

Trabajo Práctico 3

[7529/9506] Teoría de Algoritmos I
Segundo cuatrimestre de 2021

Grupo:	404
Repositorio:	github.com/lucashemmingsen/7529tp3
Entrega:	nº 2 (29/11/2021)

Integrantes del grupo 404

Padrón	Apellido, Nombre	Email
76187	Hemmingsen, Lucas Emilio	lhemmingsen@fi.uba.ar
97529	Batallan, David Leonardo	dbatallan@fi.uba.ar
102593	Ausqui, Mateo Javier	mausqui@fi.uba.ar
102649	Pagura, Sebastián Martín	spagura@fi.uba.ar
106713	Serra Labán, Eloy Alejandro Lautaro	eserra@fi.uba.ar

Índice

I. Introducción	2
I.1. Resumen	2
I.2. Lineamientos básicos	2
P1Parte 1: Una campaña publicitaria masiva pero mínima	3
a. Enunciado	3
b. Formato de los archivos	3
1. Vuelos: Propuesta	4
1.1. Pasos generales	4
1.2. Ejemplo	5
2. Vuelos: Reducción algorítmica	8
3. Vuelos: Optimalidad	8
4. Vuelos: Programa	9
4.1. Uso	9
5. Vuelos: Complejidad	10
P2Parte 2: Equipos de socorro	11
a. Enunciado	11
1. Socorro: NP-Completo	12
2. Set-dominante: NP-Completo	13
3. Tratabilidad	15
P3Parte 3: Un poco de teoría	16
a. Enunciado	16
1. Reducción polinomial	17
2. Importancia de NP-Completo	17
3. Ejercicios teóricos	18
P4Correcciones	19
1. Socorro: NP-Completo	19
2. Set-dominante: NP-Completo	20
3. Tratabilidad	23

I. Introducción

I.1. Resumen

El presente informe documenta el enunciado y la solución del tercer trabajo práctico de la materia Teoría de Algoritmos I. El mismo comprende el análisis de los problemas planteados, la implementación y comparación de complejidad del primero de ellos, y varios puntos teóricos.

I.2. Lineamientos básicos

- El trabajo se realizará en grupos de cinco personas.
- Se debe entregar el informe en formato pdf y código fuente en (.zip) en el aula virtual de la materia.
- El lenguaje de implementación es libre. Recomendamos utilizar C, C++ o Python. Sin embargo si se desea utilizar algún otro, se debe pactar con los docentes.
- Incluir en el informe los requisitos y procedimientos para su compilación y ejecución. La ausencia de esta información no permite probar el trabajo y deberá ser re-entregado con esta información.
- El informe debe presentar carátula con el nombre del grupo, datos de los integrantes y y fecha de entrega. Debe incluir número de hoja en cada página.
- En caso de re-entrega, entregar un apartado con las correcciones mencionadas

P1. Parte 1: Una campaña publicitaria masiva pero mínima

a. 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?

b. Formato de los archivos

El programa debe recibir por parámetro el path del archivo donde se encuentran los vuelos disponibles entre pares de ciudades y la cantidad maxima de pasajeros en la misma.

El archivo debe ser de tipo texto y presentar por renglón, separados por coma el pais de origen el de destino y la capacidad del mismo. Las primeras dos lineas del archivo contiene el pais de origen y de destino respectivamente

Ejemplo: "vuelos.txt"

```
A
B
A,D,140
A,E,200
D,B,100
...
```

Debe resolver el problema y retornar por pantalla la solución. Debe mostrar por consola en en que vuelos poner la publicidad. Además imprimir cual es la cantidad maxima de pasajeros que puede ir de A a B.

1. Vuelos: Propuesta

Enunciado P1.1

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

Para la solución al problema planteado se propone reducir el problema a un problema de corte de flujo y solucionarlo con el algoritmo de Edmonds-Karp.

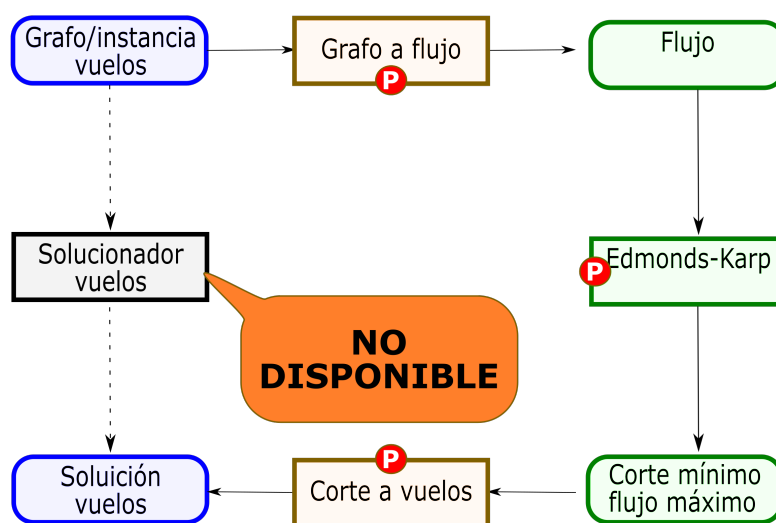


Figura 1: Reducción del problema, todas las operaciones son polinómicas.

