la rutina o pasos para resolver el problema de búsqueda, si no encontramos un ciclo entonces decimos que no hay un ciclo, pero si encontramos un ciclo entonces decimos que si hay un ciclo y como resolver búsqueda toma tiempo polinomial, entonces los pasos tomaron tiempo polinomial y resolver decisión también nos tomó tiempo polinomial.

Por lo notado, arriba concluimos que podemos resolver el problema de decisión en tiempo polinomial, si y solo si podemos resolver el problema de búsqueda también en tiempo polinomial.

Ejercicio 4

¿Cuáles de las siguientes afirmaciones son verdades y cuáles falsas? Justifica tu respuesta en cada caso.

Para comenzar, primero daremos las definiciones formales de O, Ω y Θ, que utilizaremos para resolver este ejercicio:

O

Una función $f(n)$ es $O(g(n))$ si existen constantes positivas c y n_0 tales que para todo $n \geq n_0$ se cumple que

$$0 \leq f(n) \leq c \cdot g(n)$$

Esto significa que $f(n)$ crece a lo sumo tan rápido como $g(n)$ para valores grandes de n.[?]

Ω

Una función $f(n)$ es $\Omega(g(n))$ si existen constantes positivas c y n_0 tales que para todo $n \geq n_0$ se cumple que

$$0 \leq c \cdot g(n) \leq f(n)$$

Esto significa que $f(n)$ crece al menos tan rápido como $g(n)$ para valores grandes de n.[?]

Θ

Una función $f(n)$ es $\Theta(g(n))$ si existen constantes positivas c_1, c_2 y n_0 tales que para todo $n \geq n_0$ se cumple que

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

Esto significa que $f(n)$ crece a la misma tasa que $g(n)$ para valores grandes de n.[?]

Tomando esto en cuenta, resolvamos las siguientes afirmaciones:

(a) Si $f(n) = 2^n$ y $g(n) = 2^{2^n+1}$, $f(n) = O(g(n))$

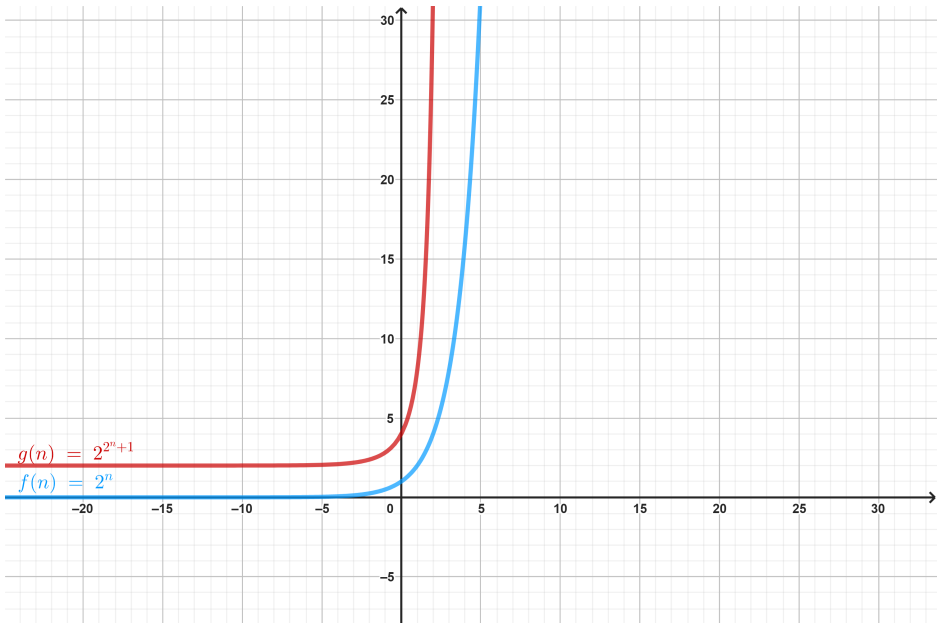
Tenemos que $f(n) = 2^n$ y $g(n) = 2^{2^n+1}$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c > 0$ tal que:

$$2^n \leq c \cdot 2^{2^n+1} \text{ para } n \geq n_0$$

Entonces, si desarrollamos la expresión dividiendo ambos lados de la desigualdad por 2^n obtenemos que:

$$1 \leq c \cdot 2^{2^n+1-n} \Rightarrow 1 \leq c \cdot 2^{2^n-n+1}$$

De esta forma, veamos que para que la desigualdad se cumpla, la expresión 2^{2^n-n+1} debe ser suficientemente grande, lo cual es verdad para valores grandes de n. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, la afirmación es **verdadera**

