Problem Set - DAA

Problemas NP, NP-Completo, NP-Hard

Marian S. Álvarez Suri - C412
Carlos A. Bresó Sotto - C412

17 de diciembre de 2024

Problemas

Asumiendo que los siguientes problemas ya se han demostrado como NP-Completos:

- SAT
- 3-Sat
- Clique
- Conjunto independiente
- Vertex Cover
- Subset Sum
- Mochila
- Hamilton (Dirigido)
- Viajante

Demuestre que los siguientes problemas son NP-Hard o NP-Completos, según corresponda.

1. Set Cover

1.1. Demostración de NP-Completitud usando reducción de SAT a Set Cover

Dado un conjunto $F = \{C_1, \ldots, C_\ell\}$ de ℓ cláusulas construidas a partir de n variables proposicionales $x_1, \ldots, x_n$, debemos construir en tiempo polinómico una instancia $\tau(F) = (U, F)$ de Cobertura Exacta tal que F sea satisfacible si y solo si $\tau(F)$ tiene una solución.
El método general para construir (U, F) a partir de $F = \{C_1, \ldots, C_\ell\}$ procede de la siguiente manera. Supongamos que

$$C_j = (L_{j1} \vee \cdots \vee L_{jm_j})$$

es la j-ésima cláusula en F, donde L_{jk} denota el k-ésimo literal en C_j y $m_j \geq 1$.
El universo de $\tau(F)$ es el conjunto

$$U = \{x_i \mid 1 \leq i \leq n\} \cup \{C_j \mid 1 \leq j \leq \ell\} \cup \{p_{jk} \mid 1 \leq j \leq \ell, 1 \leq k \leq m_j\}$$

donde en el tercer conjunto p_{jk} corresponde al k-ésimo literal en C_j.
Los siguientes subconjuntos están incluidos en F:

a) Hay un conjunto $\{p_{jk}\}$ para cada p_{jk}.

b) Para cada variable booleana x_i, los siguientes dos conjuntos están en F:

$$T_{i,T} = \{x_i\} \cup \{p_{jk} \mid L_{jk} = \overline{x_i}\}$$

que contiene x_i y todas las ocurrencias negativas de x_i, y

$$T_{i,F} = \{x_i\} \cup \{p_{jk} \mid L_{jk} = x_i\}$$

que contiene x_i y todas sus ocurrencias positivas.

Nótese cuidadosamente que $T_{i,T}$ involucra ocurrencias negativas de x_i mientras que $T_{i,F}$ involucra ocurrencias positivas de x_i.

c) Para cada cláusula C_j, los m_j conjuntos $\{C_j, p_{jk}\}$ están en F.

Queda por demostrar que F es satisfacible si y solo si $\tau(F)$ tiene una solución. Afirmamos que si v es una asignación de verdad que satisface F, entonces podemos hacer una cobertura exacta C de la siguiente manera:

Para cada x_i, ponemos el subconjunto $T_{i,T}$ en C si y solo si $v(x_i) = T$; de lo contrario, ponemos el subconjunto $T_{i,F}$ en C si y solo si $v(x_i) = F$.

Además, para cada cláusula C_j, ponemos algún subconjunto $\{C_j, p_{jk}\}$ en C para un literal L_{jk} que es hecho verdadero por v.
Por la construcción de $T_{i,T}$ y $T_{i,F}$, este p_{jk} no está en ningún conjunto en C seleccionado hasta ahora. Dado que por hipótesis F es satisfacible, tal literal existe para cada cláusula. Habiendo cubierto todos los x_i y C_j, ponemos un conjunto $\{p_{jk}\}$ en C para cada p_{jk} restante que aún no ha sido cubierto por los conjuntos ya en C.
Inversamente, si C es una cobertura exacta de $\tau(F)$, definimos una asignación de verdad de la siguiente manera:

Para cada x_i, si $T_{i,T}$ está en C, entonces asignamos

$$v(x_i) = T,$$

de lo contrario, si $T_{i,F}$ está en C, entonces asignamos

$$v(x_i) = F$$

2. Clique Máximo

2.1. Demostración de NP-Hard usando reducción de 3-SAT a Clique Máximo

Consideremos una entrada para 3-SAT, que es una fórmula F definida sobre n variables con m cláusulas.
Construimos el grafo G de la fórmula F de la siguiente manera:

1. Para cada literal en la fórmula generamos un vértice y etiquetamos el vértice con el literal al que corresponde. Cada cláusula corresponde a tres de estos vértices.