1.1. Pasos generales

En resumen, la solución completa sigue los siguientes pasos:

- Procesar el archivo en el formato dado a un grafo dirigido, utilizando lista de adyacencias.
 - Cada nombre de aeropuerto es tratado como un número indicando la posición, en la lista de adyacencias, de cada lista de destino para un mismo origen.
 - Cada elemento está compuesto por una tupla de id de destino y "peso." asociado.
 - La estructura permite que peso pueda no ser simplemente un número, y el alias puede no ser un string. Pero en este punto, por el formato de entrada, sí lo serán.
- Convertir el grafo en un diagrama de flujo, en el que:
 - En cada arco original se almacena:
 - una **capacidad máxima**, cuyo valor está dado por el anterior peso numérico del grafo.
 - y un **flujo**, inicialmente 0 (cero).
 - Cada arco original (a través de la clase `ArcoDirecto`) se obtiene un valor dado por la capacidad residual (la capacidad - el flujo). En el sentido contrario del grafo se almacena una referencia (a través de `ArcoInverso`) que, como valor, devuelve el flujo del arco directo (en el sentido original).

Por ejemplo, un arco $A \xrightarrow{10} B$ es reemplazado por un `ArcoDirecto` $A \xrightarrow{0/10} B$ que devuelve un valor $10 - 0 = 0$, y se agrega un nuevo arco $B \rightarrow A$ (que referencia al anterior) y devuelve un valor igual al flujo (0).

3. Se aplica Edmonds-Karp:

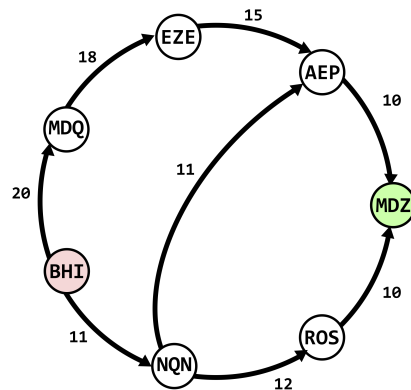
- a) Se inicializa en 0 un contador interno de flujo aumentado.
- b) Mientras haya un camino, encontrado con BFS, desde el origen y el destino dados en el archivo, cuyos arcos tengan valores mayores que cero:¹
 - 1) Busca el cuello de botella (el menor de todos los valores).
 - 2) Aumenta el camino: Si es un arco directo, aumenta el flujo (reduciendo su "valorz aumentando el del arco inverso asociado); si es un arco inverso, disminuye su flujo (aumentando su "valor", y reduciendo el del arco directo, ya que es el valor residual).
 - 3) Se aumenta un contador interno de flujo aumentado con el valor del cuello de botella.
4. Al terminar, se recorre con un BFS todos los nodos a los que pueda llegarse por un arco no saturado, es decir donde no haya un valor directo de 0 (si lo hay se lo agrega aun subconjunto de arcos **corte**), y se los agrega al subconjunto A; el resto de los nodos pertenecen al subconjunto B.
5. Se devuelve el contador de flujo como capacidad máxima de pasajeros del origen al destino, y el conjunto de arcos **corte** como los vuelos que cumplen con el criterio dado.

1.2. Ejemplo

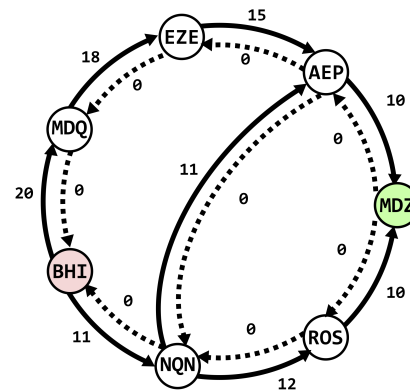
Tomaremos como ejemplo el archivo disponible en `tests/entradas/test_ejtp3.txt`. En la línea 1 y 2 tenemos el origen y destino (respectivamente) de los pasajeros a los que apunta la campaña. Y a continuación, en el epígrafe de cada figura, iremos desarrollando el ejemplo.

```
1      BHI
2      MDZ
3      AEP,MDZ,10
4      BHI,MDQ,20
5      BHI,NQN,11
6      EZE,AEP,15
7      MDQ,EZE,18
8      NQN,AEP,11
9      NQN,ROS,12
10     ROS,MDZ,10
```

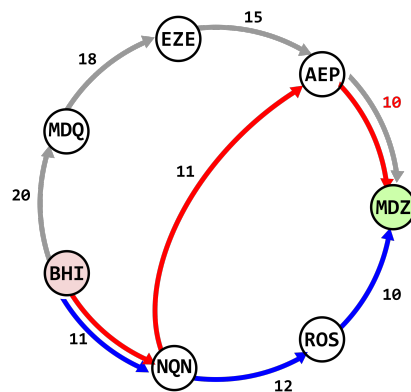
¹Esto es, para un arco directo significa que hay un valor residual, el flujo no llegó a su capacidad máxima. para un arco inverso significa que hay un flujo en el sentido contrario que podría disminuirse.



