Resolviendo *A Subway for Boroughgraph*

José Daniel Fandiño* – 201423487

Teoría de grafos – 02/12/16

Resumen

Este proyecto pretende resolver el problema de maratones *A Subway for Boroughgraph*. Grosso modo, el problema requiere que se diga si un grafo es planar o no a partir de sus nodos y sus ejes. El presente documento busca indagar la teoría de grafos planares para ver cómo se puede resolver el problema y cómo se puede aplicar algorítmicamente. Se explorarán entonces los distintos algoritmos existentes que prueban la planaridad de un grafo y se escogerá uno para implementarlo. Se explicará de manera general la implementación y se darán las instrucciones para la ejecución del programa.

Contenido

*Disponibilidad semana del 5 de diciembre: cualquier día/hora.

Introducción

La teoría de grafos es una rama de la matemática que es bastante aplicable a situaciones reales. De hecho, Leonhard Euler, pionero en esta área, se interesó en los grafos por un problema de la vida real; el problema de los siete puentes de Königsberg. Así, las aplicaciones de los grafos van desde la química hasta la malla vial, y se aplican con el propósito de entender mejor distintos problemas y darles soluciones acertadas.

La computación es un área donde los grafos son altamente aplicables, por dos razones principales. Por un lado, la teoría de grafos permite avanzar dicha área al ser útil en problemas de redes, algoritmos, entre otros problemas propios de la computación. Pero además de esto la computación permite la representación y simulación de situaciones, sirviendo como herramienta para resolver problemas de grafos de otras áreas.

Pasar de un contexto a un modelo teórico de definiciones, postulados y teoremas, y finalmente a una aplicación algorítmica no es sencillo. Por esta razón es común que en competencias de programación aparezcan problemas donde sea necesario realizar dichas transiciones, y no es de extrañarse entonces que haya una gran variedad de problemas sobre teoría de grafos.

En la página web de la Universidad de Valladolid (UVA) se encuentra un gran número de dichos problemas. Entre ellos está el problema 12941 – *A Subway for Boroughgraph* (Ver anexo), el cual puede resolverse utilizando la teoría de grafos y se desarrollará en el presente trabajo.

Contexto del problema

En resumidas cuentas, el enunciado nos presenta el siguiente dilema; ¿es un grafo planar o no? Esto en realidad no se presenta tan explícito. El problema nos introduce una ciudad ficticia llamada Boroughgraph. El alcalde de esta ciudad quiere construir un sistema de metro y tiene varias opciones en mente, pero como la zona tiene varios pantanos debe hacer los túneles a cierta profundidad consumiéndole todos los recursos disponibles y así no tiene dinero para hacer líneas en la superficie o a mayor profundidad. Es decir, todas las líneas del metro deben estar a cierto nivel bajo tierra y por esta razón no se pueden cruzar (excepto en sus extremos).

Teoría para solución

Para solucionar éste problema me basaré en la teoría de grafos planares. Esto es lo más apropiado para el problema pues ofrece distintos teoremas y algoritmos que podrían ser útiles.

Sean entonces las estaciones del metro los nodos del grafo, y las líneas entre las distintas estaciones los arcos. De esta manera vemos que tomar una aproximación mediante grafos planares es acertado pues encaja con el contexto del problema y así éste se "reduce" a resolver si un grafo es planar o no dados sus arcos y sus nodos.

Pongo “reduce” entre comillas pues igualmente no es nada sencillo averiguar si un grafo es planar o no. Detrás hay una inmensa teoría de la cual parten varios algoritmos que no son fáciles de entender o de implementar.

Partamos entonces de la característica de Euler en grafos. Resulta que para grafos planares conexos se cumple $V - E + F = 2$, donde V es el número de vértices, E es el número de arcos y F es el número de caras. De lo anterior obtenemos el siguiente corolario:

Corolario 1: Sea G un grafo conexo y simple con V vértices ($V > 3$) y E arcos. Luego $E \leq 3V - 6$.

Dem: Primero veamos que $F \leq \frac{2}{3}E$. Esto es porque una cara tiene por lo menos 3 arcos y cada uno de estos arcos limita con 2 caras. De esta manera, se reemplaza en la fórmula de Euler y obtenemos:

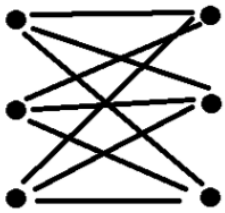
$$V - E + \frac{2}{3}E = 2$$

$$V - \frac{E}{3} = 2$$

$$3V - 6 = E$$

■

Cabe resaltar que la anterior propiedad no es bidireccional. Es decir, si tenemos un grafo con $E \leq 3V - 6$ no quiere decir que sea planar. La siguiente imagen es prueba de ello.