2. Conectamos dos vértices en el grafo si:

   a) están en diferentes cláusulas, y

   b) no son la negación uno del otro.

Vamos a probar que F es satisfacible si y solo si existe un clique de tamaño m en G:

Parte 1: Si F es satisfacible, entonces existe un clique de tamaño m en G.
Sean $x_1, \ldots, x_n$ las variables que aparecen en F, y sean $v_1, \ldots, v_n \in \{0, 1\}$ la asignación que satisface F. La fórmula F se cumple si asignamos $x_i = v_i$, para $i = 1, \ldots, n$.
Para cada cláusula C en F debe haber al menos un literal que evalúe a VERDADERO.
Elige un vértice que corresponda a dicho valor VERDADERO de cada cláusula. Sea W el conjunto resultante de vértices. Claramente, W forma un clique en G. El conjunto W tiene tamaño m, ya que hay m cláusulas y cada una contribuye con un vértice al clique.

Parte 2: Si existe un clique de tamaño m en G, entonces F es satisfacible.
Sea U el conjunto de m vértices que forman un clique en G. Necesitamos traducir el clique G_U a una asignación satisfactoria de F.

(i) Asignar $x_i \leftarrow$ TRUE si hay un vértice en U etiquetado con x_i.

(ii) Asignar $x_i \leftarrow$ FALSE si hay un vértice en U etiquetado con $\overline{x_i}$.

Vamos a probar que la asignación es válida utilizando reducción al absurdo:
Supongamos que existe x_i tal que hay dos vértices u, v en U etiquetados con x_i y $\overline{x_i}$; es decir, asignamos valores contradictorios a x_i. Entonces, u y v, por construcción, no estarán conectados en G, y como tal G_U no es un clique. Esto produce una contradicción por lo que lo supuesto es falso, y la asignación es válida.
Para demostrar que es una asignación satisfactoria, notemos que hay al menos un vértice de U en cada cláusula, por lo que hay un literal evaluado como VERDADERO en cada cláusula. Por tanto, F es VERDADERO.

Por tanto, dado un algoritmo para clique máximo en tiempo polinomial, podemos resolver 3-SAT en tiempo polinomial, lo cual implica que el problema de encontrar el clique máximo de un grafo es NP-Hard.

2.2. Demostración de que Clique (Máximo) es NP

Dado un grafo G con n vértices, un parámetro k, y un conjunto W de k vértices, verificar que cada par de vértices en W forma una arista en G toma $O(u + k^2)$, donde u es el tamaño de la entrada. Por tanto, verificar una respuesta positiva a una instancia de Clique se puede hacer en tiempo polinómico, lo que implica que pertenece a NP.

2.3. Demostración de que Clique es NP-Completo

Como el problema es NP-Hard y es NP, podemos concluir que es NP-Completo.

3. Número Cromático

Primero vamos a probar que el problema de 3-coloreo es NP-Completo y, a partir de esto, que el problema de hallar el número cromático de un grafo es NP-Completo.

3.1. Demostración de que 3-coloreo es NP

Dado G y k, un certificado de que la respuesta es sí es simplemente un k-coloreo. Se puede verificar en tiempo polinómico que se usan a lo sumo k colores y que ningún par de nodos unidos por una arista recibe el mismo color.

3.2. Demostración de NP-Completitud usando reducción de 3-SAT a 3-Coloreo

Dada una instancia de 3-SAT, con variables $x_1, \ldots, x_n$ y cláusulas $C_1, \ldots, C_k$, lo resolveremos usando una caja negra para 3-Coloreo.
Definimos nodos v_i y $\overline{v_i}$ correspondientes a cada variable x_i y su negación $\overline{x_i}$. También definimos tres ¨nodos especiales¨ T, F y B, que denominamos True, False y Base. Para comenzar, unimos cada par de nodos v_i, $\overline{v_i}$ entre sí con una arista, y unimos ambos nodos a Base. (Esto forma un triángulo en v_i, $\overline{v_i}$ y Base, para cada i). También unimos True, False y Base en un triángulo (Figura 1).