(b) Si $f(n) = 2^n$ y $g(n) = 2^{2^n+1}$, $f(n) = \Omega(g(n))$

Tenemos que $f(n) = 2^n$ y $g(n) = 2^{2^n+1}$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c > 0$ tal que:

$$2^n \geq c \cdot 2^{2^n+1} \text{ para } n \geq n_0$$

Entonces, si desarrollamos la expresión dividiendo ambos lados de la desigualdad por 2^n obtenemos que:

$$1 \geq c \cdot 2^{2^n+1-n} \Rightarrow 1 \geq c \cdot 2^{2^n-n+1}$$

De esta forma, veamos que para que la desigualdad se cumpla, la expresión 2^{2^n-n+1} debe tener un valor negativo o igual a 0. Esto es, no es posible encontrar una c que satisfaga la desigualdad. De hecho, notemos que por la respuesta del ejercicio anterior, esto no es posible que $f(n) = \Omega(g(n))$ dado que las definiciones de O y Ω son opuestas entre sí.

Por lo tanto, la afirmación es **falsa**

(c) Si $f(n) = n^3$ y $g(n) = 3n^3 + 10\sqrt{n}$, $g(n) = \Omega(f(n))$

Tenemos que $f(n) = n^3$ y $g(n) = 3n^3 + 10\sqrt{n}$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c > 0$ tal que:

$$3n^3 + 10\sqrt{n} \geq c \cdot n^3 \text{ para } n \geq n_0$$

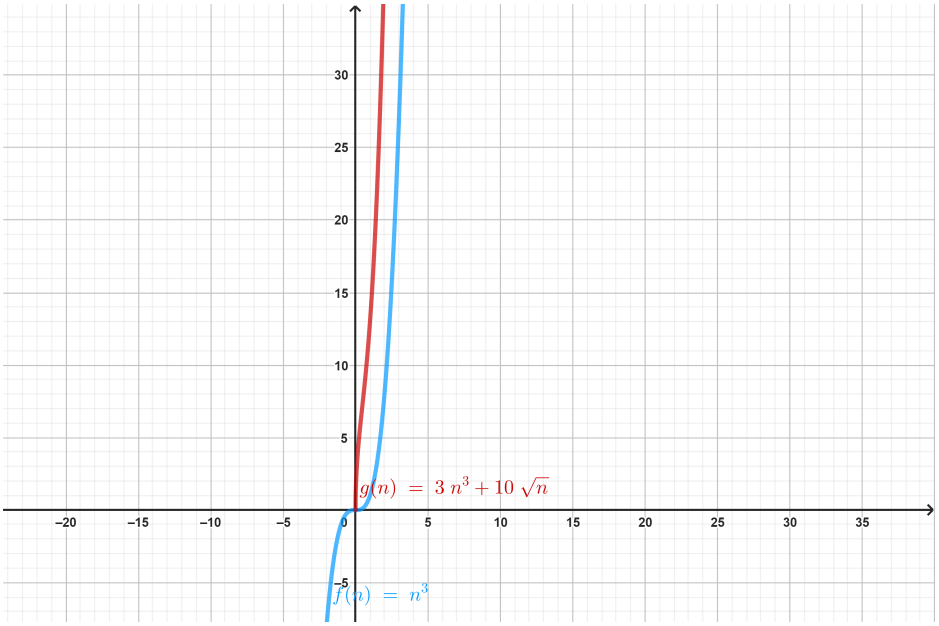
Entonces, si desarrollamos la expresión reorganizando los términos, obtenemos que:

$$3n^3 - c \cdot n^3 \geq -10\sqrt{n} \Rightarrow (3-c)n^3 \geq -10\sqrt{n}$$

Aquí, notemos que la condición para que la desigualdad se cumpla es qué $3 - c \geq 0 \Rightarrow c \leq 3$. Veamos que pasa en cada caso:

- Para $c = 3$ tenemos que:
$$0 \geq -10\sqrt{n}$$
- Para $c < 3$ tenemos que:
$$(3-c)n^3 \geq -10\sqrt{n}$$

De esta forma, veamos que la desigualdad se cumple en ambos casos para n grandes, esto es, el término $(3-c)n^3$ domina por encima de $-10\sqrt{n}$. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, la afirmación es **verdadera**

(d) Si $f(n) = \log(n)$ y $g(n) = \sqrt{n}$, $f(n) = O(g(n))$

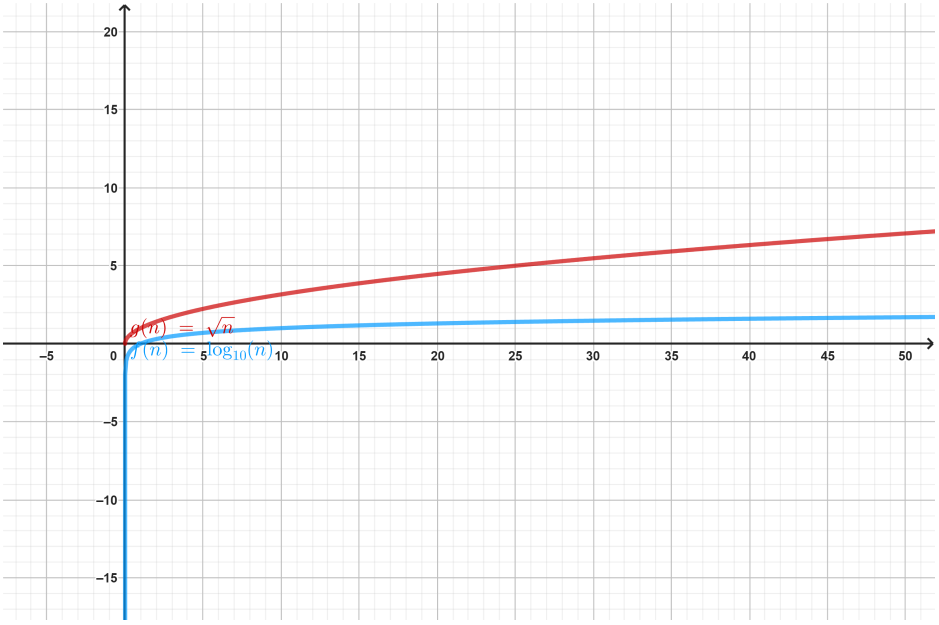
Tenemos que $f(n) = \log(n)$ y $g(n) = \sqrt{n}$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c > 0$ tal que:

$$\log(n) \leq c \cdot \sqrt{n} \text{ para } n \geq n_0$$

Entonces, notemos que si analizamos el comportamiento de $\log(n)$ y $\sqrt{n}$ encontramos que $\log(n)$ crece mucho más despacio que $\sqrt{n}$ a medida que n aumenta, por lo que siempre podremos encontrar una constante c tal que satisfaga que:

$$\log(n) \leq c \cdot \sqrt{n}$$

De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, la afirmación es **verdadera**

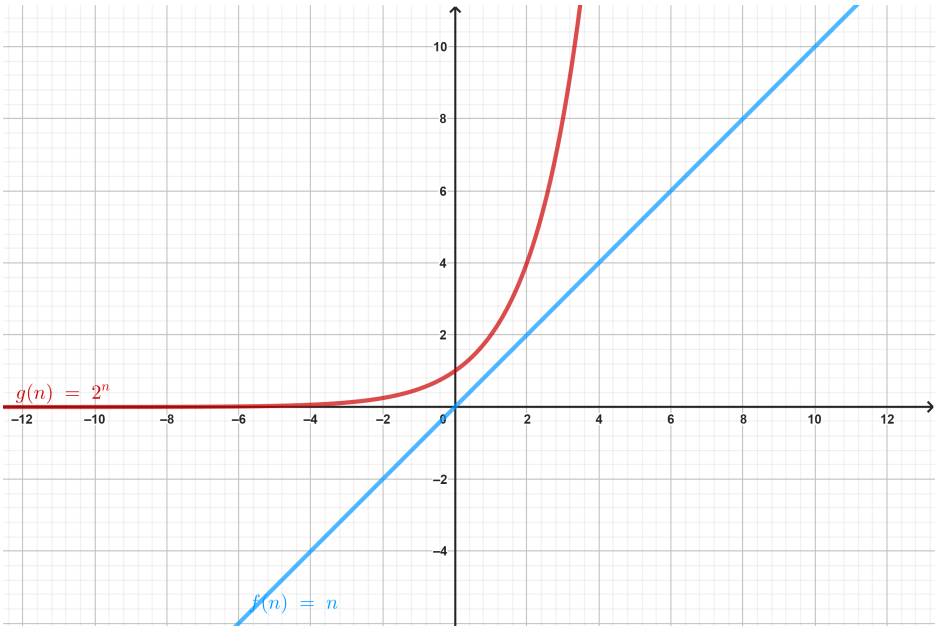
(e) Si $f(n) = n^k$ y $g(n) = c^n$, para algunas constantes $c > 1, k \geq 0$, entonces $g(n) = \Omega(f(n))$