V = 6, E = 9

Ilustración 1 (Ejemplo propio)

Vemos en la ilustración 1 que $9 \leq ((6 \times 3 - 6) = 12)$ pero no hay forma de posicionar los nodos de manera que sea planar.

Entonces, ¿qué condiciones aseguran la planaridad? Existe para esto el teorema de Kuratowski y Wagner. Antes de enunciarlo es necesario definir primero varios conceptos.

Definición 1. Subdivisión: Sea $G(V, E)$ un grafo simple conexo. Defina $G_w(V', E')$ el grafo subdivisión de G donde $V' = V \cup \{w\}$ para algún vértice $w \notin V$ y $E' = ((E - xy) + xw) + wy$ para algún $xy \in E$.

Definición 2. Contracción: Sea $G(V, E)$ un grafo simple conexo con $xy \in E$. Defina $G_w(V', E')$ el grafo contraído de G donde $V' = (V \backslash \{x, y\}) \cup \{w\}$ para algún vértice $w \notin V$ y $E' = (E - xy) + \{wz | xz \in E \text{ ó } yz \in E\}$.

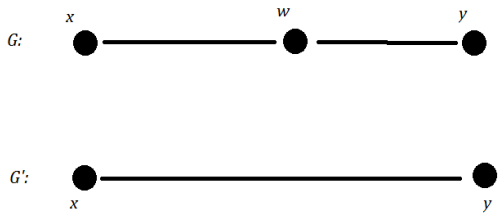


Ilustración 2 (Ejemplo propio)

Vemos en la ilustración 2 que G es subdivisión de G' y G' es la contracción de G.

Definición 3. Minor: Sea H un grafo simple. Se dice que H es *minor* de G si H puede ser creado a partir de cero o más de los siguientes pasos en G:

- Remover arcos
- Remover vértices
- Realizar contracción.

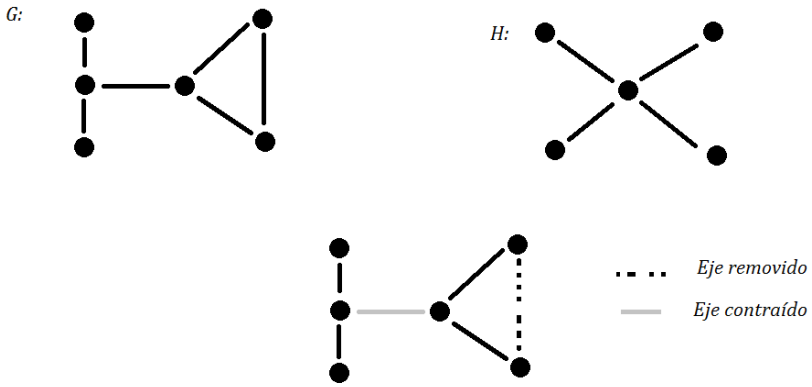


Ilustración 3 (Ejemplo propio)

La ilustración 3 nos permite observar un ejemplo donde H es *minor* de G y los pasos tomados para llegar a H desde G

Definición 4. Minor topológico: Sea H un grafo simple. Se dice que H es *minor* topológico de G si existe una subdivisión de H que sea isomorfa a un subgrafo de G.

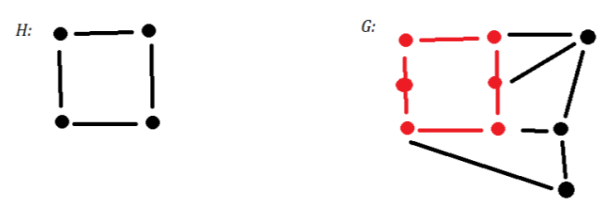


Ilustración 4 (Ejemplo propio)

En la ilustración 4 vemos que H es un minor topológico de G pues existe una subdivisión de H (marcado en rojo) que es subgrafo de G. Además, en la ilustración 2 se observa que G' es *minor* topológico y *minor* de G, mientras que en la ilustración 3 se observa que H es *minor* de G pero no *minor* topológico. De hecho, en general si H es *minor* topológico de G entonces también es $minor$, pues H puede ser creado a partir de la contracción de las subdivisiones de H y removiendo todos los nodos y ejes que no sean de H.

