Programación II

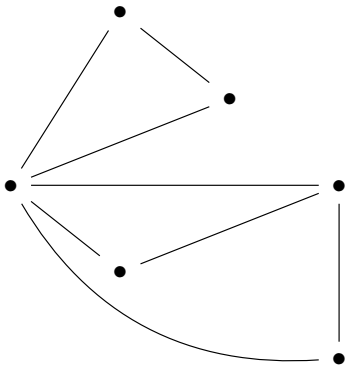
Tecnicatura Universitaria en Inteligencia Artificial

2022

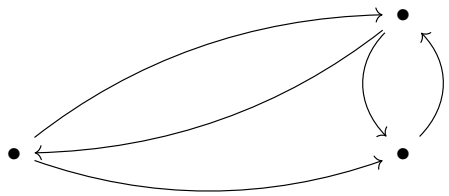
Teoría de grafos

1 Introducción

Definición Un **grafo**[1] G consiste en un conjunto de vértices[2] y un conjunto E de aristas[3] tal que cada arista $e \in E$ se asocia con un **par no ordenado** de vértices. Si la arista e está asociada con los vértices u y w, se escribe $e = (v, w)$ o $e = (w, v)$. Es importante notar que ambas son exactamente la misma arista. El siguiente ejemplo muestra un grafo.



Definición Un **grafo dirigido** G consiste en un conjunto V de vertices y un conjunto E de aristas tal que cada arista $e \in E$ se asocia con un **par ordenado** de vértices. Si la arista e esta asociada con los vertices u y w, se escribe $e = (v, w)$. Es importante considerar que en este contexto que las aristas (v, w) y (w, v) representan cosas distintas. El siguiente ejemplo muestra un grafo dirigido.



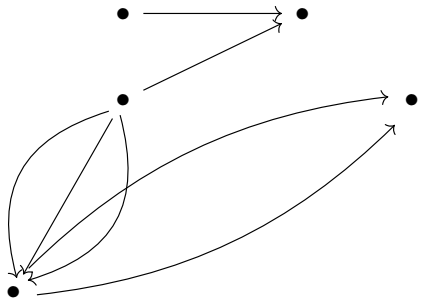
[1] o grafo no dirigido, llamado también grafo simple.
[2] tambien llamadas nodos.
[3] tambein llamadas arcos

Definición Se dice que una arista e en una gráfica (no dirigida o dirigida) que se asocia con el par de vértices v y w es *incidente* sobre v y w, y se dice que v y w son incidentes sobre e y son vértices **adyacentes**[4].

Si G es una gráfica (no dirigida o dirigida) con vértices V y aristas E, se escribe G = (V, E). Usualmente, se pide que los conjuntos E y V sean finitos y que V sea no vacío.

Definición Sea V un conjunto finito y no vacio. Decimos que G = (V, E) es un **multigrafo** si, para algunos vertices, existen dos o mas aristas en E que conectan los mismos vertices. Es decir, E es un multiconjunto. La definición de **multigrafo dirigido** se hace de la misma forma, agregando dirección a las aristas.

El siguiente ejemplo muestra un multigrafo dirigido



2 Caminos y Ciclos

Ejemplo La figura 1 muestra parte del sistema de carreteras de Wyoming. El inspector de carreteras debe reccorer estas ciudades, inspeccionando los caminos y rutas entre ellas, para poder dar un dictamen sobre el estado de las mismas. Como el inspector vive en Greybull, la forma mas económica de inspeccionar todos los caminos sería comenzar en Greybull, recorrer cada camino exactamente una vez y regresar de nuevo al punto de partida.

