FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Teoría de Algoritmos 1 [75.29/95.06]

TP3

Gonzalo Sabatino	104609
Maria Paula Bruck	107533
Kevin Javier Torrez Maldonado	101698
Mateo Capón Blanquer	104258

Índice general

Parte 1

Una campaña publicitaria masiva pero mínima

Enunciado Una empresa de turismo que vende excursiones desea realizar una campaña publicitaria en diferentes vuelos comerciales con el objetivo de llegar a todos los viajeros que parten del país A y que se dirigen al país B. Estos viajeros utilizan diferentes rutas (algunos vuelos directos, otros armando sus propios recorridos intermedios). Se conoce para una semana determinada todos los vuelos entre los diferentes aeropuertos con sus diferentes capacidades. Además parten del supuesto que durante ese periodo la afluencia entre A y B no se verá disminuida por viajes entre otros destinos.

Desean determinar en qué trayectos simples (trayecto de un viaje que inicia desde un aeropuerto y termina en otro) poner publicidad de forma de alcanzar a TODAS las personas que tienen el destino inicial A y el destino final B. Pero además desean que siempre que sea posible seleccionen la combinación que tenga el menor número de vuelos comerciales. Esto es porque pagan tanto por cantidad de vuelos como por pasajeros que cumplan con la condición de ser del país de origen A y con destino final B.

Se pide:

1. Proponer una solución algorítmica que resuelva el problema de forma eficiente. Explicarla paso a paso. Utilice diagramas para representarla.

2. Plantear la solución como si fuese una reducción de problema. ¿Puede afirmar que corresponde a una reducción polinomial? Justificar.

3. ¿Podría asegurar que su solución es óptima?

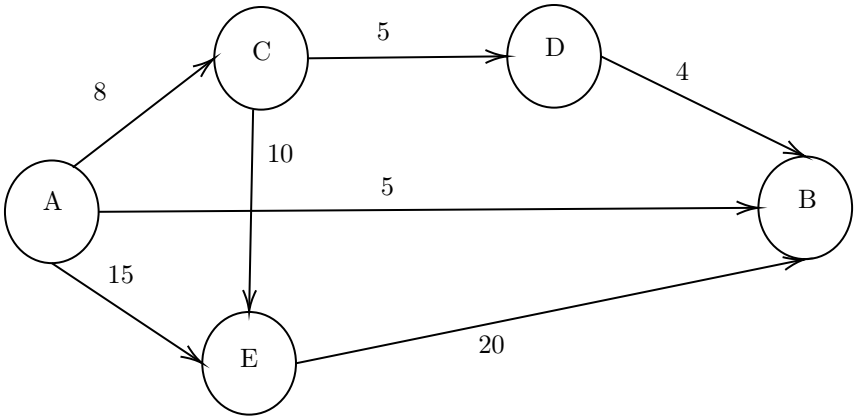
4. Programe la solución

5. Compare la complejidad temporal y espacial de su solución programada con la teórica. ¿Es la misma o difiere?

1.1. Ítem 1

El enunciado consta de resolver dos problemas principales. Por un lado, se pide la cantidad máxima de pasajeros que pueden ir de A a B. Esto es equivalente a un problema de maximización de flujo, en el que los nodos representan a las ciudades, y los ejes, a la capacidad máxima de pasajeros que acepta el transporte entre ambos aeropuertos. Por otro lado, se busca el conjunto mínimo de vuelos, de modo tal que para cualquier camino de A a B, se deba pasar por uno de estos vuelos. Se pondrán las publicidades en estos vuelos, y en consecuencia, todos los viajantes de A hacia B, las verán. Este segundo problema, está muy relacionado con el primero. A continuación exponemos una posible resolución de los mismos.

Utilizaremos como ejemplo el grafo de la siguiente figura, el cual representa a los distintos vuelos con sus capacidades.

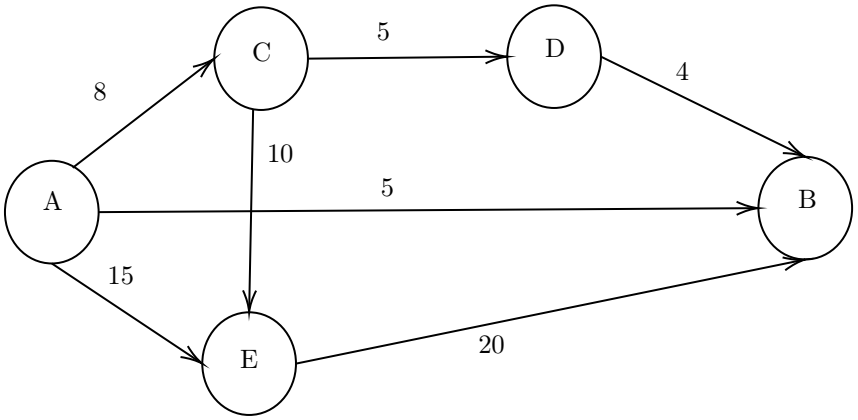


En función de resolver el primer problema, utilizaremos el algoritmo de Edmond Karps para resolver la Maximización de Flujos. De este modo, obtendremos la cantidad máxima de pasajeros que podrá haber desde la ciudad A hasta la B.

El segundo problema se resuelve de una manera muy similar. Dado el mismo grafo dirigido de la figura, modificamos todos los pesos $w_{ij} = 1$ y aplicamos Edmond Karps sobre el nuevo grafo. Dicho algoritmo busca el flujo máximo desde A hasta B, dado que todos los ejes tienen capacidad 1. Se muestra en la teórica que el flujo máximo es exactamente igual a la suma de las capacidades del corte mínimo. Pero como todas las capacidades valen 1, la suma de las mismas va a coincidir con la cantidad de aristas del corte. En consecuencia, al aplicar el algoritmo, se obtendrá el corte que tenga la menor cantidad de aristas. Este conjunto de ejes se corresponde con los vuelos comerciales a los cuales se les pondrá la publicidad.

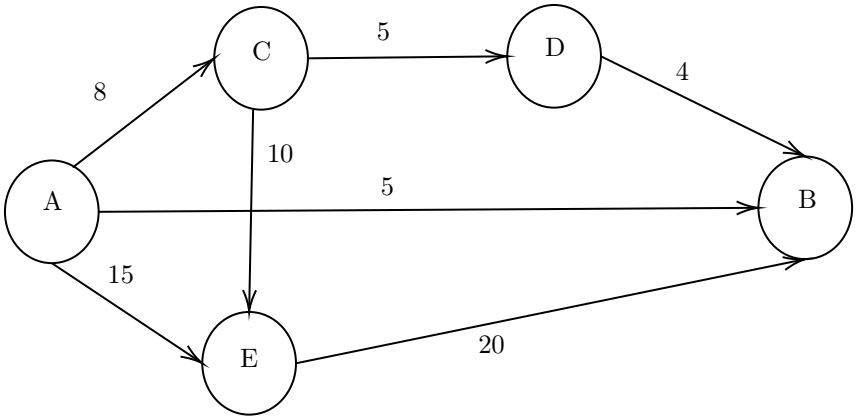
Aplicación

En este caso, se utiliza el ejemplo previamente mencionado para transformar el problema a una Maximización de Flujos. Resolvemos primero el problema de buscar la cantidad de pasajeros máximo que se pueden trasladar de A hacia B.



Las capacidades de los ejes son la capacidad de pasajeros que pueden viajar desde un aeropuerto de destino a uno de origen.