Definición 5. k-conexo: Un grafo es k-conexo si $|G| > k$ y $G - X$ es conexo para cada $X \subseteq V$ con $|X| < k$. La conectividad de un grafo es $\kappa(G)$, y es el k más grande tal que G es k-conexo.

Definición 6. Separación: La separación de un grafo conexo es la descomposición de G en los subgrafos G_1 y G_2 que tienen exactamente un nodo en común, y no hay contenencia entre ellos. Al nodo común se le llama nodo separador. Se denota $\{V(G_1), V(G_2)\}$.

También será necesario demostrar un par de propiedades que servirán para demostrar el teorema de Kuratowski – Wagner.

Corolario 2: Un grafo planar no tiene a K_5 ni a $K_{3,3}$ como *minor* topológico.

Dem: Por el corolario 1, sabemos que en un grafo planar $E \leq 3V - 6$. Para K_5, tenemos que $E = 10$ y $V = 5$, por lo tanto, no se cumple la anterior desigualdad pues $10 > 9$ y así no es planar. Ahora para $K_{3,3}$ suponga que es planar y, como todos los ciclos de este grafo tienen longitud mayor o igual a 4, se cumple que $F \leq \frac{2}{4}E$, pues cada arco limita con por lo menos 2 caras y hay mínimo 4 arcos rodeando una cara. Así se debería cumplir que $E \leq 2V - 4$ pero se tiene que $9 > 8$. Además, cualquier subdivisión de K_5 ni de $K_{3,3}$ tampoco pueden ser planares pues siguen teniendo la misma forma.

■

Lema 1: Un grafo G tiene a K_5 o a $K_{3,3}$ como *minor* sí y sólo si contiene a K_5 o a $K_{3,3}$ como *minor* topológico.

Dem: (⇒) Por un lado, si el grado máximo de un grafo es menor o igual a 3 entonces se cumple que para todo *minor* de grado máximo menor o igual que 3 también es *minor* topológico. De esta manera basta con mostrar que todo grafo G con *minor* K_5 tiene K_5 como *minor* topológico o tiene $K_{3,3}$ como *minor*. Suponga entonces que G tiene *minor* K_5, y sea K el modelo mínimo de K_5 en G. Luego cada conjunto de arcos de K induce un árbol en K, y entre cualesquiera dos de dichos conjuntos hay exactamente un arco. Tome entonces el árbol inducido de alguno de estos conjuntos V_x y le agregamos los 4 arcos que lo conectan a los otros conjuntos obtenemos un nuevo árbol, llámelo T_x. Así, T_x tiene exactamente 4 hojas (los 4 conjuntos vecinos).

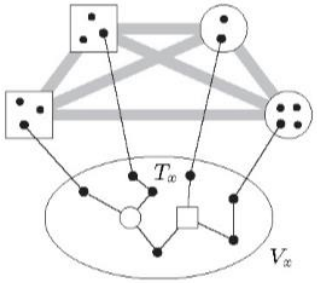


Ilustración 5 (Diestel, 1997)

Si cada uno de los T_x que se forma a partir de los conjuntos es una subdivisión de $K_{1,4}$, entonces K es subdivisión de K_5 y así G contiene a K_5 como *minor* topológico. Si uno de los T_x no es subdivisión de $K_{1,4}$, luego tiene exactamente 2 vértices de grado 3. Contraemos entonces el respectivo V_x hacia los 2 vértices de grado 3, y todos los otros conjuntos de arcos los contraemos hacia un vértice cada uno, para así obtener un grafo de 6 vértices conteniendo a $K_{3,3}$. Por lo tanto, $K_{3,3}$ es *minor* de G.

(⇐) Trivial, pues un *minor* topológico siempre es un *minor.*

■

Lema 2: Todo grafo G 3-conexo sin K_5 ni $K_{3,3}$ como *minors* es planar.