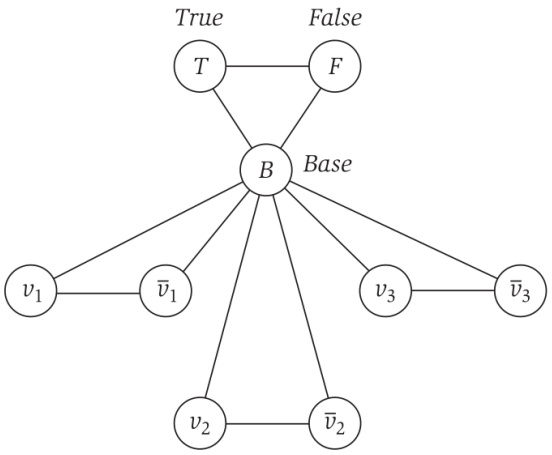

Figura 1:

El grafo G resultante tiene las siguientes propiedades:

- En cualquier 3-coloreo de G, los nodos v_i y $\overline{v_i}$ deben obtener colores diferentes, y ambos deben ser diferentes de Base.
- En cualquier 3-coloreo de G, los nodos True, False y Base deben recibir los tres colores en alguna permutación. Por lo tanto, podemos referirnos a los tres colores como el color True, el color False y el color Base, según cuál de estos tres nodos obtenga qué color. En particular, esto significa que para cada i, uno de v_i o $\overline{v_i}$ obtiene el color True, y el otro obtiene el color False. Para el resto de la construcción, consideraremos que la variable x_i está establecida en 1 en la instancia dada de 3-SAT si y solo si el nodo v_i recibe el color True.

Tenemos entonces un grafo G en el que cualquier 3-coloreo determina implícitamente una asignación de verdad para las variables en la instancia de 3-SAT. Ahora necesitamos ampliar G para que solo las asignaciones satisfactorias se puedan extender a 3-coloreos del grafo completo.

Como en otras reducciones de 3-SAT, consideremos una cláusula como $x_1 \vee x_2 \vee x_3$, que representa que al menos uno de los nodos v_1, v_2 o v_3 debería obtener el color True. Así que lo que necesitamos es un pequeño subgrafo que podamos enchufar a G, de modo que cualquier 3-coloreo que se extienda a este subgrafo debe tener la propiedad de asignar el color True a al menos uno entre v_1, v_2 o v_3. El grafo de la Figura 2 cumple con dicho objetivo.

Este subgrafo de seis nodos se adjunta al resto de G en cinco nodos existentes: True, False y aquellos correspondientes a los tres términos en la cláusula que estamos tratando de representar (en este caso, v_1, v_2 y v_3). Ahora supongamos que en algún

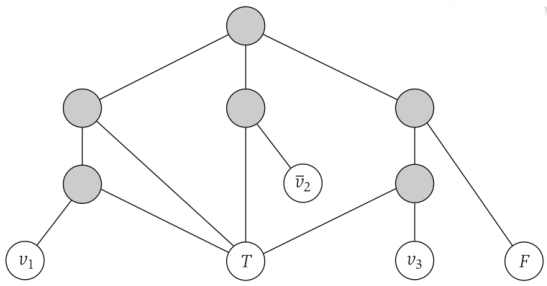

Figura 2:

3-coloreo de G todos los tres v_1, v_2 y v_3 reciben el color False. Entonces los dos nodos sombreados inferiores en el subgrafo deben recibir el color Base, los tres nodos sombreados por encima de ellos deben recibir, respectivamente, los colores False, Base y True, y por lo tanto no hay color que se pueda asignar al nodo sombreado superior. En otras palabras, un 3-coloreo en el que ninguno de v_1, v_2 o v_3 recibe el color True no se puede extender a un 3-coloreo de este subgrafo. Finalmente, y en sentido contrario, algunos chequeos manuales de casos muestran que mientras al menos uno de v_1, v_2 o v_3 reciba el color True, el subgrafo completo puede ser coloreado con 3 colores.
Entonces, a partir de esto, podemos completar la construcción: comenzamos con el grafo G definido arriba, y para cada cláusula en la instancia de 3-SAT, adjuntamos un subgrafo de seis nodos como se muestra en la Figura 2. Llamemos G' al grafo resultante.
Ahora afirmamos que la instancia dada de 3-SAT es satisfactoria si y solo si G' tiene un 3-coloreo. Primero, supongamos que hay una asignación satisfactoria para la instancia de 3-SAT. Definimos una coloración de G' coloreando primero Base, True y False arbitrariamente con los tres colores; luego, para cada i, asignando a v_i el color True si $x_i = 1$ y el color False si $x_i = 0$. Luego asignamos a $\overline{v_i}$ el único color disponible. Finalmente, como se argumentó anteriormente, ahora es posible extender esta coloración de 3 colores en cada subgrafo de cláusula de seis nodos, resultando en una coloración de 3 colores de todo G'.
En sentido contrario, supongamos que G' tiene una coloración de 3 colores. En esta coloración, cada nodo v_i recibe ya sea el color True o el color False; establecemos la variable x_i en consecuencia. Ahora afirmamos que en cada cláusula de la instancia de 3-SAT, al menos uno de los términos en la cláusula tiene el valor de verdad 1. Porque si no, entonces los tres nodos correspondientes tienen el color False en la coloración de 3 colores de G' y, como hemos visto anteriormente, no hay una coloración de 3 colores del subgrafo correspondiente consistente con esto, por lo que hemos llegado a una contradicción.

