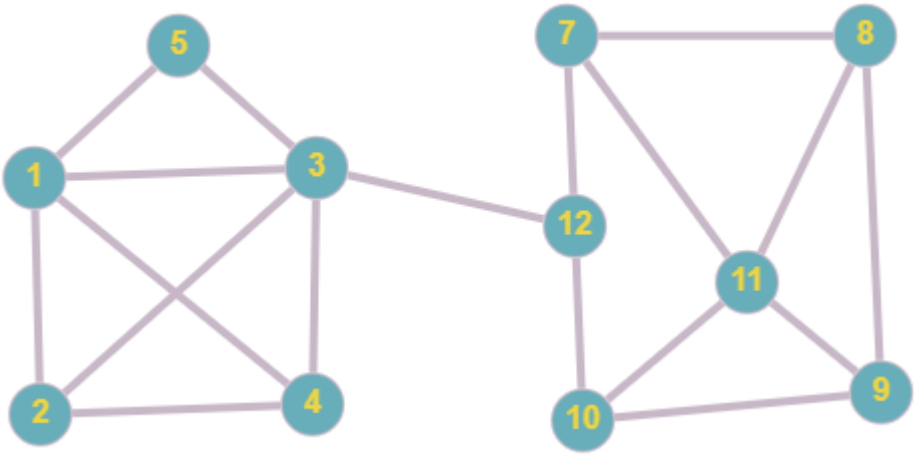
Algoritmia y Complejidad
Problema: Dominating set pertenece a NPC
David Rosales García

1) Explicación de Dominating Set

El problema Dominating Set se define de la siguiente manera:
El conjunto dominante de un grafo G = <V,k> se trata de un subgrafo D de V, tal que todo vértice que no pertenece a D, es adyacente a D, esto último quiere decir que estrictamente cualquier vértice de V, que no se encuentre en D debe de estar a una distancia de 1 respecto a al menos 1 vértice de D, el número de elementos de D debe ser menor o igual a k, esto se vé mejor con un par de ejemplos.

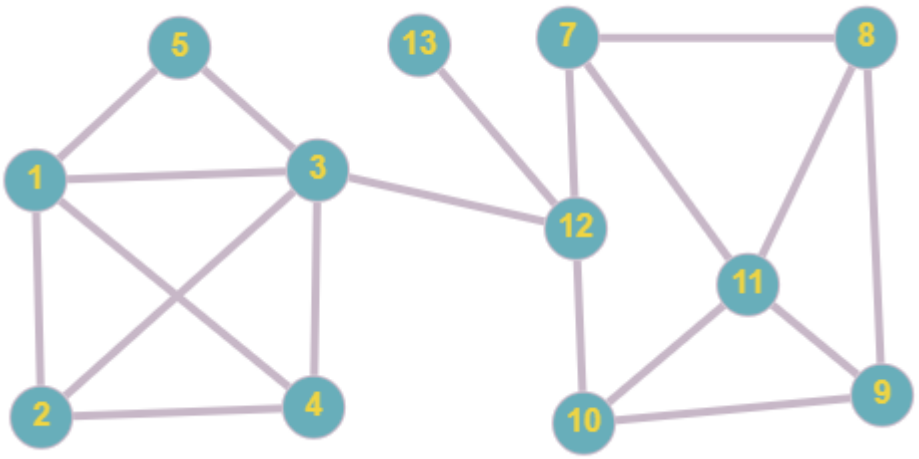
1.1) Ejemplo válido



Para este primer ejemplo tenemos un grafo G que lo componen todos los vértices del grafo, la el grafo D que es solución para este grafo sería el cualquier subgrafo que contenga a minimamente a 3 y a 11, por eso mismo el subgrafo más pequeño que sería Dominating Set sobre G es: D = {3,11}.

1.2) Ejemplo no válido

Ahora vamos a poner que nos quedamos con el mismo D, pero al grafo G se le añade el vértice 13 y este nuevo grafo pasa a llamarse G':



Como se puede ver el vértice 13 no es adyacente ni a 3 ni a 11, por lo que el D anterior no cumpliría ser Dominating Set de este nuevo grafo, para que lo fuese tendríamos que crear un nuevo subgrafo, D' que contenga a D, además de en este caso un nuevo vértice que esté conectado a 13, para este caso el subgrafo D' sería el siguiente: D = {3,11,12}.