Dem: Se hará por inducción en $|G|$. Para $|G| = 4$ tenemos $G = K_4$ y la afirmación es válida. Ahora tome $|G| > 4$, y asuma que la afirmación es válida para todo grafo menor. Como G es 3-conexo, G tiene un eje xy tal que G contraído a xy (G/xy) es otra vez 3-conexo. Como la relación *minor* es transitiva, G/xy no tiene K_5 ni $K_{3,3}$ como *minor*. Por la hipótesis de inducción sabemos entonces que G/xy es planar, y tome D su dibujo. Sea f la cara de $D - v_{xy}$ que contiene el punto v_{xy}, y sea C el borde de f. Sean los conjuntos $X = \{los\ nodos\ en\ G\ con\ conexión\ a\ x\}\backslash\{y\}$ y $Y = \{los\ nodos\ en\ G\ con\ conexión\ a\ y\}\backslash\{x\}$. Luego $X \cup Y \subseteq V(C)$, pues $v_{xy} \in f$. Entonces

$$D' = D - \{v_{xy}v \mid v \in Y\backslash X\ \}$$

Puede ser visto como el dibujo de $G - y$, donde el vértice x es representado por el punto v_{xy}. Falta agregar el nodo y al dibujo para obtener el dibujo de G.

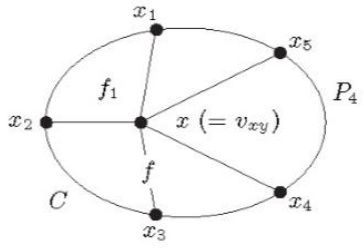


Ilustración 6: (Diestel, 1997)

Como D es 3-conexo, $D - v_{xy}$ es 2-conexo, y así C es un ciclo. Sean $x_1, \dots, x_k$ una enumeración de los vértices en X del ciclo, y sean $P_i = x_i \dots x_{i+1}$ los caminos en C entre ellos ($i = 1, \dots, k$; con $x_{k+1} = x_1$). Veamos que $Y \subseteq V(P_i)$ para algún i. Suponga lo contrario. Si y tiene un vecino $y' \in P'_i$ para algún i, tiene entonces otro vecino $y'' \in C - P_i$ y estos están separados en C por $x' = x_i$ y $x'' = x_{i+1}$. Si $Y \subseteq X$ y $|Y \cap X| \leq 2$, entonces y tiene exactamente dos vecinos y', y'' en C pero no en el mismo P_i, entonces y' y y'' están separados en C por dos vértices $x', x'' \in X$. En cualquier caso, x, y', y'' y y, x', x'' son el eje de vértices de una subdivisión $K_{3,3}$ en G, una contradicción. El único caso que faltaría es que y y x tengan tres vecinos en común en C, y por lo tanto tendrían una subdivisión K_5 con x y y, otra contradicción.

Fije i de manera que $Y \subseteq P_i$. El conjunto $C \backslash P_i$ está contenido en una de las dos caras del ciclo $C_i = x x_i P_i x_{i+1} x$; y denotamos la otra cara de C_i con f_i. Como f_i contiene puntos de f (cerca a x) pero no puntos de su borde C, entonces tenemos $f_i \subseteq f$. Es más, el eje $x x_j$ con $j \notin \{i, i+1\}$ se encuentra con C_i solo en x y termina fuera de f_i en $C \backslash P_i$, entonces f_i no se encuentra con ninguno de esos ejes. Así, $f_i \subseteq \mathbb{R}^2 \backslash D'$, es decir, f_i está contenido en (y, por lo tanto, es igual a) una cara de D'. Por lo tanto D' es un dibujo planar de G al agregarle y y sus ejes incidentes en f_i.

■

Lema 3: Sea X un conjunto de grafos 3-conexos. Sea G un grafo de separación propia $\{V_1, V_2\}$ de orden $\kappa(G) \leq 2$. Si G es eje maximal sin *minor* topológico en X, entonces también lo son $G_1 = G[V_1]$, $G_2 = G[V_2]$ y $G_1 \cap G_2 = K_2$.

Dem: Note primero que todos los vértices $v \in (S = V_1 \cap V_2)$ tiene un vecino en cada componente de $G_i - S, i = 1,2$, pues si no fuese así $S \backslash \{v\}$ separaría G, contradiciendo $|S| = \kappa(G)$. Por la maximalidad de G, cada arco e agregado a G esta en alguna subdivisión de $T \subseteq G + e$ con $T \in X$. Para todas las opciones de e (que se mostrarán a continuación), la 3-conexividad de T implica que la rama de vértices de esta subdivisión de T pertenecen al

mismo V_i, sin pérdida de generalidad, asuma que a V_1. Luego dicha subdivisión se encuentra con V_2 en un camino P que corresponde a un eje de T.