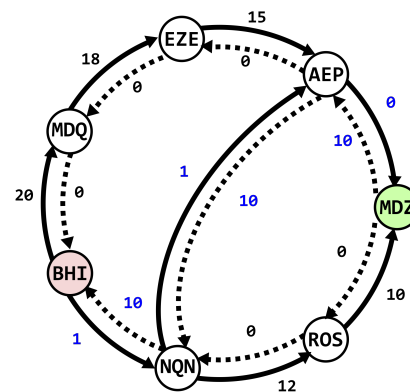
(a) Grafo inicial con vuelos y sus capacidades



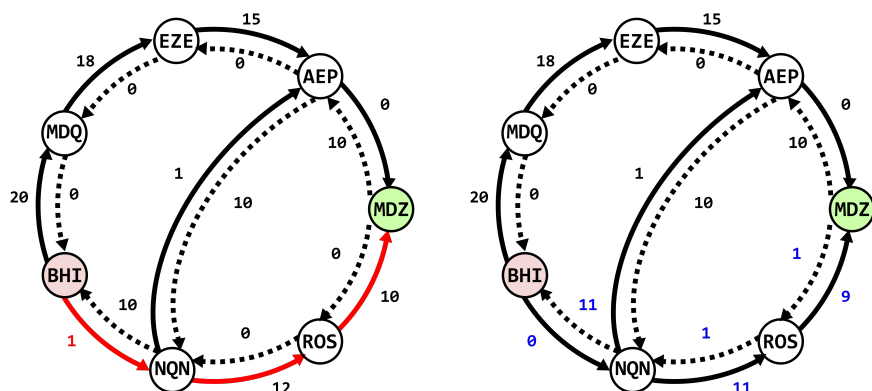
(b) Conversión a grafo residual de flujo, inicialmente flujo 0 (flecha intermitente) y residuo disponible igual a su capacidad (flecha sólida).



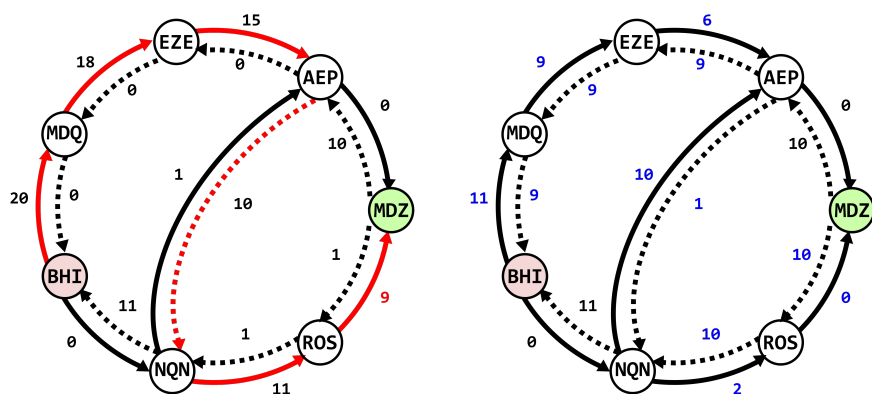
(a) **BFS:** Hay varios caminos disponibles, (b) **Aumentar:** A cada arco en sentido del



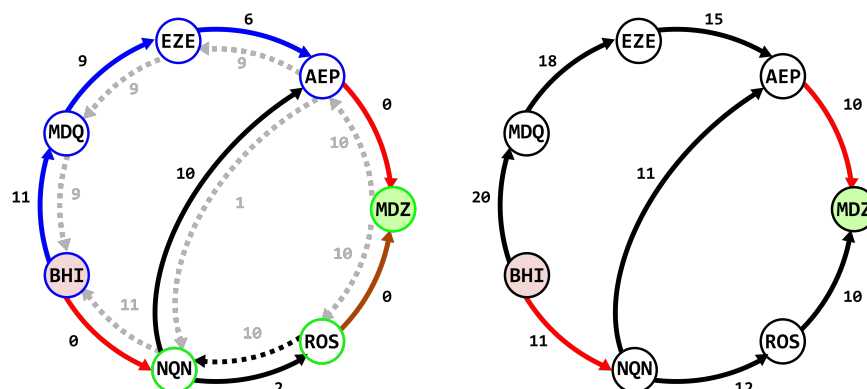
totalmente 2 mínimos (rojo y azul). To- camino se le resta el cuello (10, en), y a maremos el rojo con un cuello de botella cada arco en sentido contrario se le suma (manteniéndose la capacidad). El flujo total queda en 10.



(a) **BFS:** El siguiente camino mínimo es el (b) **Aumentar:** Se vuelven a actualizar los indicado en rojo. El camino anterior ya no valores. Ahora el flujo total queda en 11, está disponible porque tiene al menos un al sumarle a los 10 anteriores, el último arco ($AEP \rightarrow MDZ$) en 0. Siendo camino cuello. El arco que era cuello de botella directo significa que no hay residuo, está queda en 0. saturado. El cuello de botella es 1 ($BHI \rightarrow NQN$).



(a) **BFS:** Como ahora $BHI \rightarrow NQN$ que- (b) **Aumentar:** Una vez actualizado en 9, dó saturado, el único camino disponible es también aumenta el flujo total en la mis- el indicado en rojo. El cuello son los 9 de ma cantidad llegando a 20. Aumentar el camnino de reversa implicó que 9 que lle- gaban a AEP por NQN ahora lo hacen por EZE; quedando esos 9 disponibles para ir a ROS.



(a) **Subgrupos y corte:** En azul nodos del subgrupo A (y los arcos por los que instancia como un problema de flujo, se llegó). En verde el resto de los nodos, que convierte nuevamente en una instancia del problema original. En rojo los vuelos seleccionados, y la cantidad de pasajeros máximos que pertenecen al corte, y en cionados, y la cantidad de pasajeros máximos que pertenecen a un subgrupo.