Se transforma el grafo a un Grafo residual G_f.

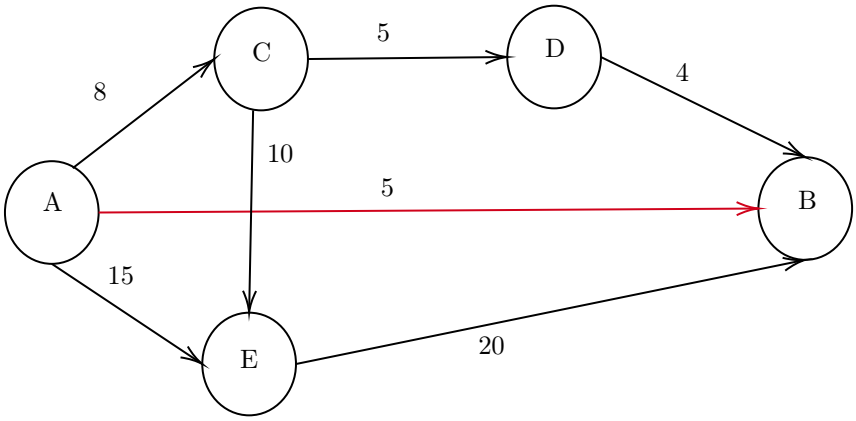


Se aprecia que G_f es exactamente igual al grafo original en la primera iteración.

1.1.1. Resolución mediante Edmond Karps

Utilizando G_f obtenido en la primera parte de este algoritmo, se ejecuta el algoritmo.

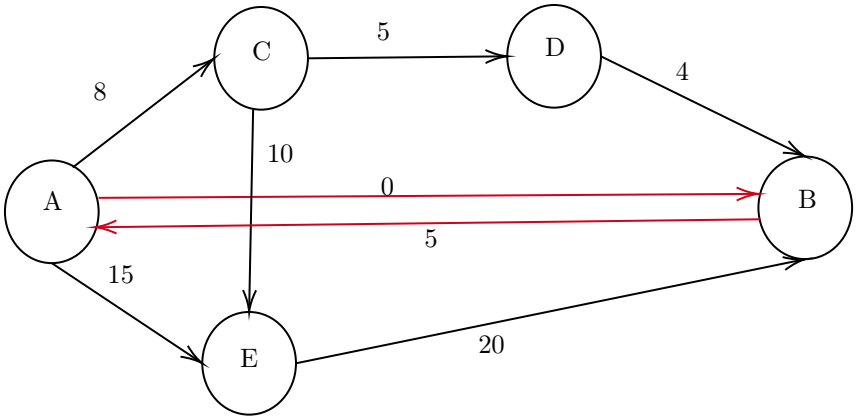
Se realiza un BFS. En este caso es trivial, porque existe un camino de distancia 1 desde A hacia B.



Se encuentra el cuello de botella, que en este caso es 5. Se guarda el camino mínimo junto con su cuello de botella.

Camino mínimo	Cuello de Botella
A-B	5

Para los ejes dentro del camino mínimo (en este caso solo está $A - B$) se resta hacia adelante y se suma hacia atrás el cuello de botella encontrado.

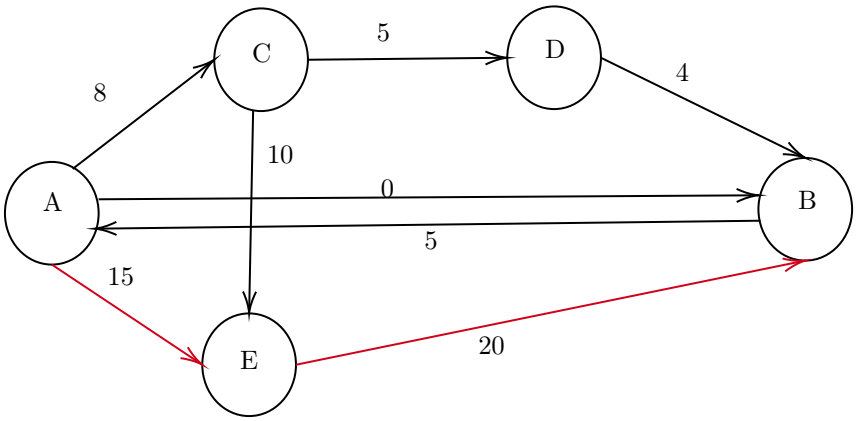


Se guarda el flujo total hasta ahora (suma de todos los cuellos de botella obtenidos):

$$Flujoactual = 5$$

.

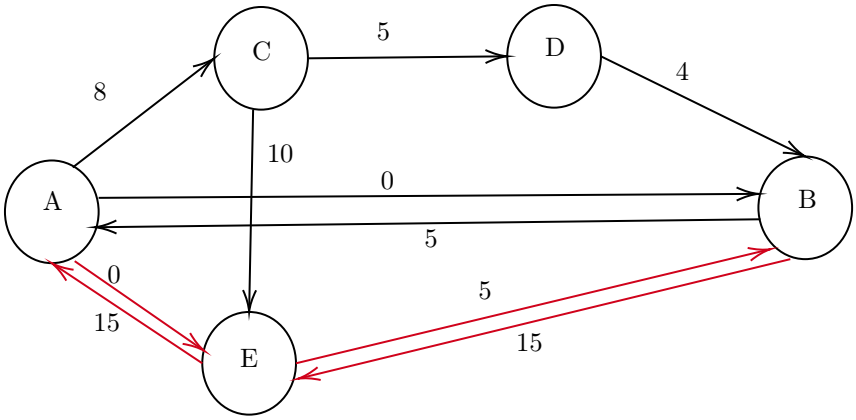
Se realiza un BFS. Desde A, se puede ir hasta C y E. Desde C, se puede ir a D. Entonces $d(A, D) = 2$. Además, se puede ir a E, pero con distancia $2 > d(A, E) = 1$. Desde E, se puede ir hasta B, quedando $d(A, B) = 2$. Se decide entonces este camino.



Se encuentra el cuello de botella, que en este caso es 15. Se guarda el camino mínimo junto con su cuello de botella.

Camino mínimo	Cuello de Botella
A-B	5
A-E-B	15

Para los ejes dentro del camino mínimo se resta hacia adelante y se suma hacia atrás el cuello de botella encontrado.

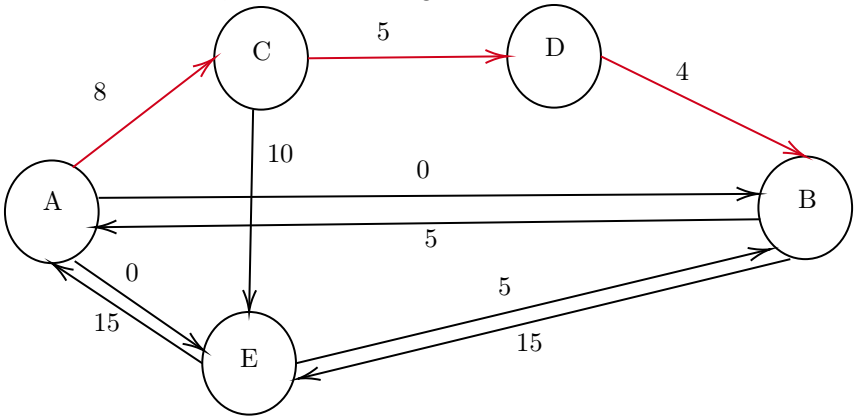


Se guarda el flujo total hasta ahora (suma de todos los cuellos de botella obtenidos):

$$Flujoactual = 20$$

.