Tenemos que $f(n) = n^k$ y $g(n) = c^n$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c' > 0$ tal que:

$$c^n \geq c' \cdot n^k \text{ para } n \geq n_0$$

Aquí, notemos que mientras que c^n es una función exponencial, n^k es una función polinomial, por lo que podemos decir que el crecimiento de c^n es mayor en comparación.

Por otro lado, veamos que dado que $c > 1$, entonces c^n aumenta exponencialmente con n, superando así a cualquier función polinómica. Esto es, existe una c' tal que c^n es al menos $c' \cdot n^k$. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función usando valores arbitrarios para c y k que estén dentro del rango indicado:



Por lo tanto, la afirmación es **verdadera**

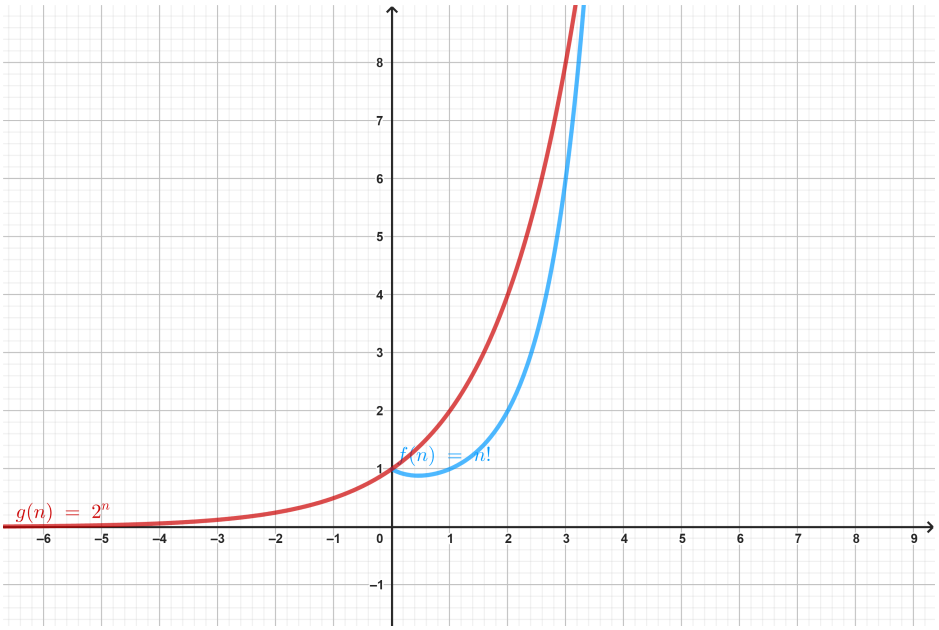
(f) Si $f(n) = n!$ y $g(n) = 2^n$, $f(n) = O(g(n))$

Tenemos que $f(n) = n!$ y $g(n) = 2^n$, por lo que usando las definiciones antes dadas, tendremos que probar si existe una constante $c > 0$ tal que:

$$n! \leq c \cdot 2^n \text{ para } n \geq n_0$$

Aquí, notemos que mientras que las expresiones factoriales crecen más rápido que cualquier función exponencial del tipo c^n, tal que c es una constante.