2. Vuelos: Reducción algorítmica

Enunciado P1.2

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

Puede considerarse una reducción algorítmica ya que se convirtió el grafo con las cantidades de pasajeros por vuelo, en un diagrama de flujo para resolverlo con Edmonds-Karp, y luego volvió a convertirse en un grafo (dado por los nodos A, B y los arcos de corte). Ambas transformaciones y la resolución son polinómicas, por lo que la **reducción es polinómicas**.

3. Vuelos: Optimalidad

Enunciado P1.3

¿Podría asegurar que su solución es óptima?

El algoritmo de Edmonds-Karp es una variante más específica de Ford Fulkerson,² por lo que también es óptimo para encontrar el flujo máximo y cuál es el conjunto de ArcoInverso por los que todo el flujo está pasando entre un subgrupo A que contiene a la fuente, y otro B que contiene al sumidero.

²especifica el criterio de selección del camino de aumento, mientras que Ford Fulkerson no.

4. Vuelos: Programa

Enunciado P1.4

Programa la solución

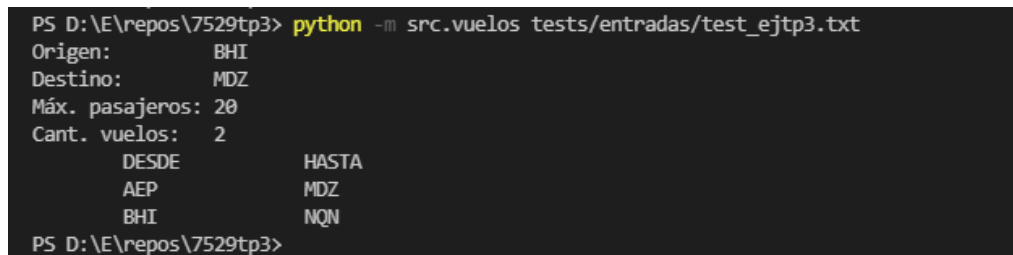
4.1. Uso

Para ejecutar el programa es necesario [descargar e instalar Python 3.10](#) y, desde la línea de comandos ubicado en la carpeta donde se descomprimió,³ ejecutar:

```
python -m src.vuelos NOMBRE_DE_ARCHIVO
```

Siendo NOMBRE_DE_ARCHIVO la ruta (relativa a la carpeta de trabajo) al archivo con las rutas, según el formato indicado en Formato de archivos. Por ejemplo, para usarse con el ejemplo dado en la sección P1.1:

```
python -m src.vuelos tests/entradas/test_ejtp3.txt
```



```
PS D:\E\repos\7529tp3> python -m src.vuelos tests/entradas/test_ejtp3.txt
Origen:      BHI
Destino:     MDZ
Máx. pasajeros: 20
Cant. vuelos: 2
      DESDE      HASTA
      AEP        MDZ
      BHI        NQN
PS D:\E\repos\7529tp3>
```

Figura 7: Salida por pantalla del programa, mismo resultado que el ejercicio.

3

5. Vuelos: Complejidad

Enunciado P1.5

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

Para analizarla tomamos como referencia los pasos indicados en el punto P1.1.

1. Procesar el archivo puede asumirse una complejidad $O(E)$: consiste en insertar como mucho 2 sublistas en una lista (la de adyacencias) y un arco como elemento de una de esas sublistas, por cada arco (E). Espacialmente, es $O(V + E)$
2. **Conversión de grafo de flujo.** Se debe iterar cada arco realizando una inserción y una modificación (pueden presumirse $O(1)$ cada una). Por lo tanto, temporalmente sería $O(E)$. Si bien aumenta su tamaño, se duplican las aristas, por lo que espacialemente sigue siendo $O(V + E)$.
3. Edmonds-Karp tiene $O(VE)$ operaciones de emphpresumiblemente $O(E)$, donde:
 - 3.b Se aplica BFS, nuestra implementación de BFS es $O(V + E)$ temporalmente (iteramos dos veces por nodo V , y recorremos una vez cada arco E) y $O(V)$ espacialemente (V para nodo anterior, V para nodos a visitar y V para el camino).
 - 3.1 Cuello de botella, es una simple iteración por cada nodo del camino, almacenando el mínimo. $O(V)$ temporalmente, $O(1)$ espacialemente.
 - 3.2 Aumentar el camino, nuevamente una iteración por cada nodo del camino. $O(V)$ temporalmente.
4. **Computar subgrupos A, B y el corte.** Es un BFS, que aunque puede terminar antes podría ser que B sea el sumidero y por lo tanto seguiría siendo temporalmente $O(V + E)$. Espacialmente tenemos que $A + B = V$, y que $corte \leq E$, por lo tanto es $O(V + E)$.
- 5 **Convertir resultado:** se debe recorrer cada arco para convertirlo e imprimirlo. Es temporalmente $O(E)$, y como se necesita convertir uno por uno también lo es espacialemente.