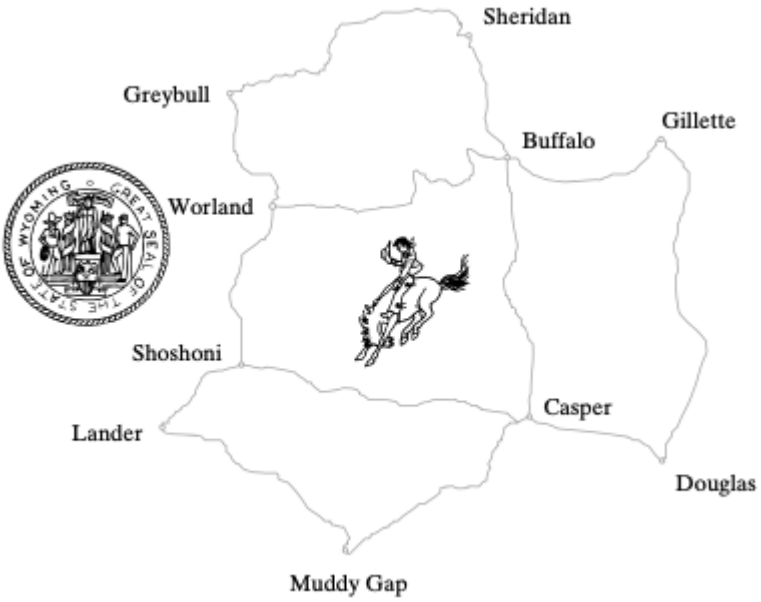
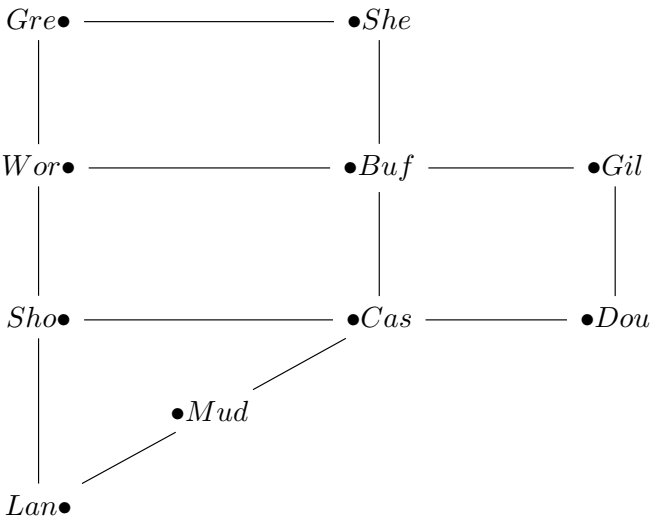


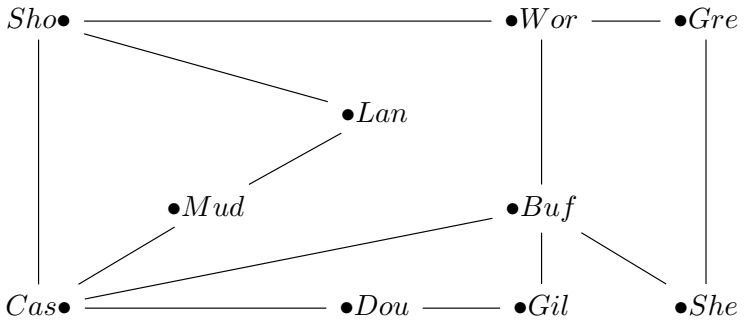
Figure 1: Parte del sistema de carreteras de Wyoming

[4] o *vecinos*

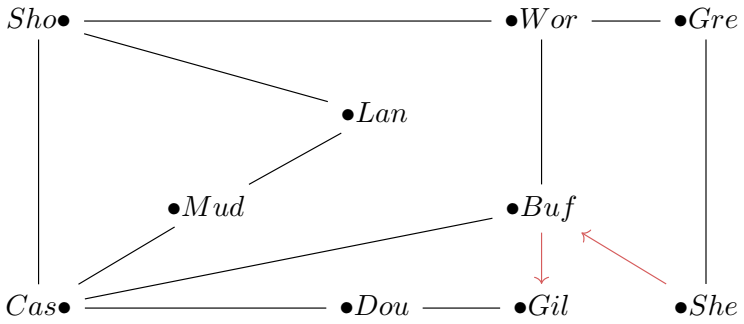
Este problema se puede modelar como un **grafo**. Los grafos se dibujan con líneas y puntos, así que realmente parecen mapas de carreteras. Los puntos se llaman **vértices** o **nodos**, mientras que las lineas se llaman **aristas**. Cada vértice fue nombrado con las primeras tres letras de la ciudad que representan.



Lo primero que debe notarse es que en un grafo solo importan que nodos estan conectados a cuales otros, por lo cual es indistinto como se lo dibuje. Por ejemplo, el siguiente dibujo da una representación alternativa, pero igualmente valida del problema de las carreteras de Wyoming.