1.3) Campo de aplicación

Este tipo de problema tiene varios usos como por ejemplo, el diseño de circuitos electrónicos o la planificación de rutas y logística, este último es el que a mi se me hace más evidente ya que el dominating set como se puede ver claramente en el apartado anterior lo que busca es en cubrir la mayor cantidad de espacio con el menor número de miembros posibles para abarcar dicho espacio conectado, a esto también se le tendrían que asignar pesos en las aristas para poder hacer un algoritmo más adecuado, pero en la simplicidad de este problema se puede ver para qué tipo de problemas este ejercicio puede ayudar.

2) Demostración de pertenencia a NP

Como bien sabemos para demostrar la pertenencia a la clase NP, tenemos 2 métodos distintos, el primero es encontrar un verificador polinómico, y el otro se trata de encontrar una máquina de Turing no determinista que compute el ejercicio en tiempo polinómico, veremos ambos en este documento.

2.1) Vía verificador polinómico

El verificador sería el siguiente: V con entrada <<G,k>, c>, donde G es el grafo donde se mirará si hay un Dominating Set, k el tamaño máximo para el subgrafo del problema, y c será el subgrafo en sí mismo, y V funciona de la siguiente manera:

1 - Se comprueba que |c| <= k.

2 - Se separarán los vértices de V que se encuentran en c de los que no están.

3 - Se comprueba que cada vértice de V tiene una arista que la conecta con al

menos 1 vértice de c.
4 - Al completarse los anteriores pasos bien se acepta, si no se rechaza.

Como apunte final a esto tenemos que ver que cuanto se tarda en cada paso de los anteriormente mencionados para saber que se hacen en tiempo polinómico, en caso de no serlo el verificador no sería válido.

Los pasos 1 y 4 son triviales, es un simple cálculo en el 1 y una comprobación sencilla en el siguiente, por lo que son O(n), en el caso del paso 2 tenemos que pasar varias veces por V y por c de manera separada para poder hacer las 2 listas que nos ayudan a filtrar para el siguiente paso, por lo que al tener que pasar por V y por c 2 veces por cada componente de c, podemos decir que le corresponde una complejidad $O(n^2)$, por último para el paso 3, primero tendremos que pasar los elementos de V, segundo tendremos que comprobar si son adyacentes a algún elemento de c, y en caso de que no lo fuesen tendríamos que comprobar si ese elemento de V se encuentra en c, por lo que se puede llegar a pasar 3 veces por la lista, por lo que la complejidad de este apartado es de $O(n^3)$.

Al sumarlas todas queda $O(n) + O(n^2) + O(n^3) + O(n) = O(n^3)$

2.2) Vía máquina de Turing no determinista.
La máquina de Turing se define de la siguiente manera, M con una entrada <G,k>, donde M se refiere a la máquina de Turing, G al grafo y k al tamaño máximo del subgrafo del Dominating Set.

1 - Primeramente M seleccionará de manera no determinista un subconjunto de G que corresponda con c.
2 - Se ejecutará V con la entrada <<G,k>, c>
3 - En caso de que se acepte, acepta, en caso contrario, rechaza.

Como se puede observar hace uso del verificador anterior, pero no se me ocurría otra manera de hacerlo.

A nivel de complejidad el primer paso solo tiene que coger elementos de la lista y hacerlo de manera no determinista, lo puede pasando solo una vez por cada elemento de la lista y con eso valdría así que su complejidad es de O(n), en el paso 2, es trivial calcularlo, tardará lo mismo para lo cual determinamos en el apartado anterior así que la complejidad será de O(n3), y el último paso es trivial y tendrá una complejidad de O(n), así que la complejidad temporal de la máquina de Turing M será de: $O(n) + O(n^3) + O(n) = O(n^3)$

Con cualquiera de los 2 apartados que se acaban de mencionar y hacer, queda demostrado que Dominating Set se trata de un problema que se encuentra dentro de la clase de complejidad NP.

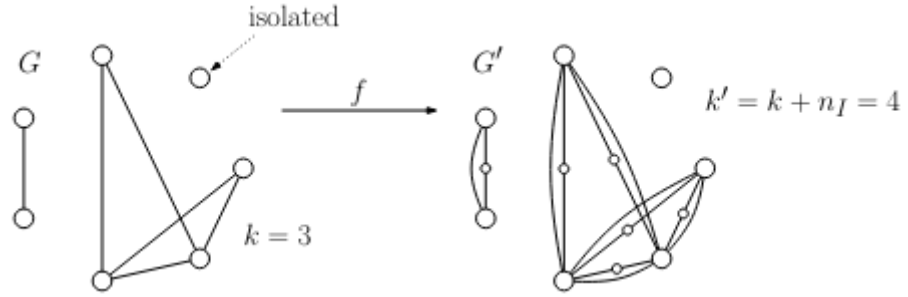
3) Reducción a otro problema NPC (conocido)
El objetivo de este apartado es mediante la reducción en tiempo polinómico, es demostrar que Dominating Set se trata de un problema NP-duro/difícil/hard, lo cual hace que junto la propiedad anteriormente demostrada, quede que Dominating Set se trata de un problema

NPC, un problema que pertenece a NP, y que a su vez puede ser reducido a cualquier problema NPC en tiempo polinómico.