En nuestra implementación el impacto en la complejidad temporal sigue dado por Edmonds-Karp, $O(VE^2)$, pudiendo descartarse las operaciones $O(V + E)$, $O(V)$ y $O(E)$. Con respecto **al problema original**, la complejidad temporal es $O(VE^2)$ siendo V la cantidad de aeropuertos y E la de viajes. Espacialmente, en el momento de mayor consumo hay 2 instancias de $O(V + E)$ por lo que pertenece a esa complejidad.

P2. Parte 2: Equipos de socorro

a. 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.

Se pide:

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.

1. Socorro: NP-Completo

Enunciado P2.1

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

El problema de situar los equipos de socorro revuelve en torno a el alcance de estos con respecto al conjunto de las estaciones. Se busca que k equipos de emergencia alcancen para que toda estación esté cubierta. Dado que un equipo sólo puede interactuar con su estación o las adyacentes a esta, el problema resulta ser una variante del problema «determinar set dominante». En este caso, este problema está restringido a determinar un set dominante S con hasta k elementos.

«Un "set dominante" S de un grafo G es un subconjunto de vértices de dicho grafo, tales que cada vértice del grafo que no se encuentre en el subconjunto S , es adyacente a aquellos que sí.» ⁴

El algoritmo solucionador, a *grosso modo*, debe determinar un set dominante S , de límite de tamaño k , el cual resulte ser el posicionamiento preferible de los equipos de socorro. El problema de «hallar set dominante» fue probado NP-Completo (consecuencia de que Richard M. Karp determinó que «set cover problem» es NP-Completo, ocasionada debido a que toda instancia de dominating set [“hallar set dominante”] es reducible a una de set cover problem y viceversa). Por lo tanto, y siendo que el algoritmo es básicamente una solución del problema “hallar set dominante”, el problema de ubicar los equipos de socorro es NP-Completo.

⁴Wayne Goddard, Michael A. Henning. «Independent domination in graphs: A survey and recent results». Discrete Mathematics, Volume 313, Issue 7, 2013, pág. 839-854, ISSN 0012-365X, <https://doi.org/10.1016/j.disc.2012.11.031>

2. Set-dominante: NP-Completo

Enunciado P2.2

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

Para probar que el problema «set dominante» es *NP – Completo* tenemos que probar que el problema sea *NP* y también *NP – Hard*.

1. Set dominante es NP: Dado un set puedo recorrer cada vértice, marcarlo junto con sus vértices adyacentes y comprobar que todos los vértices del grafo fueron marcados. Un algoritmo que certifica el problema podría ser:

```

1  def es_set_dominante(grafo, set):
2      visitados = []
3      for vertices in set:
4          if vertice not in visitados:
5              visitados.append(vertice)
6          for v_adyacente in vertice:
7              if v_adyacente not in visitados:
8                  visitados.append(v_adyacente)
9      if len(visitados) == len(grafo):
10         return true
11     return false

```

Su complejidad es de $O(n^2)$ y por lo tanto puedo decir que es un certificador eficiente. Luego, puedo decir que el problema de set dominante es *NP*.

2. Set dominante es NP-Hard: Quiero probar que el problema *Vertex Cover*, que es *NP*, se puede reducir al problema set dominante.

Si tengo un grafo con un set de vértices seleccionados y quiero ver si cumple la condición de *Vertex Cover*, puedo transformarlo en un problema de set dominante. Para lograrlo, por cada arista que une los vértices (u, v) agrego un nuevo nodo con aristas hacia u y v . Si en este nuevo grafo el set es dominante entonces el set es una cobertura de vértices.

Como puedo reducir polinomialmente *Vertex Cover* al problema de set dominante, y *Vertex Cover* es *NP*, entonces el problema de set dominante es *NP – Hard*.

underlineEjemplo: ¿El set $(1, 2, 4)$ siguiente a es una cobertura de vértices?

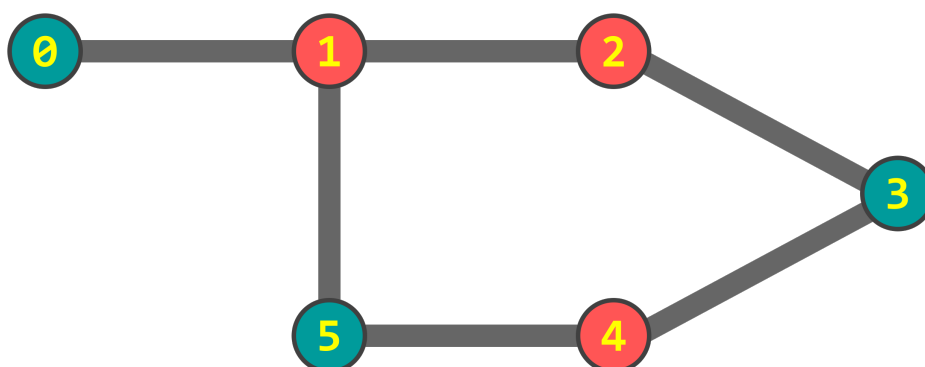


Figura 8: Grafo original de ejemplo

Para verlo podemos agregar un nodo para cada arista y unirlo a los vértices originales:

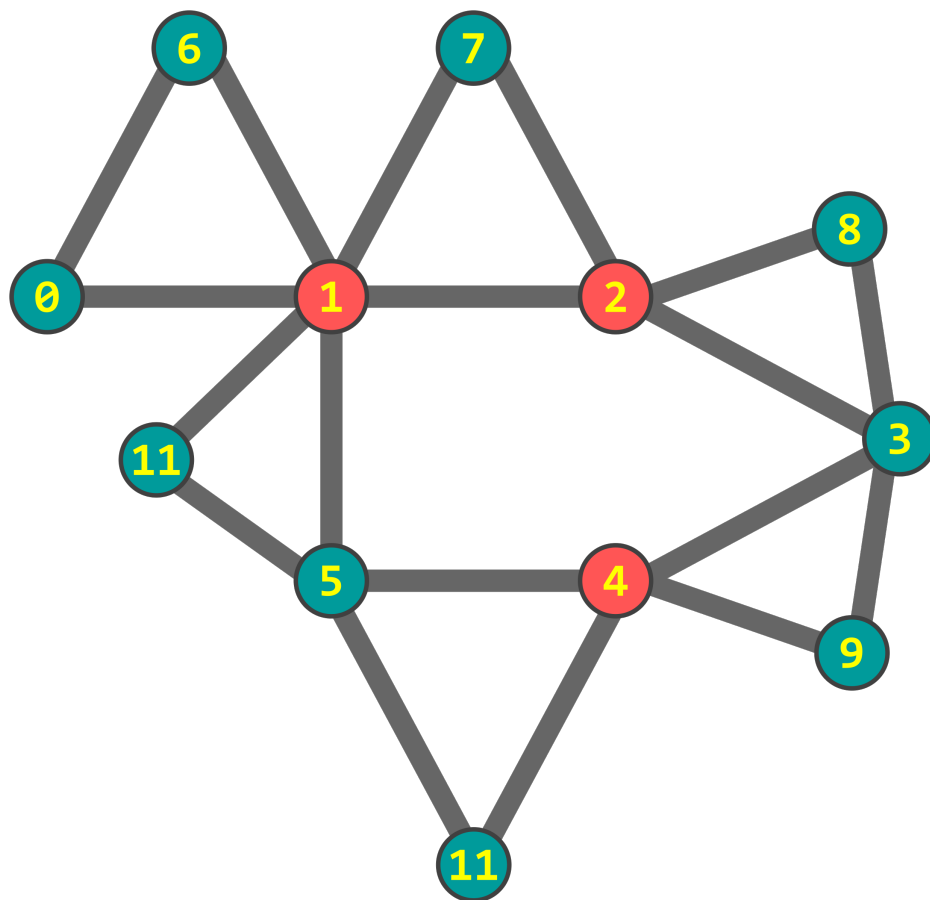


Figura 9: **Grafo transformado**. Nótese por ejemplo que se dada la arista $0 \rightarrow 1$, se agrega ⑥ unido a ambos.

Y como el set es un set dominante para el nuevo grafo, entonces el set es un vertex cover para el grafo original.

3. Tratabilidad

Enunciado P2.2

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

Basado en las demostraciones previas, el problema resulta ser *NP-Completo*. Si este problema fuese tractable (bajo la definición dada en la materia, resuelto en forma polinomial), estaríamos probando que $P = NP$. Fuera de eso, el problema planteado tiene símil complejidad a «dominating set», el cual no puede ser resuelto en tiempo polinomial. En consecuencia, el problema planteado no puede ser resuelto de manera eficiente.

P3. Parte 3: Un poco de teoría

a. 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?

1. Reducción polinomial

Enunciado P3.1

Defina y explique qué es una reducción polinomial y para qué se utiliza.

Una **reducción algorítmica** es un procedimiento por el cual se transforma una instancia de un problema original (o *reducible*), en otra instancia de otro problema (*reducido*) que puede resolverse por un algoritmo dado (a modo de «caja negra»). Y luego se transforma la solución obtenida en una solución del problema original.

Eso puede llevarse a cabo porque:

- No se sabe resolver el problema original y sí el reducido.
- Se conoce un algoritmo para resolver el problema original, pero el algoritmo para reducir el problema reducido tiene una menor complejidad.

Cuando estas transformaciones se realizan en tiempo polinomial, entonces decimos que es una **reducción polinomial**.

La reducción polinomial también es de utilidad como **medida de complejidad** en el caso de saber la complejidad de alguno de los problemas. Por ejemplo, dados los problemas X e Y , se dice $Y \leq_p X$ si Y es polinomialmente reducible a X . Entonces, si $Y \leq_p X$, entonces podemos decir que X es al menos tan difícil de resolver que Y .

2. Importancia de NP-Completo

Enunciado P3.2

Explique detalladamente la importancia teórica de los problemas NP-Completo.

Un problema es NP-Completo si pertenece a NP y también pertenece NP-Hard, lo cual quiere decir que es uno de los problemas más difíciles de resolver dentro de NP. Esta condición de ser de los problemas más difíciles de resolver dentro de NP los hace importantes ya que si se encontrara la solución a un problema NP-Completo en tiempo polinómico, se probaría que $P = NP$.

Esto ocurre dado que sabemos que $P \subseteq NP$,⁵ y si se pudiera probar que el problema más difícil de NP puede solucionarse en tiempo polinomial, entonces todos los problemas de NP se podrían solucionar en tiempo polinomial y por lo tanto $NP \subseteq P$.

Finalmente, si $NP \subseteq P$ y $P \subseteq NP$, entonces $P = NP$.