Si se inicia en el vértice v0, se viaja por una arista al vértice v_1, por otra arista al vértice v_2, etcétera, y con el tiempo se llega al vértice v_n; este viaje completo recibe el nombre de **camino**[5] de v_0 a v_n. Por ejemplo, el camino que comienza en She, va Buf y termina en Gil aparece resaltado en el siguiente dibujo:



[5] A veces también llamado trayectoria o ruta.

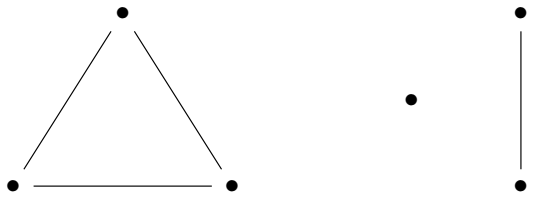
Llamamos **longitud** del camino a la cantidad de aristas que recorre. En el ejemplo, vemos un camino de longitud 2. Es importante notar que existen "caminos" de longitud 0, basta con elegir cualquier vértice y no utilizar ninguna arista.

Observe que la definición de camino permite tener aristas repetidas. Llamaremos **camino simple** a un camino donde no hay aristas repetidas. Además, llamamos **ciclo** a un camino que empieza y termina en el mismo vértice. Un **ciclo simple** es un ciclo que no repite aristas.

El problema del inspector de caminos de Wyoming puede escribirse formalmente como: ¿Existe un camino desde Greybull hacia Greybull que pase por cada arista exactamente una vez?

Es posible demostrar que el inspector de carreteras no puede comenzar en Greybull, viajar por cada tramo de camino justo una vez y regresar a Greybull. O bien, en términos de teoría de grafos, no existe un camino del vértice Gre al vértice Gre que recorra todas las aristas exactamente una vez. Para ver esto, suponga que existe tal trayectoria y considere el vértice Wor. Cada vez que se llega a Wor por alguna arista, se debe salir de Wor por una arista diferente. Más aún, cada arista que toca Wor se debe usar. Entonces las aristas en Wor ocurren en pares. Se concluye que un número par de aristas debe tocar Wor. Como tres aristas tocan a Wor, se tiene una contradicción. Por lo tanto, no existe una trayectoria del vértice Gre al vértice Gre en el grafo. El argumento se aplica a una gráfica arbitraria G. Si G tiene una trayectoria del vértice v al vértice v que recorre todas las aristas exactamente una vez, un número par de aristas deben tocar cada vértice.

Definición Un **grafo conexo** es un grafo donde, dado cualesquiera dos vértices, siempre existe un camino que comienza en uno y termina en el otro. Todos los ejemplos que grafos que hemos visto hasta ahora son conexos. A continuacion, se da un ejemplo de un grafo **no conexo**.



Intuitivamente, un grafo conexo es aquel "de una sola pieza". Mientras que los grafos no conexos parecen, a simple vista, estar formado por varias partes. A cada una de estas "partes" se la llama **componente conexa**. Se nota con $\kappa(G)$ a la cantidad de componentes conexas del grafo. En el ejemplo, $\kappa(G) = 3$.

Definición Sea $G = (V, E)$ un grafo. $G' = (V', E')$ es un subgrafo de G si se cumple:

1. $V' \subset V$
2. $E' \subset E$
3. Para toda arista $e \in E'$, si e incide en los vértices v y w, entonces $v, w \in V'$

Esta definición nos dice que un grafo es subgrafo de otro cuando esta compuesto por algunos de sus vértices (posiblemente todos), algunas de sus aristas (posiblemente todas) y añadiendo la condición especial de que todas las aristas del subgrafo deben tener sus correspondientes vértices en el subgrafo.

Definición Sea $G = (V, E)$. Si $U \subset V$, el **subgrafo de G inducido por** U es el subgrafo cuyo conjunto de vértices es U y que contiene todas las aristas de G de la forma (v, w) donde $v, w \in U$.

Definición Un subgrafo G' de un grafo $G = (V, E)$ es un **subgrafo inducido** si existe un conjunto $U \subset V$ tal que G' es el subgrafo de G inducido por U.

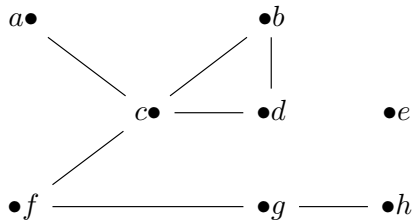
Definición Sea v un vertice en un grafo (dirigido o no dirigido) $G = (V, E)$. El subgrafo de G denotado por $G - v$ tiene el conjunto de vertices $V' = V - \{v\}$ y el conjunto de aristas $E' \subset E$, donde E' contiene todas las aristas en E, excepto aquellas que son incidentes en el vértice v.