Por otro lado, notemos que para valores pequeños de n, 2^n puede ser mayor que $n!$, pero en cierto punto $n!$ empezará a ser más grande y continuará creciendo mucho más rápido. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, la afirmación es **falsa**

(g) Si $f(n) = O(n)$ y $g(n) = \Theta(n)$, entonces:

i. $f(n) + g(n) = O(n)$

Primero, notemos que, ya que las funciones f y g están dadas como $f(n) = O(n)$ y $g(n) = \Theta(n)$, entonces por definición tenemos que:

- Como $f(n) = O(n) \Rightarrow f(n) \leq c_1 \cdot n$ para algún $c_1 > 0$ y $n \geq n_0$
- Como $g(n) = \Theta(n) \Rightarrow g(n)$ está acotada inferior y superiormente por n, esto es, existen constantes $c_2, c_3 > 0$ tales que $c_2 \cdot n \leq g(n) \leq c_3 \cdot n$ para $n \geq n_0$.

De esta forma, para el primer ejercicio tenemos que:

$$f(n) \leq c_1 \cdot n \quad \text{y} \quad g(n) \leq c_3 \cdot n$$

Entonces, sustituyendo obtenemos que:

$$f(n) + g(n) \leq c_1 \cdot n + c_3 \cdot n = (c_1 + c_3) \cdot n$$

Esto es, $f(n) + g(n)$ están acotadas superiormente por una constante $(c_1 + c_3)$ multiplicada por n, por lo que:

$$f(n) + g(n) = O(n)$$

Por lo tanto, la afirmación es **verdadera**

i. $f(n)\, g(n) = O(n^2)$

Tomando en cuenta las expresiones dadas en el inciso anterior, sabemos que:

$$f(n) \leq c_1 \cdot n \quad \text{y} \quad g(n) \leq c_3 \cdot n$$

Entonces, sustituyendo obtenemos que:

$$f(n) \cdot g(n) \leq (c_1 \cdot n)(c_3 \cdot n) = c_1 \cdot c_3 \cdot n^2$$

Esto es, $f(n) \cdot g(n)$ están acotadas superiormente por una constante $(c_1 \cdot c_3)$ multiplicada por n^2, por lo que:

$$f(n) \cdot g(n) = O(n^2)$$

Por lo tanto, la afirmación es **verdadera**

i. $f(n) - g(n) = O(1)$

Tomando en cuenta las expresiones dadas en el primer inciso, sabemos que:

$$f(n) \leq c_1 \cdot n \quad \text{y} \quad c_2 \cdot n \leq g(n) \leq c_3 \cdot n$$

Entonces, si hacemos la resta $f(n) - g(n)$ tendremos 2 casos:

- Sí $f(n) < g(n)$, entonces $f(n) - g(n)$ tendrá un valor negativo y decreciente, por lo que no podrá ser acotado por una constante.
- Sí $f(n) > g(n)$, entonces $f(n) - g(n)$ tendrá un valor positivo y creciente, haciendo que la diferencia crezca junto con n, ya que $f(n)$ y $g(n)$ son lineales, por lo que su diferencia no podrá ser constante.

Esto es, $f(n) - g(n)$ no puede ser acotada por una constante para valores grandes de n.

Por lo tanto, la afirmación es **falsa**

(h) Existen funciones f, g tales que no se cumple que $f(n) = O(g(n))$ ni $f(n) = \Omega(g(n))$

Sí, existen funciones f y g tales que cumplen las condiciones dadas, a continuación, veamos dos ejemplos de este tipo de funciones:

1. **Ejemplo 1:**

   Sean $f(n) = n^{\sin(n)}$ *y* $g(n) = n$

   Aquí, notemos que las funciones f y g no cumplen que $f(n) = O(g(n))$ ni $f(n) = \Omega(g(n))$. Esto es:

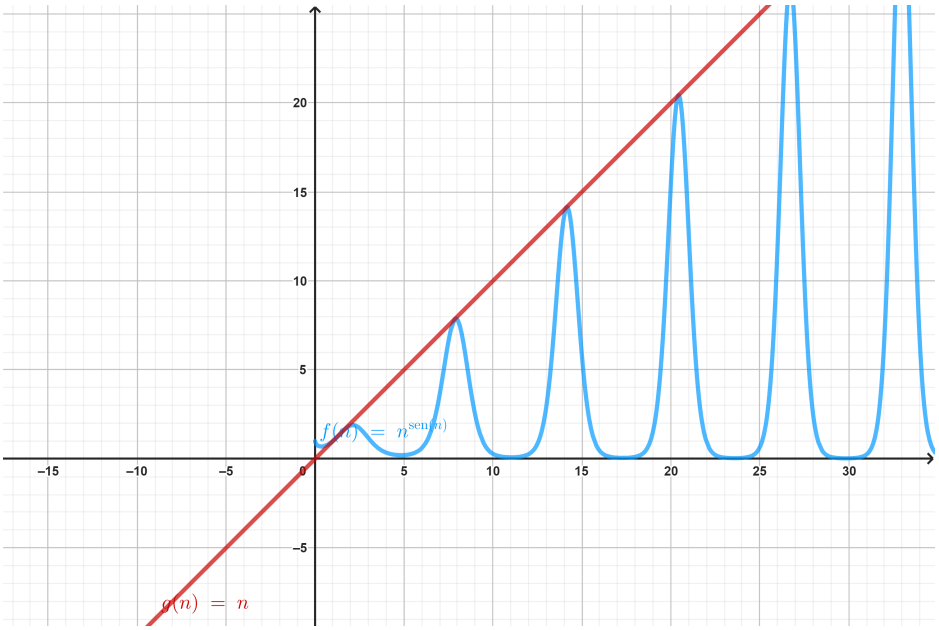
   Supongamos qué $f(n) = O(g(n))$. Tomando en cuenta las definiciones dadas al principio de este ejercicio, veamos esto significaría que existen constantes c y n_0 tales que para todo $n \geq n_0$ se cumple que:

   $$0 \leq f(n) \leq c \cdot g(n)$$

Sin embargo, notemos que $f(n) = n^{\sin(n)}$ oscila entre n^{-1} y n, lo que hace imposible encontrar una constante c que satisfaga la condición dada para todos los valores de n.
Por otro lado, también supongamos qué $f(n) = \Omega(g(n))$. Nuevamente, considerando las definiciones dadas al comienzo de este ejercicio, esto significaría que existen constantes c y n_0 tales que para todo $n \geq n_0$ se cumple que:

$$0 \leq c \cdot g(n) \leq f(n)$$

Similarmente, debido a la oscilación de $\sin(n)$, $f(n)$ puede ser tan pequeña como n^{-1}, hace imposible encontrar una constante c que satisfaga la condición dada para todos los valores de n. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, $f(n) = n^{\sin(n)}$ y $g(n) = n$ son un ejemplo de funciones que no cumplen que $f(n) = O(g(n))$ ni $f(n) = \Omega(g(n))$.

2. **Ejemplo 2:**

   Sean $f(n) = \log(\log(n))$ *y* $g(n) = \log(n)$

   Aquí, notemos que las funciones f y g no cumplen que $f(n) = O(g(n))$ ni $f(n) = \Omega(g(n))$. Esto es:

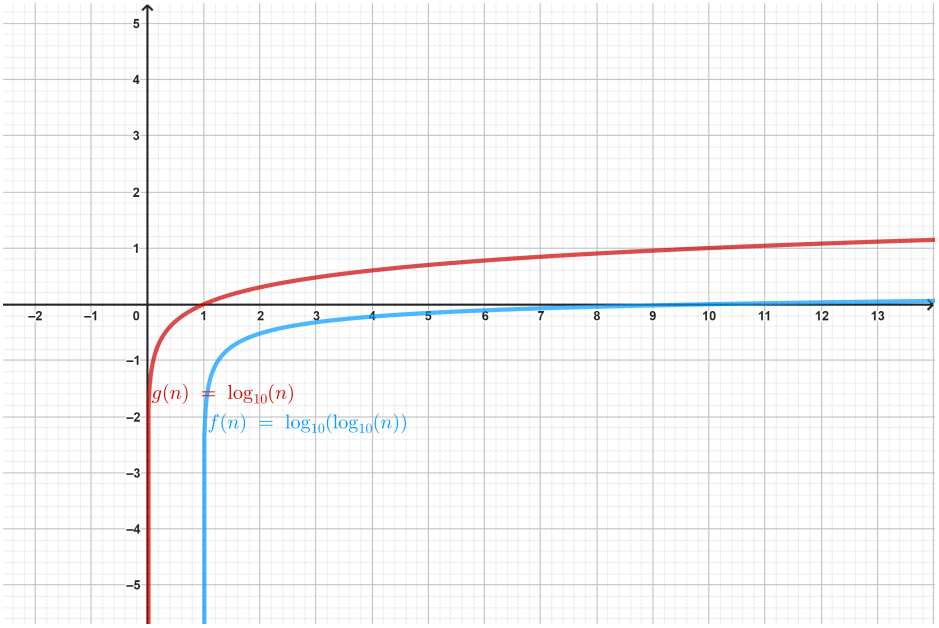
   Supongamos qué $f(n) = O(g(n))$. Tomando en cuenta las definiciones dadas al principio de este ejercicio, veamos esto significaría que existen constantes c y n_0 tales que para todo $n \geq n_0$ se cumple que:

   $$0 \leq f(n) \leq c \cdot g(n)$$

   Sin embargo, notemos que $\log(\log(n))$ crece mucho más lentamente que $\log(n)$, lo que hace imposible encontrar una constante c que satisfaga la condición dada para todos los valores de n.
   Por otro lado, también supongamos qué $f(n) = \Omega(g(n))$. Nuevamente, considerando las definiciones dadas al comienzo de este ejercicio, esto significaría que existen constantes c y n_0 tales que para todo $n \geq n_0$ se cumple que:

   $$0 \leq c \cdot g(n) \leq f(n)$$

Similarmente, debido a que $\log(\log(n))$ es siempre menor que $\log(n)$ para n suficientemente grande, hace imposible encontrar una constante c que satisfaga la condición dada para todos los valores de n. De hecho, veamos que esto sí sucede viendo las gráficas dadas para cada función:



Por lo tanto, $f(n) = \log(\log(n))$ y $g(n) = \log(n)$ son un ejemplo de funciones que no cumplen que $f(n) = O(g(n))$ ni $f(n) = \Omega(g(n))$.

Ejercicio 5

Considera el siguiente problema:

La facultad tiene clubes en los que se organizan miembros de la comunidad para realizar diferentes actividades (deportes, talleres, seminarios, etc.), de los cuales en el más grande hay m miembros, todos estudiantes; toma en cuenta que los estudiantes pueden estar afiliados a más de un club. El director de la facultad quiere hacer un evento en honor de tales actividades estudiantiles.

Desafortunadamente, dadas las limitantes de espacio, solo se permite que haya un número máximo de k invitados. Por tal motivo, el director debe limitar la lista de invitados a k estudiantes tal que cada club tenga al menos un miembro asistente.

- Enuncia el problema anterior como un problema de búsqueda.
  Solución.
  Ejemplar: Dado un conjunto de m estudiantes organizados en distintos n clubes, donde cada club tiene un numero de estudiantes y un estudiante puede estar afiliado a mas de un club, y un entero positivo k
  Pregunta: Encontrar un subconjunto que satisfaga las siguientes condiciones:

  - El subconjunto debe tener a lo mas k estudiantes.
  - Cada club debe estar representado por al menos un miembro del subconjunto.

- Enuncia el problema anterior, en su versión canónica, como un problema de decisión.
  Solución.
  Ejemplar: Dado un entero positivo k, un subconjunto de clubes $C_1, C_2, ..., C_n$, donde cada club C_i tiene un conjunto de miembros S_i
  Pregunta: ¿Existe un subconjunto S' tal que $|S'| \leq k$ y cada club C_i esta representado por al menos un miembro de S'?