⁵Si se tiene un algoritmo P que resuelve un problema de decisión, puede usarse como "algoritmo certificador" NP, para verificar la solución dada sea correcta simplemente comparando que sean la misma. Por esto, todo algoritmo de decisión P es también NP; pero no necesariamente al revés.

3. Ejercicios teóricos

Enunciado P3.2

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.

1. 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)
2. 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)
3. ¿Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es NP-C?

1. 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)

$B \leq_p A$. Que A es al menos tan difícil de resolver como B. No me dice nada sobre la complejidad de A.

2. 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)

$A \leq_p B$. Se puede decir que $A \subseteq P$. Si B es igual o más difícil de resolver que A, y B es P; entonces A tiene que ser P.

3. ¿Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es NP-C?

A es NP-C \implies A es NP-H y A es NP.

- 1) $B \leq_p A$ Puedo decir que B es al menos NP.
- 2) $A \leq_p B$ Entonces puedo decir que B es NP-Hard.

P4. Correcciones

1. Socorro: NP-Completo

Enunciado P2.1

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

El problema enunciado consiste en encontrar el número de equipos de socorro necesarios en las estaciones ferroviarias de los distintos ramales y subramales de la red ferroviaria de un país de manera tal que ante una emergencia en una estación haya un equipo de socorro en dicha estación, o en el peor de los casos en una estación vecina que tenga un trayecto directo entre ellas dos. Además el número de equipos mínimo necesario debe ser menor a k .

Analizando todo esto, se puede ver que la red ferroviaria formará un grafo no dirigido, en el que cada vértice será una estación y las aristas serán los caminos entre estaciones. Este grafo probablemente será desconexo, ya que no necesariamente haya comunicación entre todos los ramales.

Por ende, si representamos la red ferroviaria como un grafo no dirigido $G(V, E)$ nuestro problema consistirá en encontrar el menor conjunto de vértices (estaciones) V' de tamaño k' de manera tal que cualquier vértice de V pertenezca a V' o en su defecto sea adyacente a al menos un vértice que pertenezca a V' . Además se debe cumplir con que k' sea menor a k .

Un set dominante en un grafo no dirigido $G(V, E)$ es un subconjunto de vértices V' tal que cada vértice en el grafo pertenece a V' o es adyacente a algún vértice en V' . Dado un grafo G y un número entero k , el problema consiste en encontrar un set dominante de tamaño k para un grafo dado G . El número dominante $\gamma(G)$ es el cardinal del menor conjunto dominante de G .

Por ende, en nuestro problema nos están pidiendo el número dominante $\gamma(G)$ para un problema de set dominante y que este sea menor a k .

- Dado $G = (V, E)$:
- Donde k tamaño máximo del conjunto dominante
- T certificado = subconjunto de nodos de V
- Y puedo verificar en tiempo polinomial:

$$\text{Si } (T \leq k \wedge (\forall v \in V, v \in T \vee v, w/w \in T)) \implies \in NP$$

Si probamos que el problema de set dominante es *NP-Completo* quedará demostrado entonces que este problema también lo es.

2. Set-dominante: NP-Completo

Enunciado P2.2

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

Para probar que el problema «set dominante» es *NP-Completo* tenemos que probar que el problema sea *NP* y también *NP-Hard*.

1. Set dominante es NP: Dado un set puedo recorrer cada vértice, marcarlo junto con sus vértices adyacentes y comprobar que todos los vértices del grafo fueron marcados. Un algoritmo que certifica el problema podría ser:

```

1  def es_set_dominante(grafo, set):
2  visitados = []
3  for vertice in set:
4      if vertice not in visitados:
5          visitados.append(vertice)
6      for v_adyacente in vertice:
7          if v_adyacente not in visitados:
8              visitados.append(v_adyacente)
9  if len(visitados) == len(grafo):
10     return true
11     return false

```

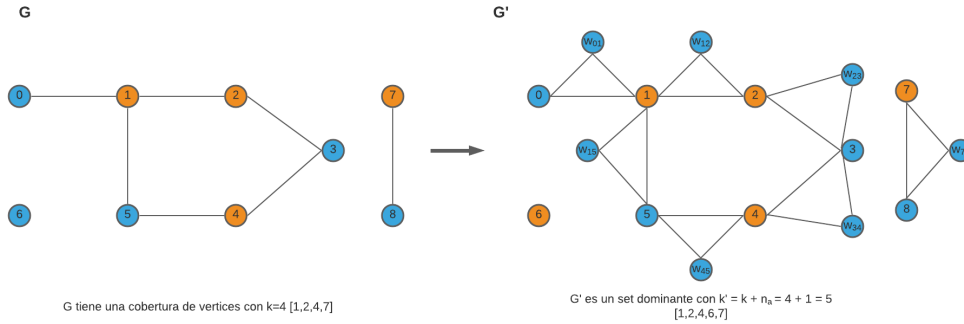
Su complejidad es de $O(n^2)$ y por lo tanto puedo decir que es un certificador eficiente. Luego, puedo decir que el problema de set dominante es *NP*.

2. Set dominante es NP-Hard: Para demostrar que el problema del conjunto dominante es *NP-Hard*, necesitamos mostrar que un problema conocido *NP-Completo* es reducible en tiempo polinómico a un problema de conjunto dominante. En este caso elegimos el problema de cobertura de vértices (*vertex cover*) como problema *NP-Completo* conocido.