Si $T = \emptyset$, obtenemos una contradicción al escoger e con un extremo en V_1 y el otro en V_2. Si $S = \{v\}$, sea e la conexión entre un vecino v_1 de v en $V_1 \backslash S$ y un vecino v_2 de v en $V_2 \backslash S$. Luego P contiene a v y al eje $e = v_1v_2$; y reemplazando el segmento vPv_1v_2 con el eje vv_1 obtenemos una subdivisión de T en $G_1 \subseteq G$, una contradicción.

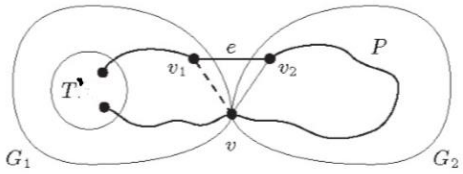


Ilustración 7: (Diestel, 1997), donde T' es la subdivisión de T

Entonces $|S| = 2$, sea $S = \{x, y\}$. Si $xy \notin G$, tomamos $e = xy$, y en la subdivisión de T reemplazamos e por cualquier camino de x a y que pase por G_2. Esto produce una subdivisión de T en G, contradicción. Entonces $xy \in G$, y $G[S] = K_2$.

Falta mostrar entonces que G_1 y G_2 son eje maximales sin *minor* topológico en X. Sea e' un eje adicional para G_1, sin pérdida de generalidad. Reemplazando xPy con xy de ser necesario, obtenemos una subdivisión de T en $G_1 + e'$ (lo cual muestra la maximalidad de G_1), o en G_2 (lo cual contradice $G_2 \subseteq G$).

■

Lema 4: Si $|G| \geq 4$ y G es eje maximal sin subconjuntos que sean subdivisiones de K_5 ni de $K_{3,3}$, entonces G es 3-conexo.

Dem: Se hará por inducción en $|G|$. Para $|G| = 4$ tenemos $G = K_4$ y la afirmación es válida. Ahora tome $|G| > 4$, y sea G arco-maximal sin algún subgrafo subdivisión de K_5 ni de $K_{3,3}$. Suponga que G tiene $\kappa(G) \leq 2$, y tome G_1 y G_2 como en el lema 3. Para $X = \{K_5, K_{3,3}\}$, el lema afirma que $G_1 \cap G_2$ es K_2, con vértices x, y, y que G_1 y G_2 son eje maximales sin subgrafo subdivisión de K_5 ni de $K_{3,3}$. Entonces G_1 y G_2 son un triángulo o 3-conexos por hipótesis de inducción. Como no pueden contener K_5 ni $K_{3,3}$, incluso como *minor*, son planares por el lema 2. Para cada $i = 1, 2$ tome un dibujo de G_i, una cara f_i con el eje xy en su frontera, y un vértice $z_i \neq x, y$ en la frontera de f_i. Sea K una subdivisión de K_5 o de $K_{3,3}$ en el grafo abstracto $G + z_1z_2$.

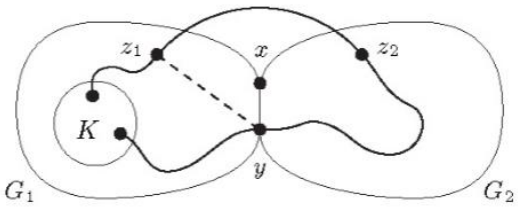


Ilustración 8: (Diestel, 1997)

Si todos los vértices ramas de K pertenecen al mismo G_i, entonces $G_i + xz_i$ o $G_i + yz_i$ contiene una subdivisión de K_5 o de $K_{3,3}$ contradiciendo el corolario 2 pues los grafos son planares por la escogencia de z_i. Como $G_i + z_1z_2$ no contiene cuatro caminos independientes entre $(G_1 - G_2)$ y $(G_2 - G_1)$, entonces estos subgrafos no pueden ambos contener un vértice rama subdivisión de K_5, y no pueden contener dos vértices ramas subdivisiones de $K_{3,3}$. Por lo tanto K es subdivisión de $K_{3,3}$ con solo un vértice rama v en, digamos, $(G_2 - G_1)$, pero luego también el grafo $G_1 + v + \{vx, vy, vz_1\}$,el cual es planar por la escogencia de z_1, contiene una subdivisión de $K_{3,3}$. Esto también contradice al corolario 2.

■

Con lo anterior, ya podemos enunciar el teorema de Kuratowski – Wagner.

Teorema 1. (Kuratowski – Wagner): Las siguientes afirmaciones son equivalentes:

i) G es planar;
ii) G no contiene a K_5 (grafo completo de 5 vértices) ni a $K_{3,3}$ (grafo bipartito completo de 6 vértices) como *minors*;
iii) G no contiene a K_5 ni a $K_{3,3}$ como *minors* topológicos.