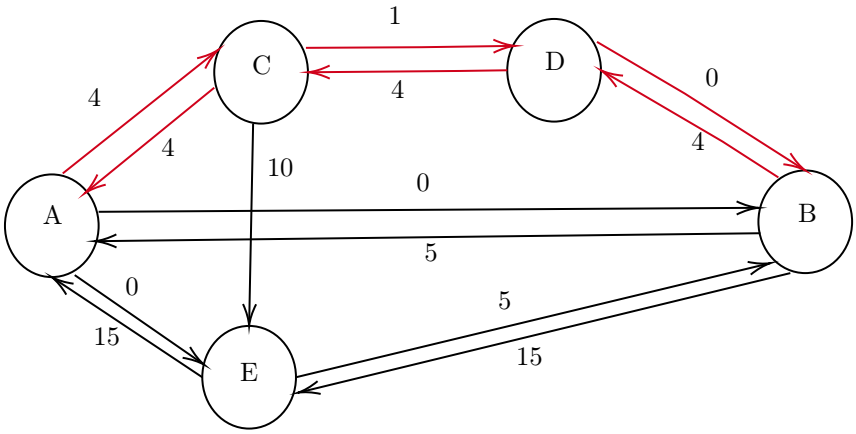
Se realiza un BFS, obteniendo el siguiente camino mínimo.



Se encuentra el cuello de botella, que en este caso es 4. Se guarda el camino mínimo junto con su cuello de botella.

Camino mínimo	Cuello de Botella
A-B	5
A-E-B	15
A-C-D-B	4

Para los ejes dentro del camino mínimo se resta hacia adelante y se suma hacia atrás el cuello de botella encontrado.

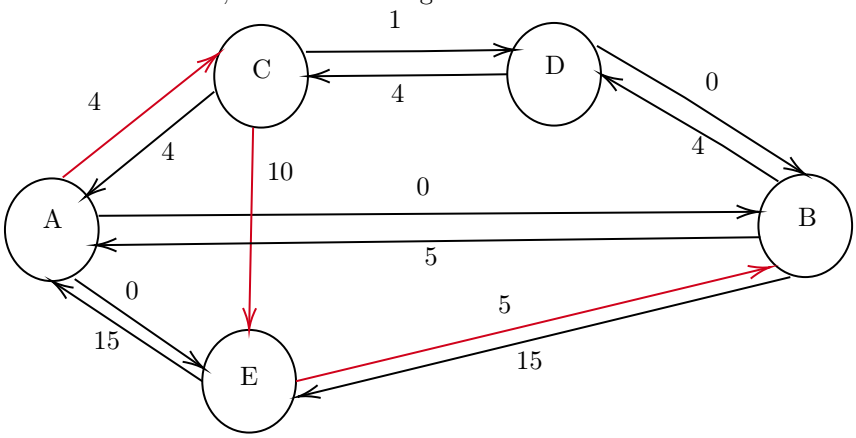


Se guarda el flujo total hasta ahora (suma de todos los cuellos de botella obtenidos):

$$Flujoactual = 24$$

.

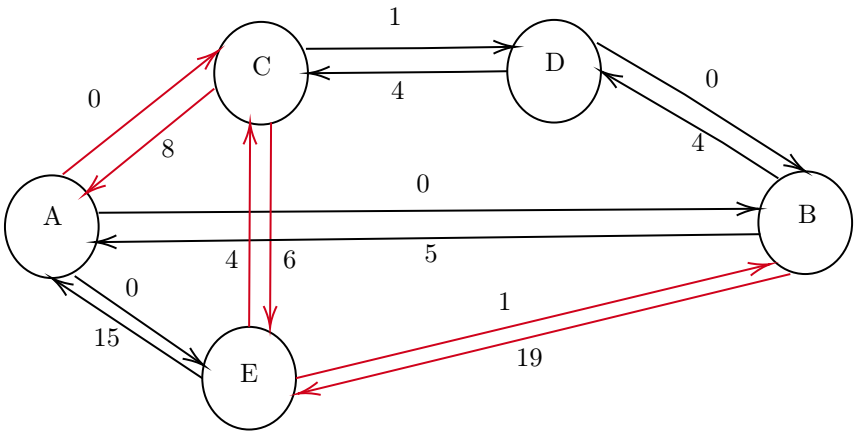
Se realiza un BFS, obteniendo el siguiente camino mínimo.



Se encuentra el cuello de botella, que en este caso es 4. Se guarda el camino mínimo junto con su cuello de botella.

Camino mínimo	Cuello de Botella
A-B	5
A-E-B	15
A-C-D-B	4
A-C-E-B	4

Para los ejes dentro del camino mínimo se resta hacia adelante y se suma hacia atrás el cuello de botella encontrado.



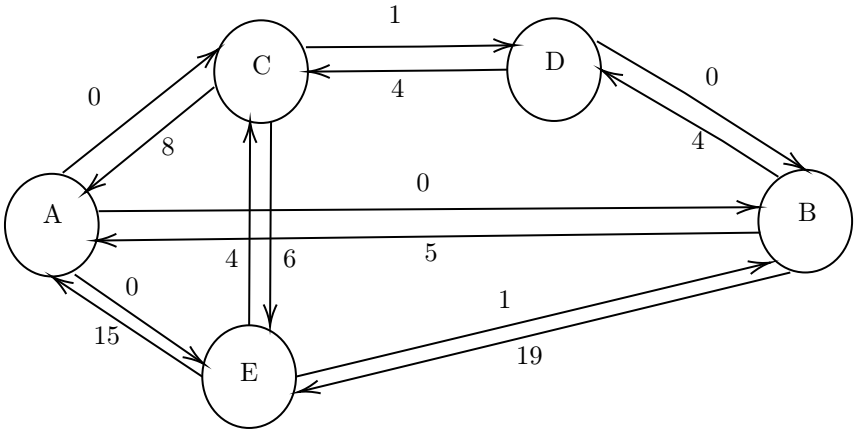
Se guarda el flujo total hasta ahora (suma de todos los cuellos de botella obtenidos):

$$Flujoactual = 28$$

.

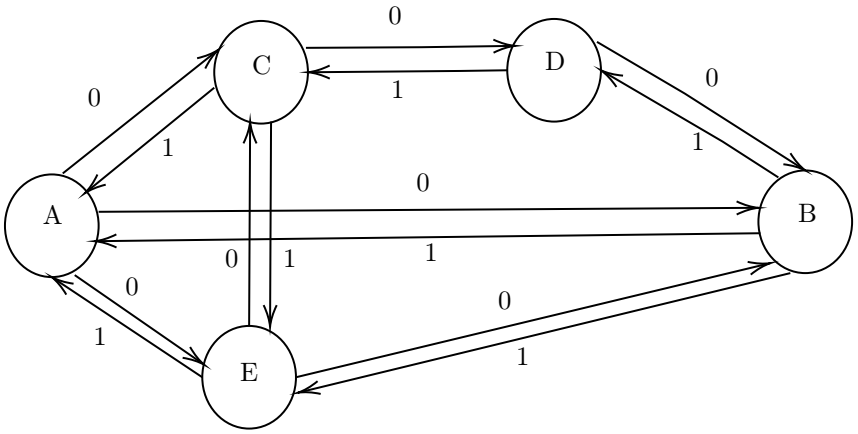
Grafo residual final

Se muestra el grafo residual final obtenido por Edmond Karps.