3.3. Demostración de que hallar el número cromático es NP-Completo

Entonces, el problema de hallar el número cromático de un grafo es NP-Completo porque a partir de él podemos determinar si la instancia de 3-SAT es satisfacible: Por la demostración anterior, si el número cromático del grafo G' es 3, entonces la fórmula es satisfacible, y en caso contrario no lo es.

4. Retroalimentación de Vértices

Primero vamos a probar que el problema de retroalimentación de vértices de tamaño k es NP-Completo y, a partir de esto, que el problema de Retroalimentación de Vértices es NP-Hard.

4.1. Demostración de que retroalimentación de vértices de tamaño k es NP

Dada una instancia de retroalimentación de vértices y un conjunto solución D, bastaría verificar que el tamaño de D es menor o igual que k y que al eliminar esos vértices del grafo, este es acíclico, lo cual se puede hacer haciendo un recorrido por el grafo.

4.2. Demostración de NP-Hard usando reducción de Vertex Cover a retroalimentación de vértices de tamaño k

Dada una instancia de Vertex Cover con $G = (V, E)$ crearemos un grafo dirigido $G' = (V', E')$ con $V' = V$ y por cada arista $(a, b) \in E$ crearemos las aristas (a, b) y (b, a) en E', generando un ciclo en G'.
Para Vertex Cover necesitamos un conjunto mínimo tal que cada arista es cubierta, para retroalimentación de vértices necesitamos un conjunto de vértices mínimo tal que si lo eliminamos del grafo, resulta en un grafo sin ciclos.
Dado que cada arista en G de Vertex Cover se transforma en dos aristas dirigidas en G' formando un ciclo, el conjunto seleccionado para satisfacer retroalimentación de vértices, dígase F, debe incluir al menos uno de los dos vértices para romper ese ciclo.
Si tenemos entonces F en G, entonces F debe cubrir cada arista en G, eso significa que para cada ciclo dirigido correspondiente en G', F debe contener al menos uno de los vértices de ese ciclo para romperlo. Al eliminar F en G', todos los ciclos dirigidos se eliminan, lo que hace que G' sea acíclico. Por tanto, si en Retroalimentación de Vértices hay una solución de tamaño k, entonces también la hay en Vertex Cover.
Habiendo demostrado que este problema es NP-Completo, podemos reducir el mismo al de hallar el mínimo conjunto que satisfaga retroalimentación de vértices, pues si el mínimo es menor que k entonces hay una solución de longitud k y si no lo es, pues no la hay. Por tanto, el problema de hallar el mínimo conjunto que satisfaga Retroalimentación de Vértices es NP-Hard.

5. Problema Retroalimentación de Arcos

Al igual que en el problema anterior, demostraremos primero que el problema de encontrar un conjunto de arcos de longitud k tal que al quitarlos no queden ciclos en G es NP-Completo. Para ello intentaremos reducir el problema de Retroalimentación de Vértices a este en tiempo polinomial.