Dem: A partir del corolario 2, y los lemas 1, 2 y 4.

■

Implementación

La teoría expuesta nos permite entonces tener una idea a grandes rasgos sobre lo que hay que hacer. Primero se verifica que el grafo cumpla la desigualdad $E \leq 3V - 6$, y si no se cumple se descarta inmediatamente que sea planar. Si no, es necesario buscar si el grafo tiene a K_5 o a $K_{3,3}$ como *minors* o intentar dibujarlo a partir de algún ciclo. Lo primero es sencillo y rápido, y permite descartar una buena cantidad de grafos sin consumir muchos recursos. Lo segundo es mucho más complejo para programar y, si no se hace bien, costoso en recursos.

Existen varios algoritmos para realizar esto, y los mejores de ellos lo hacen en tiempo lineal en el caso promedio. El más efectivo es el de Hopcroft-Tarjan. Su complejidad es $O(|V|)$, aunque su complejidad para programarlo y entenderlo es mayor.

Buscando en la bibliografía alternativas más sencillas para implementar, me topé con el algoritmo de DMP (Gould, 2012). Es uno de los más sencillos de entender, así que decidí implementarlo. La bibliografía nos da a grandes rasgos una descripción de pasos generales que realiza el algoritmo, que explicaré a continuación.

Primero se recomienda realizar un pre-procesamiento. Este consiste en los pasos 1 al 5 (los pasos 0 y 6 son propuestas mías):

0. Si $V \leq 4$, el grafo es planar. Esto es evidente por el teorema de Kuratowski-Wagner.
1. Si $|E| > 3V - 6$, entonces el grafo no es planar.
2. Si el grafo es disconexo, considerar cada componente por aparte.
3. Si el grafo contiene un nodo de corte, es planar sí y sólo si cada uno de los bloques es planar.
4. Ciclos y múltiples arcos no cambian nada.
5. Realizar contracción de todos los nodos que tengan grado 2. Yo contraje también los de grado 1, pues en un dibujo siempre va a poder pegarse lo más que se pueda al nodo al que está conectado y así parecer uno solo.
6. Repetir pasos 0 y 1, pues la contracción modifica el número de arcos y de ejes.

En resumidas cuentas, el algoritmo intenta construir el grafo planar a partir de un ciclo cualquiera del grafo. A este ciclo le va agregando los arcos y nodos faltantes teniendo en cuenta la distribución de las caras, y así, si no es posible agregar un arco, logra confirmar o descartar que el grafo sea planar.

A continuación, se muestran los pasos que describe la bibliografía:

Algoritmo DMP

Entrada: Grafo pre-procesado.

Salida: Verdadero si es planar, falso de lo contrario.

Estrategia: Ir agregando ejes a partir de un ciclo C.

1. Buscar el ciclo C y guardarlo como H_1. Inicializar $i = 1$ y $r = 2$, donde r nos muestra el número de caras que lleva H_i e i nos dice el número de la iteración.
2. Si $r = |E| - N + 2$ (característica de Euler), parar pues significa que el grafo H ya tiene su máximo de caras. Si no, determine todos los segmentos S de H en G, donde un segmento es el conjunto de arcos y nodos que inician en un nodo en H y terminan en otro nodo diferente en H y están en G. También que sólo comparten dos nodos con H. Para cada segmento determine las posibles caras en donde se puede ubicar en H.
3. Si existe algún segmento que no tenga caras posibles, parar y decir que G no es planar. Si no, pero existe un segmento que tenga sólo una cara posible para ser ubicado, escoger dicho segmento para el siguiente paso. Si no, escoger cualquiera.

4. Encontrar un camino P en S que conecte los dos vértices en H. Una H_i con P para obtener H_{i+1}. Tenga en cuenta la cara en la que se ubica P.
5. Actualizar las variables $i = i + 1$, y $r = r + 1$. Repetir desde el paso 2.

Como se puede observar, la bibliografía da una descripción general y muy sencilla de entender, pero a la hora de implementar surgen miles de dudas sobre cómo hacerlo en concreto. ¿Cómo representar una cara? ¿Qué estructura utilizar para el grafo? Esas y otras dudas me surgieron durante la implementación. Además, no logré encontrar ninguna implementación o pseudocódigo más específico de éste algoritmo, así gran parte del código lo implementé como quisiera. Logré basarme en distinta bibliografía para esto, como el libro de Algoritmos de Sedgewick para DFS y BFS (las diapositivas de la clase también me ayudaron), y unas diapositivas de la universidad de Washington para el algoritmo que encuentra el nodo de corte.