En el grafo residual, no se puede ir a ningún nodo partiendo desde A (todos los ejes salientes de A tienen capacidad 0). Por lo tanto el corte mínimo viene dado por $A - C, A - B$ y $A - E$. Además, y más importante, el flujo máximo es de 28. Esto es, habrán a lo sumo 28 pasajeros desde el aeropuerto A hasta el B.

Por el otro lado, resolver el mismo problema con las capacidades en 1, es muy similar. Luego de las iteraciones del algoritmo, queda el siguiente grafo residual.



Además, el flujo máximo, que es lo mismo que la cantidad de aristas mínimas del corte, es de 3. Y la publicidad se pondrá en $A - C, A - B$ y $A - E$.

1.2. Ítem 2

El primer paso es cambiar el dominio del problema a un problema de decisión, por ende, lo que se plantea decidir si el flujo máximo se encuentra entre dos límites dados.

Si se considera tanto que Edmond Karps como Ford Fulkerson tienen complejidades polinomiales (todo dependerá de sus vértices, nodos y capacidades), entonces podemos afirmar que: **Edmond Karps puede ser reducido en tiempo polinomial por Ford Fulkerson**, puesto que Edmond Karps es (a lo sumo), tan difícil de resolver como lo es Ford Fulkerson. Las transformaciones del problema de Edmond Karps a Ford Fulkerson son polinomiales (particularmente, no se necesita ninguna transformación, ya que ambos parten de las mismas condiciones iniciales). Además, tenemos garantizado que Ford Fulkerson devolverá la misma respuesta que Edmond Karps, puesto que el segundo es simplemente una optimización del primero.

Y como tenemos dos subproblemas que usan Edmond Karps, utilizamos dos transformaciones: Para la primera se puede proponer cualquier constante en ambos límites, para la segunda, el flujo máximo equivale a todos los ejes que se quitan para que el conjunto que contiene al nodo fuente sea desconexo respecto del conjunto que contiene al nodo sumidero, por lo tanto equivaldrá a la cantidad de anuncios que se pondrán, entonces se pueden proponer dos límites teniendo en cuenta que será la cantidad de publicidades a poner.

1.3. Ítem 3

El problema lo dividimos en dos subproblemas:

- Obtener el flujo máximo de pasajeros que viajan de A a B
- Obtener en qué recorridos entre dos aeropuertos poner publicidades.

Para el primer problema, utilizamos un algoritmo de maximización de flujos (Edmond Karps), que ya tiene demostrada su optimalidad para este tipo de problemas. El presente problema se corresponde con el problema de maximizar flujos, ya que se puede representar a cada aeropuerto como un nodo; y a cada eje, como un vuelo que tiene una capacidad de pasajeros máxima.

Para el segundo problema, al utilizar un grafo equivalente al inicial, pero con todas las capacidades de sus ejes siendo 1, para demostrar optimalidad se tiene que demostrar que el corte mínimo contiene la menor cantidad de aristas para ir del conjunto A (que contiene al nodo fuente) al conjunto B (que contiene al nodo sumidero).

En primer lugar, al hacer un corte en el grafo, por definición se tienen dos conjuntos disjuntos. Si se quitan las aristas pertenecientes a un corte, tal que en un conjunto está A, y en el otro está B, no habrá ningún camino que una A y B. Dicho de otra forma, si no se quitan estás aristas, todos los caminos que van desde A hacia B, pasarán por al menos una de estas. Mostramos entonces, que al elegir un corte del grafo y al poner las publicidades sobre los vuelos pertenecientes al mismo, todos los pasajeros que van de A a B, recibirán la publicidad.

En segundo lugar, debemos probar que se ponen la publicidad en la menor cantidad de vuelos posibles utilizando el algoritmo presentado. Tanto el algoritmo de Edmond Karps, como el de Ford Fulkerson, resuelven el problema de flujo máximo. Está demostrado (no presentamos la prueba) que el flujo máximo entre dos nodos A y B es igual a la sumatoria de las capacidades del corte mínimo. Pero en este caso, la sumatoria de las capacidades del corte equivale a la cantidad de aristas (vuelos) que hay en el corte. Por lo tanto, el flujo máximo que entregue el algoritmo, será la mínima cantidad de vuelos en los que hay que poner la publicidad. Siendo que además, utilizando estos algoritmos, se puede identificar cuál es el corte, se tendrá el conjunto de vuelos en los que se debe poner la publicidad.

1.4. Ítem 4

1.4.1. Pseudo código del algoritmo

El pseudo-código de la solución es el siguiente:

```
procedure AIRPORT ADVERTISING(graph)
    Generar new_graph a partir de graph, con pesos en 1
    max_flow = edmond_karps(graph)
    edmond_karps(new_graph)
    Obtener el corte mínimo con new_graph modificado por Edmond Karps
    for Every edge in new_graph do
        if edge no está en min_cut y edge.node_origin está en min_cut then
            Colocar una publicidad en ese eje
        end if
    end for
    return max_flow
end procedure
```

Donde se muestran ambas iteraciones con Edmond Karps. A continuación, se presente el pseudo-código de Edmond Karps

```
procedure EDMOND KARPS(graph)
    Inicialmente el grafo residual G_f equivale a graph
    max_flow = 0
    while Existe camino en G_f do
        Path = BFS(G_f)
        flow = augment(G_f, path)
        Actualizar G_f
        Sumar flow en max_flow
    end while
    return max_flow
end procedure
```

```
procedure AUGMENT(G_f, path)
    Sea bottleneck = bottleneck(path)
    for Every edge in path) do
        if edge hacia adelante then
            f(e)-=b en G_f
        end if
        if edge hacia atrás then
            f(e)+=b en G_f
        end if
    end for
    return bottleneck
end procedure
```

1.4.2. Cómo probar el archivo

A continuación, figuran todos los pasos para poder ejecutar y probar el código entregado. La línea de ejecución es:

```
python publicidades_aeropuertos.py ruta_archivo/archivo.txt
```

Donde *ruta_archivo* se puede omitir si el txt se encuentra dentro del mismo directorio que el código a ejecutar.

1.5. Ítem 5

1.5.1. Complejidades teóricas

Como complejidad temporal, el mayor peso del algoritmo presentado se lo llevan ambas iteraciones de Edmond Karps, por lo tanto, la complejidad temporal teórica será la misma que dicho algoritmo de maximización de flujos, es decir, $\mathcal{O}(V \cdot E^2)$.

La complejidad espacial será, puesto que solo hay que guardar dos grafos en memoria (el original y el residual), la complejidad espacial teórica es $\mathcal{O}(V + E)$.

1.5.2. Complejidades reales

Para la complejidad temporal, la implementación utiliza punteros entre nodos y ejes. Se puede apreciar que la mayor complejidad la tiene Edmond Karps también(ver comentarios en código para mayor profundidad) con la misma complejidad teórica ($\mathcal{O}(V \cdot E^2)$).

Para la complejidad espacial, utilizamos un grafo con V nodos. Cada nodo tiene un puntero a E ejes. Cada eje tiene un nodo de origen y un nodo de destino. Por ende, la complejidad espacial real de un grafo en nuestra implementación es $\mathcal{O}(V \cdot E)$. Además, guardamos caminos de aumento, donde cada uno tiene E ejes, por lo tanto su complejidad espacial es $\mathcal{O}(E)$. Es decir que la complejidad espacial real es $\mathcal{O}(V \cdot E)$, que difiere de la teórica.