5.1. Demostración de NP-Completitud usando reducción de Retroalimentación de Vértices a Retroalimentación de Arcos

Dada una instancia (G, k) de Retroalimentación de Vértices con $G = (V, E)$, crearemos una instancia (G', k') de Retroalimentación de arcos de la siguiente forma: Por cada nodo $v \in V$ creamos dos nodos v_1 y v_2 en G' y el arco (v_1, v_2), a los que llamaremos arcos internos. Luego, por cada arco $(u, v) \in E$ crearemos en E' el arco (u_2, v_1), que serán los arcos externos.
Cada vértice en G se duplica en G', creando dos copias de vértices y las aristas del grafo original se dividen en aristas internas (entre las copias de un mismo vértice) y aristas externas (entre copias de vértices diferentes).
Al resolver Retroalimentación de Arcos en G' buscamos un conjunto de aristas que rompa todos los ciclos en G'. Todos los ciclos que involucran aristas externas

también pasan por aristas internas, por lo que eliminar solo aristas internas (que corresponden a vértices en G) es suficiente para romper los ciclos.
Por lo tanto, cualquier conjunto de aristas internas que forme una solución para Retroalimentación de Arcos en G' corresponde a un conjunto de vértices que forman una solución para Retroalimentación de Vértices en G. De esta forma, resolver Retroalimentación de Arcos en G' es equivalente a resolver Retroalimentación de Vértices en G, ya que los ciclos que rompes en G' equivalen a los ciclos que rompes en G.
Análogamente, como en el problema anterior, demostramos que hallar el mínimo conjunto es NP-Hard, pues en este problema también sirve para demostrar que hallar el mínimo conjunto de aristas es NP-Hard.

6. 3D-Matching

6.1. Demostración de que 3D Matching es NP

Dada una colección de tríos $T \subseteq X \times Y \times Z$, un certificado de que hay una solución podría ser una colección de tríos $T' \subseteq T$. En tiempo polinómico, se podría verificar que cada elemento en $X \cup Y \cup Z$ pertenece exactamente a uno de los tríos en T'.

6.2. Demostración de NP-Completitud usando reducción de 3-SAT a 3D-Matching

Consideremos una instancia de 3-SAT, con n variables $x_1, \ldots, x_n$ y k cláusulas $C_1, \ldots, C_k$. Primero diseñaremos componentes que codifiquen las elecciones independientes involucradas en la asignación de verdad a cada variable; luego agregaremos componentes que codifiquen las restricciones impuestas por las cláusulas. Al realizar esta construcción, inicialmente describiremos todos los elementos en la instancia de 3D-Matching simplemente como elementos, sin tratar de especificar para cada uno si proviene de X, Y, o Z. Al final, observaremos que naturalmente se descomponen en estos tres conjuntos.
Aquí está el componente básico asociado con la variable x_i. Definimos elementos $A_i = \{a_{i1}, a_{i2}, \ldots, a_{i2k}\}$ que constituyen el núcleo del componente; definimos elementos $B_i = \{b_{i1}, \ldots, b_{i2k}\}$ en las puntas del componente. Para cada $j = 1, 2, \ldots, 2k$, definimos un trío $t_{ij} = (a_{ij}, a_{i,j+1}, b_{ij})$, donde interpretamos la adición módulo $2k$. En el componente i, llamaremos a un trío t_{ij} par si j es par, e impar si j es impar. De manera análoga, nos referiremos a una punta b_{ij} como par o impar.
Estos serán los únicos tríos que contienen los elementos en A_i, por lo que ya podemos decir algo sobre cómo deben cubrirse en cualquier matching perfecto: debemos usar todos los tríos pares en el componente i, o todos los tríos impares en el componente i. Esta será nuestra forma básica de codificar la idea de que x_i puede establecerse en 0 o 1; si seleccionamos todos los tríos pares, esto representará establecer $x_i = 0$, y si seleccionamos todos los tríos impares, esto representará establecer $x_i = 1$.
Si decidimos usar los tríos pares, cubrimos las puntas pares del componente y dejamos libres las puntas impares. Si decidimos usar los tríos impares, cubrimos las puntas impares del componente y dejamos libres las puntas pares. Así, nuestra decisión de cómo establecer x_i se puede ver de la siguiente manera: dejar las puntas impares libres corresponde a 0, mientras que dejar las puntas pares libres corresponde a 1.
Hasta ahora, podemos hacer esta elección par/impar de forma independiente para cada uno de los n componentes de variables. Ahora añadimos elementos para

modelar las cláusulas y para restringir las asignaciones que podemos elegir. Consideremos el ejemplo de una cláusula:

$$C_1 = x_1 \vee \overline{x_2} \vee x_3$$

En el lenguaje de 3D-Matching, representa que la coincidencia en los núcleos de los componentes debería dejar libres las puntas pares del primer componente; o debería dejar libres las puntas impares del segundo componente; o debería dejar libres las puntas pares del tercer componente. Entonces, añadimos un componente de cláusula que hace precisamente esto.