Mi implementación, que se encuentra en una carpeta anexa, no se centró en la eficiencia pues el algoritmo implicaba mucho trabajo. Entonces, para tenerlo listo lo más pronto posible, muchas cosas las implementé de una manera que seguramente no eran las más eficientes. El tiempo y la extensión del algoritmo no me permiten demostrar su corrección, y de hecho creo que es posible que aún hay errores, pero unos casos base interesantes funcionaron bien (estos casos también irán adjuntos). Cabe resaltar que, para ser un problema de maratones, la solución es mucho más larga de lo que se esperaría.

Para la solución construí 5 clases en Java que se describen a continuación.

- DMP.java: Ésta clase hacía los pasos descritos anteriormente. Para las operaciones más complejas hacía llamados a las otras clases.
- Main.java: Esta clase se encarga de leer el input y crear el grafo para pasárselo a DMP, y así imprimir si el grafo es planar o no.
- SimpleGraph.java: Esta clase representa un grafo mediante listas de adyacencias. Además de las operaciones básicas, tiene otras como contraerse y obtener los componentes conexos.
- RegionGraph.java: Esta clase representa un grafo. Extiende de SimpleGraph, así que tiene todas sus funciones, pero adicionalmente tiene en cuenta las caras. Para éstas, utilicé listas de nodos, donde una cara es representada por los nodos que la rodean.
- SpanningTree.java: Clase que representa un spanning tree. Se utiliza cuando un grafo quiere calcular su nodo de corte.

Los casos de prueba que se incluyen son los siguientes:

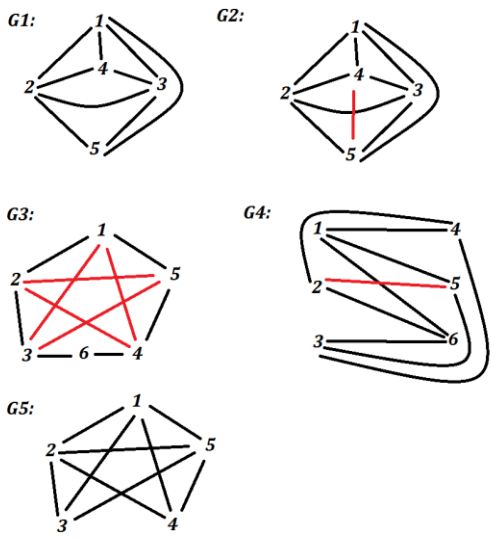


Ilustración 9

Los grafos G1 y G2 corresponden a los casos de prueba propuestos en el enunciado. G3 y G4 corresponden a K_5 con una subdivisión en el eje $3 \rightarrow 4$ y $K_{3,3}$ respectivamente, los cuales sabemos que no son planares. El último caso es $K_5 - \{3 \rightarrow 4\}$, el cual es planar (no es tan evidente, aunque por el teorema de Kuratowski-Wagner si lo es).

Instrucciones de ejecución

La carpeta Proyecto Grafos contiene la siguiente estructura:

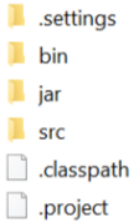


Ilustración 10

Donde las carpetas importantes son jar y src. La carpeta src contiene el código fuente del programa. La carpeta jar contiene un .jar que permite ejecutar el programa y el archivo test.in con los casos descritos anteriormente. Se puede ejecutar de dos maneras. Por un lado, se puede importar el proyecto a Eclipse y correrlo desde ahí. Por otro, se puede abrir la consola de comando en la carpeta jar (abrir la carpeta, hacer clic derecho mientras se mantiene oprimida la tecla *shift*, y seleccionar "abrir ventana de comandos aquí") y ejecutar el comando "java -jar PlanarTest.jar < test.in".

Ilustración 11

Si no se quiere ejecutar las pruebas predeterminadas, sino insertar otras manualmente, simplemente omitir el "< test.in". Para utilizar el programa se siguen las reglas expuestas por el enunciado del problema. Es decir, la primera línea debe ser el número de estaciones de metro que hay (N). Luego siguen N líneas donde cada una representa las conexiones directas que tiene a otra estación, en orden ascendente y separadas por un espacio sencillo. La línea i-ésima corresponde a la i-ésima estación. Al terminar de escribir la última línea se imprime en la consola 'Y' si es posible construir el metro, o 'N' si no. El programa sigue esperando nuevos casos, así que se puede seguir ingresando más. Para salir del programa oprimir las teclas Ctrl+C.