Parte 2

Equipos de socorro

Enunciado

El sistema ferroviario de un país cubre un gran conjunto de su territorio. El mismo permite realizar diferentes viajes con transbordos entre distintos ramales y subramales que pasan por sus principales ciudades. Dentro de su proceso de mejoramiento del servicio buscan que ante una emergencia en una estación se pueda llegar de forma veloz y eficiente. Consideran que eso se lograría si el equipo de socorro se encuentra en esa misma estación o en el peor de los casos en una estación vecina (que tenga una trayecto directo que no requiere pasar por otras estaciones). Como los recursos son escasos desean establecer la menor cantidad de equipos posibles (un máximo de k equipos pueden solventar). Se solicita nuestra colaboración para dar con una respuesta a este problema.

1. Utilizar el problema conocido como "set dominante" para demostrar que corresponde a un problema NP-Completo.

2. Asimismo demostrar que el problema set dominante corresponde a un problema NP-Completo.

3. Con lo que demostró responda: ¿Es posible resolver de forma eficiente (de forma "tratable") el problema planteado?

HINT: podría ser una buena idea utilizar 3SAT o VERTEX COVER.

2.1. Item 1

NP-Completos

Un problema B en NP es NP-completo si, para cualquier problema A en NP, se puede reducir A a B en tiempo polinómico. Dicho de otra forma, un problema $B \in NP$ es NP-completo si $\forall A \in NP$, $A \leq B$. Que un problema B sea NP-Completo quiere decir que podemos traducir las instancias de cualquier problema $A \in NP$ a instancias de B, de forma que, si pudiésemos resolver B, también podríamos resolver A. Esto es, B es tan general que es capaz de expresar las restricciones y objetivos de cualquier otro problema de NP.

Si podemos demostrar que cualquier problema NP-completo está en P, entonces P sería igual a NP. De la misma forma, si se demostrase que algún problema NP-Completo no se puede resolver en tiempo polinómico, entonces el resto de los problemas NP-completos tampoco se podría resolver en tiempo polinómico.

Entonces, si podemos demostrar que un problema es NP-Completo, nos dedicaremos a encontrar una buena aproximación que a buscar una solución exacta. La idea que normalmente se usa

es ubicar el problema y luego probar que el otro problema es NP-completo tratando de reducirlo al ya conocido como NP-Completo.

¿Cómo podemos demostrar si este es un problema NP-Completo?

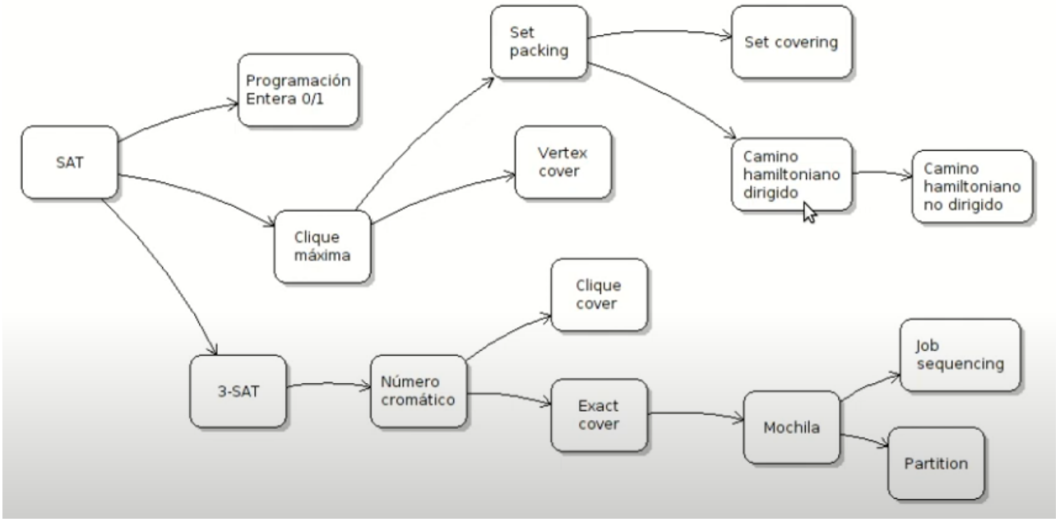
Veamos la definición:

Si un problema B pertenece NP es NP-Completo, si $A \leq pB$ para todo problema A perteneciente a NP, tenemos que:

$$A \leq pC \leq pB$$

Es decir, tomamos un problema conocido que sabemos que es NP-Completo, Viajante de Comercio (TSP) por ejemplo, y lo reducimos a nuestro problema, entonces para todo problema A que esta en NP, A reduce al problema C porque C es NP-Completo, por definición de NP-Completo, todo otro problema NP se reduce a él y ese problema reduce al nuestro, entonces podemos aplicar las transformaciones polimoniales una a continuación de la otra y reducir de A a B.
Entonces, al haber demostrado que tenemos una reducción polimonial de C a mi problema B demuestro que existe una reducción polimonial de cualquier otro problema en NP a mi problema B a través de C.

Grafico de Problemas NP-completos



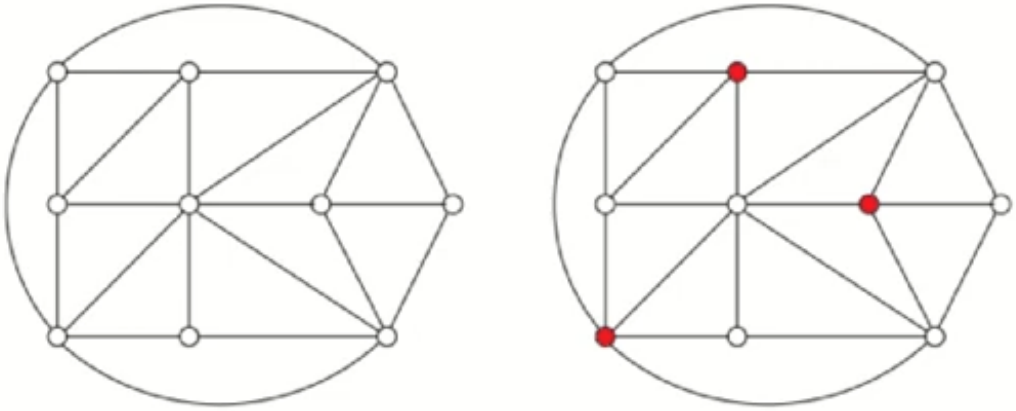
En este punto, tomaremos como Set Dominante (3-SAT) ya que sabemos que este es NP-Completo. Es decir, tomaremos un Problema de Lógica y lo transformaremos a un problema de grafos haciendo un una clase de construcción, de manera que podamos representar el grafo.
Reducción 3-SAT al conjunto independiente máximo.

1. Entrada: Fórmula Proporcional con 3 proposiciones por clausula.
2. Salida: ¿Existe una evaluación que haga verdadera la formula?.