- Da un ejemplar concreto para el problema de decisión y muestra una posible solución para el ejemplar dado.
  Solución.
  Supongamos que tenemos 3 clubes:

  - $C_1 = \{A, B\}$
  - $C_2 = \{B, C\}$
  - $C_3 = \{C, D\}$

  y tenemos $k = 3$, entonces proponemos el subconjunto $S' = \{B, C, D\}$ y veamos que cumple las condiciones:

  - $|S'| \leq 3$
  - C_1 esta representado por B, C_2 esta representado por B y C, C_3 esta representado por C y D

- Propón un algoritmo de búsqueda exhaustiva para resolver el problema.
  Solución.
  Proponemos el siguiente algoritmo de busqueda exhaustiva:

  - Generar todos los subconjuntos posibles de estudiantes de tamaño $\leq k$

- Para cada subconjunto, verificar si cada club C_i tiene al menos un miembro en dicho subconjunto.
- Si existe un subconjunto que cumpla la condicion, devolver ´´si´´
- Si no existe ningun subconjunto que satisfaga la condicion, devolver ´´no´´

- ¿Cuál es la complejidad del algoritmo propuesto? Justifica ampliamente tu respuesta.
  Solución.
  Sabemos que generar todos los subconjuntos de un conjunto dado es $O(2^m)$ donde m es el número de elementos del conjunto, entonces generar todos los subconjuntos de alumnos es $O(2^m)$ donde m es el número de alumnos.
  Para saber si un subconjunto cumple la condición $|S'| \leq k$ toma tiempo constante.
  Para verificar que cada club este representado por al menos un alumno en S', debemos iterar por todos los clubes, supongamos que tenemos n número de clubes, y tenemos que iterar por cada miembro de S' para verificar si existe en C_i, y haremos a lo mas k busquedas, ya que nuestro subconjunto S' tiene a lo mas k alumnos, por lo que este paso toma $O(n \cdot k)$, entonces, la complejidad de nuestro algoritmo es $O(2^m \cdot n \cdot k)$.

- Ilustra la ejecución del algoritmo propuesto con un ejemplar concreto.
  Solución.
  Supongamos que tenemos los siguientes clubes y estudiantes:

  - $C_1 = \{A, B\}$
  - $C_2 = \{B, C\}$

  Tenemos $k = 2$.

  - **Estudiantes**: $\{A, B, C\}$
  - **Máximo de invitados** $k = 2$
  - **Subconjuntos de tamaño** $\leq k$:
    - $\{A\}$
      Verificación:
      - Cubre Club 1 (sí, porque $A \in \{A, B\}$).
      - No cubre Club 2 (no, porque $A \notin \{B, C\}$).
      Conclusión: No cubre todos los clubes.
    - $\{B\}$
      Verificación:
      - Cubre Club 1 (sí, porque $B \in \{A, B\}$).
      - Cubre Club 2 (sí, porque $B \in \{B, C\}$).
      Conclusión: **Cubre todos los clubes**.
    - $\{C\}$
      Verificación:
      - No cubre Club 1 (no, porque $C \notin \{A, B\}$).
      - Cubre Club 2 (sí, porque $C \in \{B, C\}$).
      Conclusión: No cubre todos los clubes.
    - $\{A, B\}$
      Verificación:
      - Cubre Club 1 (sí, porque $A \in \{A, B\}$ y $B \in \{A, B\}$).
      - Cubre Club 2 (sí, porque $B \in \{B, C\}$).
      Conclusión: **Cubre todos los clubes**.

- $\{A, C\}$
  Verificación:
  - Cubre Club 1 (sí, porque $A \in \{A, B\}$).
  - No cubre Club 2 (no, porque $C \in \{B, C\}$, pero falta B).
  Conclusión: No cubre todos los clubes.
- $\{B, C\}$
  Verificación:
  - Cubre Club 1 (sí, porque $B \in \{A, B\}$).
  - Cubre Club 2 (sí, porque $B \in \{B, C\}$ y $C \in \{B, C\}$).
  Conclusión: **Cubre todos los clubes**.

Observamos que cuando encontramos un subconjunto que cumple las condiciones simplemente devolvemos ´´si´´ y termina la ejecución pero decid mostrar todos los subconjuntos para observar que hay diferentes soluciones.

Referencias

[1] Big-O notation. (s.f.). NIST. https://xlinux.nist.gov/dads/HTML/bigOnotation.html

[2] Omega. (s.f.). NIST. https://xlinux.nist.gov/dads/HTML/omegaCapital.html

[3] Theta. (s.f.). NIST. https://xlinux.nist.gov/dads/HTML/theta.html

[4] GeeksforGeeks. (2019, October 31). Sum of subsets of all the subsets of an array

[5] 14. Some Graph Theory. (2024). Mit.edu. `https://math.mit.edu/~djk/18.310/18.310F04/some_graph_theory.html`