Observación El grafo $G - v$ es el subgrafo inducido por $V - \{v\}$.

Definición De manera similar, si e es un arista en un grafo (dirigido o no dirigido) $G = (V, E)$, el subgrafo $G - e$ es el subgrafo que contiene todas las aristas de G, excepto e, y el mismo conjunto de vértices.

Definición Sea G un grafo dirigido o no dirigido, o un multigrafo. Para cada vertice v de G, el **grado** de v, denotado por $\delta(v)$ es el número de aristas incidentes en v.

Por ejemplo, en el siguiente grafo, $\delta(a) = 1$, $\delta(b) = 2$, $\delta(c) = 4$, $\delta(d) = 2$, $\delta(e) = 0$, $\delta(f) = 2$, $\delta(g) = 2$ y $\delta(h) = 1$



Si $G = (V, E)$ es un grafo no dirigido o un multigrafo, luego $\sum_{e \in V} \delta(v) = 2|E|$. Esto se debe a que cada arista de la forma $(a, b) \in E$ suma 1 al grado de a y uno al grado de b, es decir, suma 2 a la sumatoria $\sum_{e \in V} \delta(v)$. Entonces $2|E|$ es el resultado de sumar los grados de todos los vértices en V. Esto implica que el número de vértices de grado impar debe ser par.

3 Grafos ponderados

Definición Llamamos **grafo ponderado** o **grafo con pesos** a un grafo $G = (V, E)$ con una función $w : E- > \mathbb{R}$, es decir, un grafo donde a cada arista se le asigna un peso. Notamos $w(i, j)$ al peso de la arista que va de i a j, si esta existe.

Definición En un grafo con pesos, modificamos la definición de **longitud de un camino**. La longitud del camino en un grafo con pesos es la suma de los pesos de las aristas que recorre dicho camino.

Ejemplo A menudo en la manufactura, es necesario hacer agujeros en hojas de metal. Luego se atornillan las componentes a estas hojas de metal. Los agujeros se perforan utilizando un taladro controlado por computadora. Para ahorra tiempo y dinero, el taladro debe moverse tan rápido como sea posible. La figura 3 sirve para ilustrar el problema. Se modelará la situación como un grafo. Los vértices del grafo corresponden a los agujeros. Cada par de vértices se conecta por una arista. En cada arista se escribe el tiempo para mover el taladro entre los hoyos correspondientes.

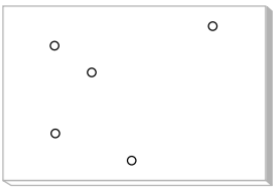
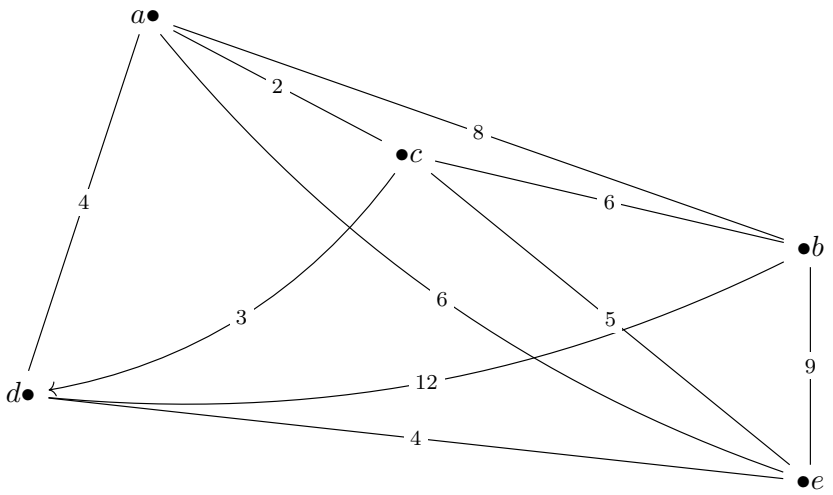


Figure 2:



Primero notemos que este es un problema muy distinto al de las carreteras de Wyoming. Para empezar, ahora tenemos un costo en las aristas que hay que considerar. Además, no estamos interesado en un camino que recorra todas las aristas, si no en un camino que recorra todos los vértices exactamente una vez, además de hacerlo de la forma óptima.