Por ejemplo, la siguiente fórmula tiene no mas de 3 proposiciones por clausula:

$$(\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$$

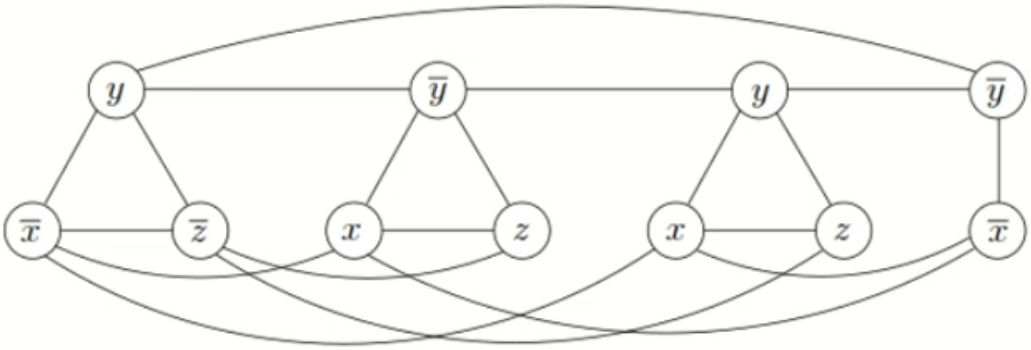
Entonces, vamos a reducir nuestro problema (Conjunto Independiente Máximo). Creando un conjunto independiente en un Grafo donde estos no sean vecinos con ningún otro vértice.



Conjunto Independiente Máximo.

- Entrada: Un Grafo G y un número k perteneciente a z+.
- Salida: ¿Existe un conjunto independiente en G de tamaño k o mayor?.

Entonces: Veamos que $3 - SAT \leq pMIS$, dada una instancia / de 3-SAT (es decir dada una Formula), construiremos un grafo del siguiente modo.



Para cada clausula pondremos un Clique, entonces en cada triangulo del grafo serán vecinos con todos, luego uniremos con una arista los vértices que corresponden a literales opuestos. a través de la formula construimos el grafo como a nosotros nos convenga, por lo tanto formaran un conjunto independiente de tamaño 4.

- Si la formula tiene C cláusulas (C = 4 en el ejemplo), entonces fijamos k = C.
- TEOREMA. La fórmula es satisfactible si y solo si el grafo tiene un conjunto independiente de tamaño k o superior.
- Esto muestra que $3 - SAT \leq pMIS$ (problema propuesto), como 3-SAT es NP-Completo entonces nuestro problema también es NP-completo.

2.2. Item 2

Primero a las definiciones

¿Qué es 3-SAT?

Es cierto problema de decisión, es decir, es un problema para el cual de cada instancia se produce un resultado binario - podríamos decir, existe una máquina de Turing que recibe un string inicial (determinado por el problema en particular; en el caso de 3-SAT, un string representando los valores binarios de las variables del circuito) y se detiene en "cierto" o "falso". Esto significa que primero, 3-SAT es miembro de la clase de problemas de decisión NP , i.e. que existe un algoritmo que, dada una solución para una instancia de 3-SAT, verifica la validez de esa solución en tiempo polinomial. Y segundo, significa que dado cualquier problema R en NP , poder resolver 3-SAT (es decir, poseer un oráculo que te resuelve 3-SAT inmediatamente) te permite resolver R en tiempo polinomial. El que 3-SAT sea un problema NP -completo significa que ningún otro problema en NP puede ser más "difícil", y que si se encuentra una solución polinomial para 3-SAT, se obtiene una solución polinomial para todo problema en NP ; es decir, se demuestra que $P = NP$. Como se cree que $P \neq NP$, se cree que no existe tal solución, ya que a nadie le ha salido ese problema.

Dados estas definiciones, podemos decir que. 3-SAT es NP-completo porque resulta que cada instancia de un problema arbitrario R en NP se puede codificar como una instancia de 3-SAT, y esta codificación se puede lograr en tiempo polinomial. Para ser más preciso, la demostración de que 3-SAT es NP -completo construye una máquina de Turing que traduce una instancia arbitraria del problema R a una instancia de 3-SAT; cuando se hace un análisis cuidadoso de cómo esta máquina de Turing corre, se obtiene que lo hace en tiempo polinomial. Y esa es la razón: 3-SAT es NP-completo porque puedes usarlo para "simular" cualquier otro problema en NP en tiempo polinomial; porque todo problema NP es 3-SAT disfrazado.

2.3. Item 3

Para resolver este problema podemos ver a las ciudades del sistema ferroviario como un grafo G=(V,E) en donde varias de ellas tienen ya asignados recorridos de los distintos ramales que las conectan.

Para poder cumplir con lo pedido que es, mejorar el servicio para que en caso de emergencia se pueda llegar de una estación a otra de manera rápida y fácil y con trayecto directo se puede usar el problema del conjunto independiente.

Y así encontrar la cantidad de nodos que son independientes , es decir que no exista a,b perteneciente a C tal que exista eje (a,b) perteneciente a E.

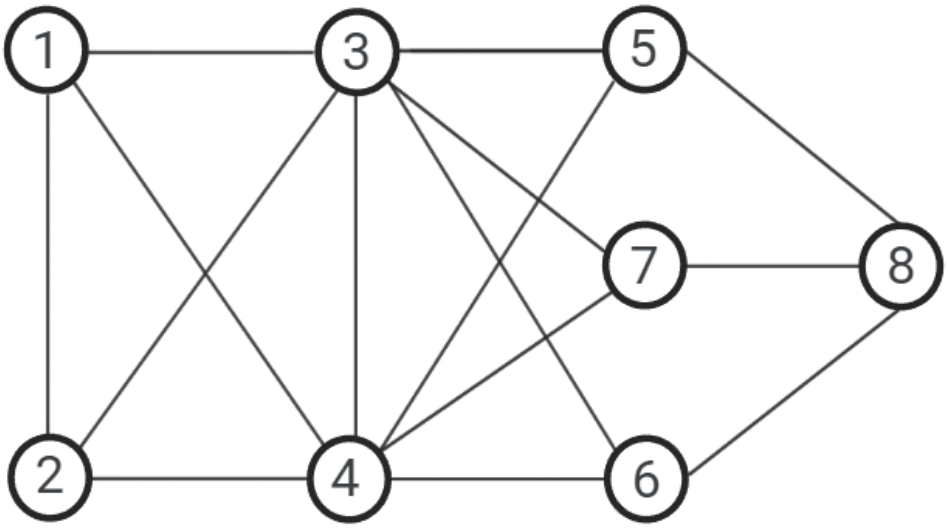
Para poder resolverlo hay que tener en cuenta que el conjunto independiente de nodos debe ser de tamaño k siendo k igual a la cantidad de equipos con los que se cuenta para dicha reforma .

De esta manera se podrá colocar un equipo de socorro en cada uno de estos nodos independientes y así asegurarse que los equipos de socorro estén distribuidos de manera que tengan un trayecto directo a las ciudades sin que requieran pasar por otras estaciones y por lo tanto garantizar la eficacia de la reforma solicitada.

¿Es posible resolver de forma eficiente el problema planteado?

Podemos resolver de forma particular este tipo de problema ya que tenemos información suficiente. Plantearemos el problema dado de la siguiente manera:

Tomaremos como ejemplo el siguiente Grafo G:



Buscaremos un Conjunto Independiente y que este sea el Máximo, como para poder determinar donde deben estar ubicados los equipos de socorro para cubrir que todas las ciudades en situaciones de emergencia puedan recibir al equipo de socorro mediante un trayecto directo.