Yo haré la reducción respecto a Vertex Cover, es decir haré la reducción VC $\leq_p$ DS (VC es Vertex Cover, y DS es Dominating Set).

Queremos enseñar que para una instancia del problema VC, podemos producir de manera equivalente una instancia de DS en tiempo polinómico, para ello crearemos un grafo G', el cual será inicialmente igual a G, y para cada arista de G se creará un nuevo vértice entre ambos, por ejemplo para las aristas (u, v) se creará un vértice entre ambos llamado w_{uv}, entonces las aristas se dividen y pasan a conectar los pares de vértices (u, w_{uv}) y (w_{uv}, v). I denota el número de vértices aislados y k' = k + n_I. Hay que decir que el hecho de se cree un nuevo vértice y que haya 2 nuevas aristas no quiere decir que las aristas que había antes sean reemplazadas, sino que simplemente se añaden.

Un ejemplo de esta reducción queda de la siguiente manera:



Para establecer que la reducción es correcta necesitamos enseñar que G tiene un VC si y sólo si G' tiene un DS.

Primero argumentamos que si V' tiene un VC para G, entonces V'' = V' U V_I será un DS para G'. Observamos que:

$$|V''| = |V' \cup V_I| <= k + n_I = k'$$

|V''| puede ser más pequeño que k + n_I, en caso de que haya algún vértice aislado en V', para ese caso concreto se añadirán tantos vértices como sean necesarios para hacer que ambos coincidan.
Para ver si en V'' hay DS primero nos fijamos en los vértices aislados ya que estos deben estar dentro del conjunto. Después miramos los vértices del tipo w_{uv} en G', que corresponden o bien con u o v, y forman parte del VC de V', por lo que w_{uv} está dentro de DS en V''. Para acabar cada vértice no aislado del conjunto original de vértices está "pegado" a al menos 1 arista de G, y por lo tanto de V' o si no todos sus "vecinos" están en V'.
Por el otro lado, podemos decir que G' tiene un DS en V'' de tamaño k' = k + n_I y G tiene un VC, V' de tamaño k. Es importante notar que todos los vértices aislados de G' deben estar en DS, hay tantos como n_I. Primero V''' = V'' / V_I serán los k vértices restantes.
No nos hará falta usar ninguno de los vértices que hemos creado en V''', ya que para cualquier vértice w_{uv} está pegado a u y v, y con que cualquiera de ellos esté dentro del DS, este domina a los otros 2, en incluso más, porque w_{uv} está conectado exclusivamente a u y v, en cambio los otros pueden estar conectados a más.

Podemos decir que V' tiene un VC para G, porque si por el contrario hubiese una arista de G que no estuviese cubierta, entonces el vértice no estaría adyacente a V'' en G', contradiciendo la hipótesis de que V'' tiene un DS en G'.
Y con esto quedaría demostrada la corrección de la reducción.

4)Preguntas

Para las preguntas he pensando en empezar con una sencilla, dando un par de conjuntos a los cuales se les pediría a los compañeros buscarles el set dominante más pequeño posible, o como variación de ese mismo ejercicio se podrían plantear unas cuantas posibilidades y plantear cuales son Dominating Set correctos y cuáles no.
Como segundo ejercicio puede pedirse algo muy parecido al primero, o bien hacer que los compañeros propongan un grafo que corresponda con un gadget correcto, o bien proporcionar unos ejemplos y que los compañeros decidan cuales son correctos y cuáles no.
Para acabar y como tercer ejercicio se podría preguntar a los compañeros si intuyen algún campo en el cual este ejercicio puede trasladarse al mundo real, ya que con ayuda de los grafos es visualmente intuitivo muchos de los posibles usos que tiene.

Referencias

https://people.computing.clemson.edu/~goddard/texts/theoryOfComputation/19b.pdf

https://theory.stanford.edu/~rajeev/CS154/solution7.pdf

https://cs.rhodes.edu/welshc/COMP355_S14/Lecture26.pdf

https://www.cs.umd.edu/class/fall2017/cmsc451-0101/Lects/lect21-np-clique-vc-ds.pdf