Consiste en un conjunto de dos elementos centrales $P_1 = \{p_1, p_1'\}$, y tres tríos que los contienen. Uno tiene la forma (p_1, p_1', b_{1j}) para una punta par b_{1j}; otro incluye p_1, p_1', y una punta impar $b_{2,j'}$; y un tercero incluye p_1, p_1', y una punta par $b_{3,j''}$. Estos son los únicos tres tríos que cubren P_1, por lo que sabemos que uno de ellos debe ser utilizado; esto impone la restricción de la cláusula.

En general, para la cláusula C_j, creamos un componente con dos elementos centrales $P_j = \{p_j, p_j'\}$, y definimos tres tríos que contienen a P_j de la siguiente manera. Supongamos que la cláusula C_j contiene un término t. Si $t = x_i$, definimos un trío $(p_j, p_j', b_{i,2j})$; si $t = \overline{x_i}$, definimos un trío $(p_j, p_j', b_{i,2j-1})$. Nota que solo el componente de cláusula j utiliza las puntas b_{im} con $m = 2j$ o $m = 2j - 1$; por lo tanto, los componentes de las cláusulas nunca competirán entre sí por las puntas libres.

Casi hemos terminado con la construcción, pero todavía hay un problema. Supongamos que el conjunto de cláusulas tiene una asignación satisfactoria. Entonces hacemos las elecciones correspondientes de par/impar para cada componente de variable; esto deja al menos una punta libre para cada componente de cláusula, y así todos los elementos centrales de los componentes de las cláusulas también se cubren. El problema es que no hemos cubierto todas las puntas. Comenzamos con $n \cdot 2k = 2nk$ puntas; los tríos $\{t_{ij}\}$ cubrieron nk de ellas; y los componentes de las cláusulas cubrieron k adicionales. Esto deja $(n-1)k$ puntas por cubrir.

Manejamos este problema con un truco muy simple: añadimos $(n-1)k$ componentes de limpieza a la construcción. El componente de limpieza i consta de dos elementos centrales $Q_i = \{q_i, q_i'\}$, y hay un trío (q_i, q_i', b) para cada punta b en cada componente de variable.
Por lo tanto, si el conjunto de cláusulas tiene una asignación satisfactoria, entonces hacemos las elecciones correspondientes de par/impar para cada componente de variable; como antes, esto deja al menos una punta libre para cada componente de cláusula. Usando los componentes de limpieza para cubrir las puntas restantes, vemos que todos los elementos centrales en los componentes de variable, cláusula y limpieza han sido cubiertos, y todas las puntas también han sido cubiertas.
Por el contrario, supongamos que hay una coincidencia tridimensional perfecta en la instancia que hemos construido. Entonces, como argumentamos anteriormente, en cada componente de variable la coincidencia elige todos los $\{t_{ij}\}$ pares o todos los $\{t_{ij}\}$ impares. En el primer caso, establecemos $x_i = 0$ en la instancia de 3-SAT; y en el segundo caso, establecemos $x_i = 1$. Ahora considere la cláusula C_j; ¿ha sido satisfecha? Debido a que los dos elementos centrales en P_j han sido cubiertos, al menos uno de los tres componentes de variable correspondientes a un término en C_j tomó la decisión correcta de par/impar, y esto induce una asignación de variable que satisface C_j.
Esto concluye la prueba, excepto por una última cosa de qué preocuparse: ¿realmente hemos construido una instancia de Coincidencia Tridimensional? Tenemos

una colección de elementos y tríos que contienen algunos de ellos, pero ¿pueden los elementos realmente ser divididos en los conjuntos apropiados X, Y y Z de igual tamaño?

Afortunadamente, la respuesta es sí. Podemos definir X como el conjunto de todos los a_{ij} con j par, el conjunto de todos los p_j y el conjunto de todos los q_i. Podemos definir Y como el conjunto de todos los a_{ij} con j impar, el conjunto de todos los p'_j y el conjunto de todos los q'_i. Finalmente, podemos definir Z como el conjunto de todas las puntas b_{ij}. Ahora es fácil verificar que cada trío consiste en un elemento de cada uno de los conjuntos X, Y y Z.