Observemos que si tomamos el conjunto de vértices:

$$\{1; 5; 7; 6\}$$

El siguiente conjunto no es adyacente entre ellos así que podemos afirmar que es un Conjunto Independiente, y el tamaño de este conjunto es igual o mayor que 4.

$$\alpha(G) \geq 4$$

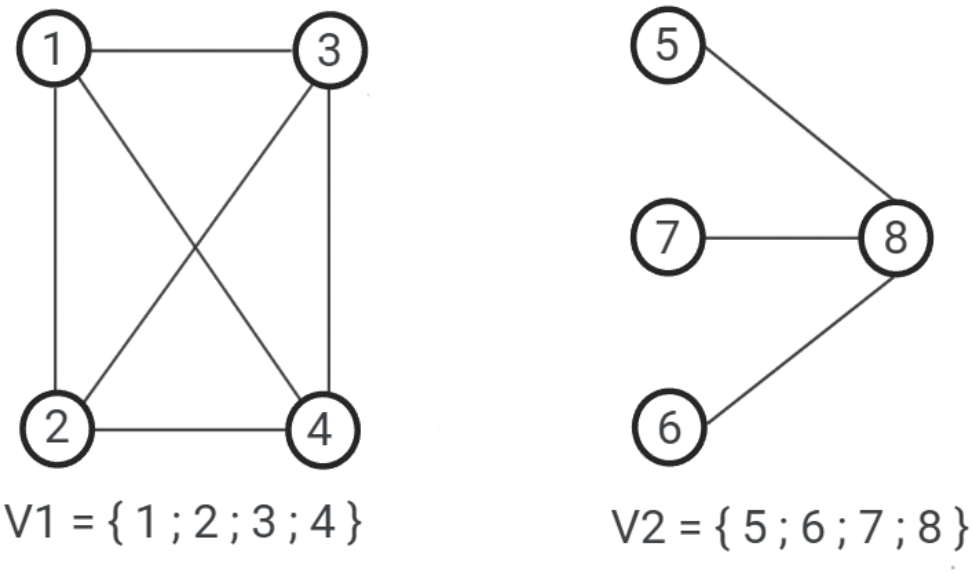
Para poder afirmar que ese Conjunto es el mayor, debemos probar que no existe otro conjunto mayor a el.
Para ello tomaremos dos subconjuntos de vértices:

$$V1 = \{1; 2; 3; 4\}$$

$$V2 = \{5; 6; 7; 8\}$$

Si observamos que V1 $\bigcup V2 = V(\mathrm{G})$

Entonces podemos formar los siguiente Subgrafos de G;



V1 es un SubGrafo k4. Por lo tanto podemos decir que el conjunto independiente es $\alpha\ (G1) = 1$

Este es un grafo completo y cada vértice del grafo es adyacente a las restantes vértices del mismo, entonces podemos decir que solo hay un vértice independiente.

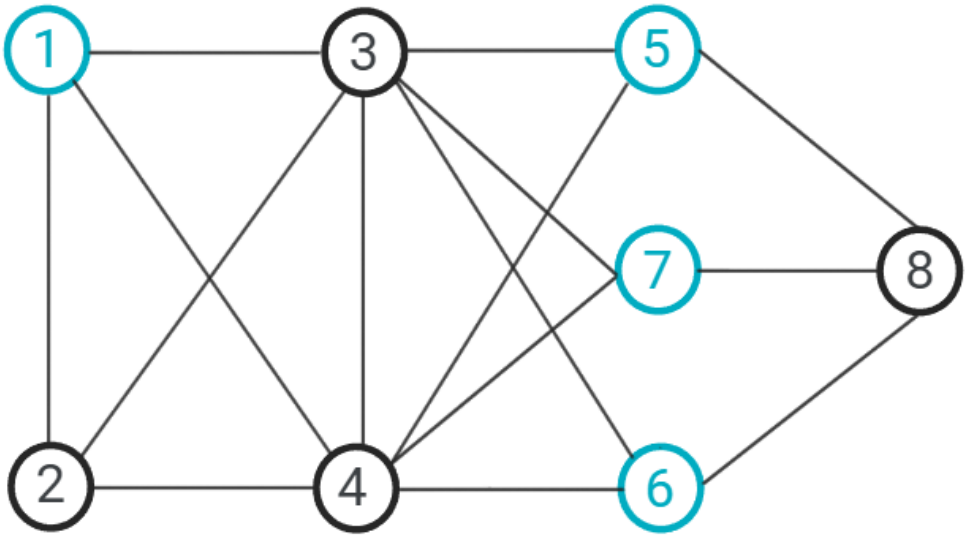
En el caso V2 es un Subgrafo k1,3. Es un Grafo tripartito y claramente el conjunto independiente es 3, entonces $\alpha\ (G2) = 3$

Por lo tanto:

$\alpha\ (G) \leq \alpha(G1) + \alpha(G2)$

Entonces podemos afirmar que $\alpha\ (G) \leq 4$.

Juntando las 2 condiciones podemos concluir que el tamaño máximo del Conjunto Independiente Máximo es 4 y ver en el ejemplo que se asegura la correcta distribución de los equipos de socorro convirtiéndose en una solución posible al problema planteado .



Conjunto Independiente Maxima = 4
Vertices = { 1 ; 5 ; 7 ; 6 }

Parte 3

Un poco de teoría

Enunciado

1. Defina y explique qué es una reducción polinomial y para qué se utiliza.
2. Explique detalladamente la importancia teórica de los problemas NP-Completos.
3. Tenemos un problema A, un problema B y una caja negra NA y NB que resuelven el problema A y B respectivamente. Sabiendo que B es P
   a) Qué podemos decir de A si utilizamos NA para resolver el problema B (asumimos que la reducción realizada para adaptar el problema B al problema A es polinomial)
   b) Qué podemos decir de A si utilizamos NB para resolver el problema A (asumimos que la reducción realizada para adaptar el problema A al problema B es polinomial)
   c) ¿Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es NP-C?

3.1. Item 1

Reduccion Polinomial:

Una reducción de tiempo polinómico es un método para resolver un problema utilizando otro en donde uno muestra que si existe una subrutina hipotética que resuelve el segundo problema, entonces el primer problema puede resolverse transformándolo o reduciéndolo a entradas para el segundo problema y llamando a la subrutina una o más veces. Si tanto el tiempo requerido para transformar el primer problema en el segundo como el número de veces que se llama a la subrutina son polinomiales , entonces el primer problema es polinomial-tiempo reducible al segundo.

Existen muchos tipos de reducciones: basadas en el método de reducción, como las reducciones de Cook, las reducciones de Karp y las reducciones de Levin, y las basadas en la cota de la complejidad, como la reducción en tiempo polinomial o la reducción de espacio logarítmica. Una de las reducciones más utilizadas es la reducción en tiempo polinomial, lo cual significa que el proceso de reducción toma un tiempo polinomial.

Este procedimiento es utilizado abundantemente en la resolucion de problemas computacionales y se lo puede pensar como una caja negra ya que es tractable.Es decir podemos tener un problema que no sepamos resolver pero si transformar en otro problema del cual tengamos una caja negra que dada una instancia de aquel problema si me lo resuelva.

Las dos transformaciones que se realizan dentro del proceso, una para transformar una instancia de un problema en una instancia de otro tipo de problema y en el otro caso para transformar la solucion de un problema a la solucion de otro problema, queremos que se realicen en tiempo polinomial. Se puede usar como caja negra (generalmente se usa este metodo para problemas tractables).