Reflexión final

Para cerrar el proyecto haré una reflexión final. *A subway for Boroughghraph* es un problema que a simple vista parece sencillo, pero mirándolo a fondo se descubre su complejidad. No por nada nadie lo ha resuelto en UVA y no se encuentran soluciones de este problema en internet. La teoría detrás para resolverlo es extensa y muy interesante, la cual se puede explorar más a fondo. Si bien existen algoritmos eficientes para resolverlo, es difícil encontrar alguna implementación y si no se implementa con cuidado puede que el algoritmo llegue a ser menos óptimo de lo esperado. Explorar de qué otras maneras se pueden implementar dichos algoritmos para hacerlos lo más eficientes es algo en lo que se podría indagar y así llegar cada vez más a mejores soluciones.

Por ahora quedo satisfecho con lo que aprendí y, aunque no alcancé a demostrar la corrección de mi algoritmo (y estoy casi seguro que más de un error de implementación se me debió escapar), estoy conforme con que mi algoritmo haya resuelto de manera correcta los casos de prueba presentados.

Bibliografía

A Subway for Boroughgraph. (n.d.). Recuperado el 10 de noviembre del 2016, de https://uva.onlinejudge.org/external/129/12941.pdf

Diestel, R. (1997). *Graph theory* (5th ed.).

Gould, R. (2012). Chapter 6 Planarity. En *Graph Theory* (pp. 171-196).

Holder, L. (n.d.). *Graph Algorithms: Applications*. Recuperado el 30 de noviembre del 2016, de http://www.eecs.wsu.edu/~holder/courses/CptS223/spr08/slides/graphapps.pdf

Sedgewick, R., & Wayne, K. D. (2011). *Algorithms, Fourth Edition*. Addison-Wesley Professional.

Sheffer, A. (2015, Enero 28). *Ma/CS 6b - Class 10: Kuratowski's Theorem*. Recuperado el 29 de noviembre del 16, de www.pma.caltech.edu

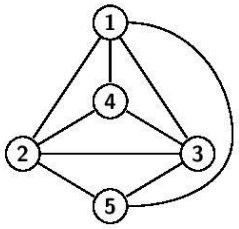
Anexo

B: A Subway for Boroughgraph

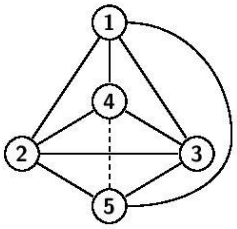
Source file name: `subway.c`, `subway.cpp`, *or* `subway.java`

The mayor of Boroughgraph has been promising its inhabitants a new subway system for quite a few years, and the people are growing impatient. With barely enough money to complete the construction and even less time, he has hired you, a brilliant Boroughgraphian programmer, to help him with this project.

The mayor, of course, wants the new subway to favor mostly those whom he thinks will vote for him, so he has already determined the subway plan. The only problem is that Boroughgraph was built over a series of swamps, so the subway tunnels can be drilled only at a certain fixed depth below ground without risking collapse. Since there is no leftover money to invest in overpasses or ground-level lines, two subway lines can never intersect (except possibly at their endpoints), but their paths need not be straight.



The subway can be built



The subway cannot be built

This is where you come in. Given a specification of a subway network, can you write a program that tells the mayor whether it is possible to build it in Boroughgraph?

Input

The input consists of several subway network specifications. Each specification begins with a line containing a single integer N indicating the number of subway stations ($2 \leq N \leq 64$), which are numbered from 1 to N. Then follow N lines indicating the layout of the subway system: line i contains exactly d_i blank-separated integers from 1 to N (excluding i), indicating the stations to which station i should be connected ($1 \leq d_i \leq N-1$). The subway network is bidirectional, so if station i appears in station j's line, then it is guaranteed that station j will appear in station i's line.

The input must be read from standard input.

Output

For each specification, print a line with the character '`Y`' if the subway can be built in Boroughgraph, or with the character '`N`', otherwise.

The output must be written to standard output.

Sample Input	**Sample Output**
5 2 3 4 5 1 3 4 5 1 2 4 5 1 2 3 1 2 3 5 2 3 4 5 1 3 4 5 1 2 4 5 1 2 3 5 1 2 3 4	Y N