La cobertura de vértices en un grafo no dirigido $G(V, E)$ es un subconjunto de vértices V' de V , tal que cada arista en el grafo G está conectada con al menos un vértice de V' . Este problema entonces busca si existe una cobertura de vértices de un tamaño dado k para un grafo no dirigido dado $G(V, E)$.

Los dos problemas parecen ser bastante similares, pero hay una diferencia esencial, que se da en el caso que el grafo tenga vértices aislados. En el caso de *vertex cover* estos vértices pueden ignorarse ya que no hay ninguna arista conectada a ellos. En cambio, en el caso del *set dominante*, esos vértices deberían incluirse en el conjunto dominante, ya que no hay ningún vértice adyacente a ellos.

Queremos demostrar que dada una instancia de un problema de *vertex cover* (G, k) , podemos producir una instancia equivalente de un problema *set dominante* en tiempo polinómico. Para reducir del problema de *vertex cover* al de *set dominante*, necesitamos construir un nuevo grafo G' desde el grafo G dado. Al inicio, G' es un duplicado de G . Luego, por cada arista u, v en G creamos un nuevo vértice w_{uv} en G' y agregamos aristas u, w_{uv} y v, w_{uv} en G' . Sea n_a el número de vértices aislados y $k' = k + n_a$. Esta reducción se muestra en la figura siguiente.



```

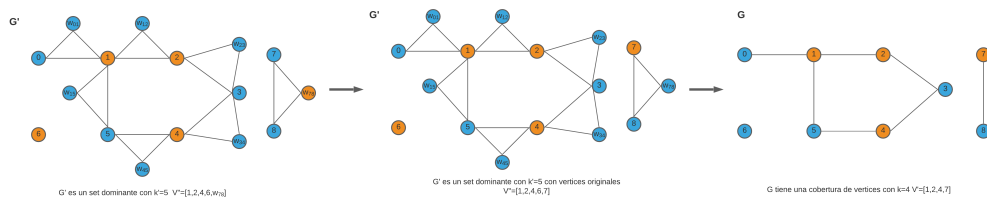
1  def es_set_dominante(grafo, set):
2      visitados = []
3      for vertice in set:
4          if vertice not in visitados:
5              visitados.append(vertice)
6          for v_adyacente in vertice:
7              if v_adyacente not in visitados:
8                  visitados.append(v_adyacente)
9      if len(visitados) == len(grafo):
10         return true
11     return false

```

La reducción tiene complejidad $O(V + E)$ y por lo tanto se puede reducir polinomialmente. Para establecer la correctitud de la reducción necesitamos demostrar que un grafo G tiene una cobertura de vértices de tamaño k si y sólo si G' tiene un set dominante de tamaño k' .

\Rightarrow) Como primera medida chequeamos que si V' es una cobertura de vértices de G , entonces $V'' = V' \cup V_a$ es un set dominante para G' donde V_a son los vértices aislados. Para observar si V'' es un set dominante, primero observemos que todos los vértices aislados son parte de V'' . Luego, todos los vértices agregados w_{uv} en G' corresponden a una arista u, v en G , implicando que u o v pertenecen a la cobertura de vértices V' . Por ende, w_{uv} va a estar dominado por el mismo vértice en V'' . Finalmente, cada uno de los vértices no aislados originales, va a estar conectado al menos a una arista, y por consiguiente va a ser parte de V' o por el contrario todos sus vecinos van a ser parte de V' . Entonces, cada vértice no aislado original está en V'' o es adyacente a un vértice en V'' .

\Leftarrow) Ahora queremos demostrar que si G' tiene un set dominante V'' de tamaño $k' = k + n_a$, entonces G tiene una cobertura de vértices V' de tamaño k . Todos los vértices aislados de G' deben ser parte del set dominante. Sea V''' los vértices pertenecientes al set dominante V'' que no son aislados, algunos de estos vértices van a ser de los que fueron especialmente creados y que no son parte del grafo original G . Podemos afirmar que no es necesario utilizar ninguno de estos vértices creados en V''' , ya que si algún vértice nuevo w_{uv} (que es adyacente solo a los vértices originales u y v) pertenece a V''' , entonces podemos modificar V''' reemplazando w_{uv} por u o v . Suponiendo que lo reemplazamos por u , u seguirá dominando a v y w . Por ende reemplazando w con u en el set dominante, se siguen dominando los mismos vértices, y potencialmente incluso alguno más. Denotemos como V' el set dominante luego de esta modificación.



Finalmente, podemos afirmar que V' es una cobertura de vértices para G . Si por el contrario, existiese alguna arista u, v de G que no estuviese cubierta, es decir, que u o v no estuviesen en V' , entonces los vértices creados w_{uv} no serían adyacente a ningún vértice de V'' en G' , contradiciendo la hipótesis de que V'' era un set dominante para G' . Por ende, podemos concluir que el problema del set dominante es un problema *NP-Completo*.

3. Tratabilidad

Enunciado P2.2

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

Basado en las demostraciones previas, el problema resulta ser *NP-Completo*. Si este problema fuese tractable (bajo la definición dada en la materia, resuelto en forma polinomial), estaríamos probando que $P = NP$. Fuera de eso, el problema planteado tiene símil complejidad a «dominating set», el cual no puede ser resuelto en tiempo polinomial. En consecuencia, el problema planteado no puede ser resuelto de manera eficiente.