Por Ejemplo: Tenemos una instancia y de un problema Y, y a esa instancia y la se transformar mediante un proceso de transformación en una instancia x del problema X . Ahora tenemos nuestra caja negra, que vendría ser un algoritmo "B"que resuleve cualquier instancia del problema x . Entonces aplicando la caja negra obtenemos la solucion de X . Luego aplicamos otra transformación para transformar esa solucion de X al resultado del problema . Entonces tengo 2 problemas X e Y y como Y es polinialmente reducible en tiempo a X lo expresamos como:

$$Y <= pX \tag{3.1}$$

al poder transformar cualquier instancia de Y en una instancia de X en tiempo polinomico y al poder transformar cualquier solucion de X en una solucion de Y tambien en tiempo polinomico nos termina quedando definida finalmente nuestra reduccion polinomial.

Tambien tiene otro uso que es como medida de complejidad como forma de determinar la pertenencia de ciertos problemas a ciertas clases comparandolo con otros problemas que ya sabemos a que clase pertenecen. Este uso como forma de comparar y clasificar los problemas fue propuesto por Richard Karp.

Para comparar problemas mediante reducciones polinomicas decimos que al tener dos problemas X e Y, si podemos a Y reducir polinomialmente a X el problema de X es al menos tan dificil como el problema Y. Y con esto nos referimos a si es polinomico o mas complejo que polinomico .

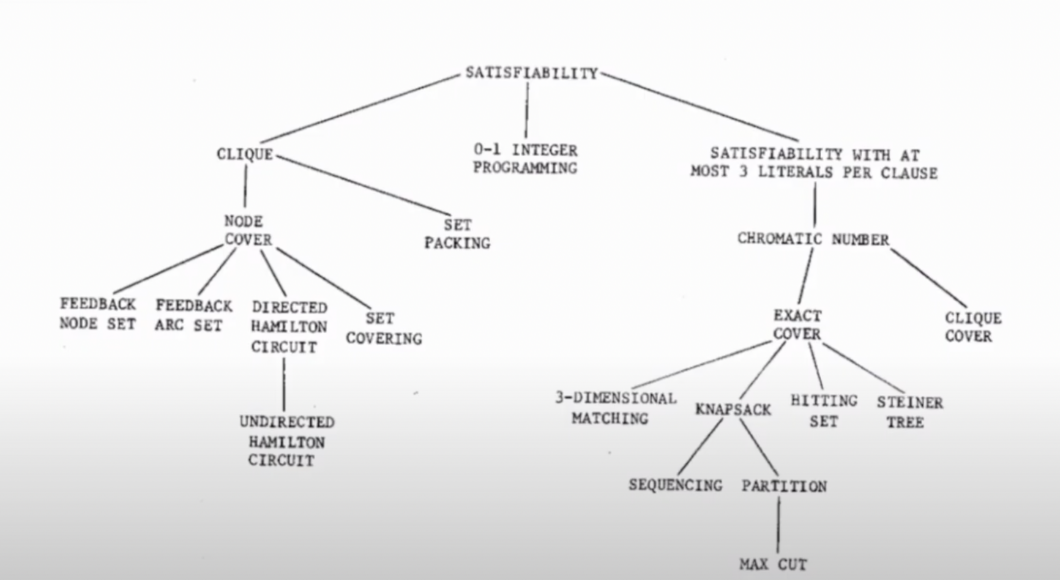
3.2. Item 2

Importancia de los problemas NP-Completos:

La clase NP contiene numerosos problemas de importancia práctica. Típicamente el problema tiene que ver con la construcción o existencia de un objeto matemático que satisface ciertas especificaciones. La versión de la construcción misma es un problema de optimizacion, donde la mejor construcción posible es la solución deseada. En la versión de decision, que es la versión que pertenece a la clase NP, la pregunta es si existe por lo menos una configuración que cumpla con los requisitos del problema. Se puede decir que la mayoría de los problemas interesantes de computación en el mundo real pertenecen a la clase NP. La aplicación “cotidiana” de la teoría de complejidad computacional es el estudio sobre las clases a las cuales un dado problema pertenece: solamente a NP o tambien a P, o quizas a ninguna de las dos.

Aquellos problemas X que cumplen con las condiciones de que pertenecen a NP-Hard y tambien pertencen a NP son NP-Completo . Podemos expresarlo coloquialmente como que X es uno de los problemas mas dificiles dentro de NP y viendolo a nivel conjunto vemos que si tenemos el conjunto Np y el conjunto NP-Hard la interseccion (aquellos que pertenecen NP-Hard y NP) son NP-C. A su vez sabemos gracias al teorema de Cook y Levin que SAT es NP-C y en este teorema se prueba que NP se puede reducir polinomialmente a SAT y al mismo tiempo SAT pertence a NP . Por ahora hay un solo problema que es que SAT es el problema mas dificil dentro de NP.

En el paper Reducibility Among Combinatorial Problems"se presenta la reduccion polinomial y lo que son 21 problemas donde para cada uno se esos problemas NP de decision se toma un problema anteriormente demostrado por NP completo y se lo reduce polinomialmente a X y con esta reduccion automaticamente este problema X pasa a ser NP-C por la definicion de reduccion polinomial. Si un problema es el mas dificil dentro de NP y ese problema que es el mas dificil en NP lo reduzco polinomialmente a otro estoy diciendo que ese otro es igual o mas complejo que el que estaba en NP-C pero como era el mas completo eso hace que sea equivalente en complejidad y que el problema X tambien pase a ser NP-C .



3.3. Item 3

3.3.1. Subitem 1

Utilizamos NA para resolver el problema B asumiendo que la reducción realizada para adaptar el problema B al problema A es polinomial podemos decir de A: Al tener el problema A y B , como B = "Pƶ se resuleve con NA podemos decir que B es polinomialmente reducible en tiempo a A y por lo tanto podemos transformar cualquier instancia de B en una instancia de A que luego puede resolverse mediante NA en tiempo polinomico siendo entonces el problema A al menos tan dificil como el probema B.

3.3.2. Subitem 2

Utilizando NB para resolver el problema A asumiendo que la reducción realizada para adaptar el problema A al problema B es polinomial podemos decir de A: Al tener el problema A y B , como B = "Pƶ A se resuleve con NB podemos decir que A es polinomialmente reducible en tiempo a B y por lo tanto podemos transformar cualquier instancia de A en una instancia de B que luego puede resolverse mediante NB en tiempo polinomico siendo entonces el problema B al menos tan dificil como el problema A y siendo A= "P".

3.3.3. Subitem 3

En los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es NP-C podemos decir: Para el primer caso donde tenemos A Y B , A=NP-C y se resuelve B con NA podemos decir que B es polonomialmente reducible en tiempo de A por lo tanto podemos transformar cualquier instancia de B en una instancia de A que luego puede resolverse mediante NA en tiempo polinomico siendo entonces el problema A uno de los mas dificiles dentro de NP.

Y en el Segundo caso donde tenemos A Y B , A=NP-C y se resuelve A con NB podemos decir que A es polinomialmente reducible en tiempo a B y por lo tanto podemos transformar cualquier instancia de A en una instancia de B que luego nos lleva a demostrar que B =NP-C ya que el problema A =NP-C , se puede reducir polinomialmente a B .

Bibliografía

[1] Cormen, T; Leiserson, C; Rivest, R; Stein, C (2009). *Introduction to Algorithms. Third Edition.*

[2] Kleinberg, J; Tardos, É (2006). *Algorithm Design.*

[3] Videos de cátedra https://www.youtube.com/user/vpode.