La tabla 1 muestra todas las rutas que visitan los nodos comenzando por a y terminando en e.

Camino	Costo
a, b, c, d, e	21
a, b, d, c, e	28
a, c, b, d, e	24
a, c, d, b, e	26
a, d, b, c, e	27
a, d, c, b, e	22

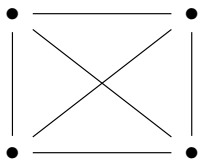
Table 1: Costos de recorrer todos los nodos comenzando en a y terminando en e

Se ve que la ruta que visita los vértices a, b, c, d, e, en ese orden, tiene longitud mínima. Por supuesto, un par diferente de vértices de inicio y terminación produciría una ruta aún más corta. Numerar todas las trayectorias posibles es un método muy ineficiente para encontrar el camino de longitud mínima que visita todos los vértices exactamente una vez. Por desgracia, nadie conoce un método que sea mucho más práctico para grafos arbitrarias. Este problema es una versión del problema del agente viajero. Este problema se investiga actualmente para producir mejores aproximaciones a la respuesta, dado que la respuesta óptima es muy difícil de conseguir.

4 Grafos especiales

Algunos grafos aparecen con tal frecuencia que se les ha dado un nombre.

Definición El grafo completo de n vértices, notado como K_n, es el grafo simple con n vértices y una arista entre cada par de vértices distintos. A modo de ejemplo, se muestra el grafo K_4.



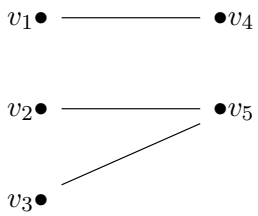
Definición Un grafo $G = (V, E)$ es un **bipartito** si existen subconjuntos V_1 y V_2 de V tales que:

1. $V_1 \cap V_2 = \emptyset$

2. $V_1 \cup V_2 = V$

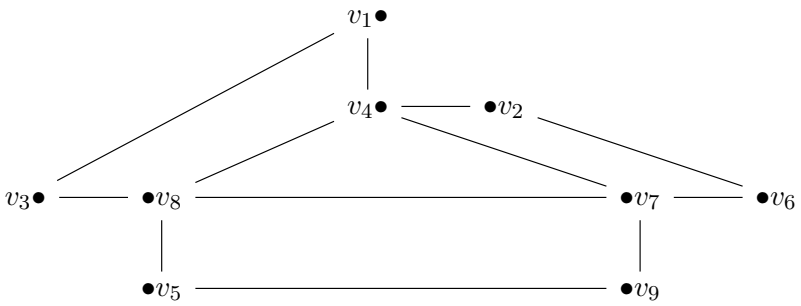
3. Cada arista en E es incidente en un vértice V_1 y un vértice en V_2.

Por ejemplo, el grafo de la siguiente figura es bipartito con $V_1 = \{v_1, v_2, v_3\}$ y $V_2 = \{v_4, v_5\}$.

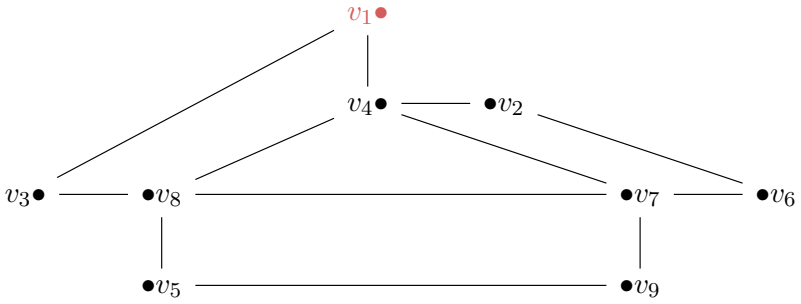


Para identificar si un grafo es bipartito, conviene proceder por *coloreo* de dos colores de los vértices del grafo. La idea es pintar los vértices vecinos del grafo con colores progresivamente. Si al finalizar hemos pintado todo el grafo, concluimos que es bipartito, y los colores nos dan la asignación de V_1 y V_2 que necesitamos para que se cumpla la definición. Si, por el contrario, en algún momento encontramos un vértice que no podemos pintar de ninguno de los dos colores (porque tiene vecinos de ambos colores), hemos de concluir que el grafo no es bipartito.

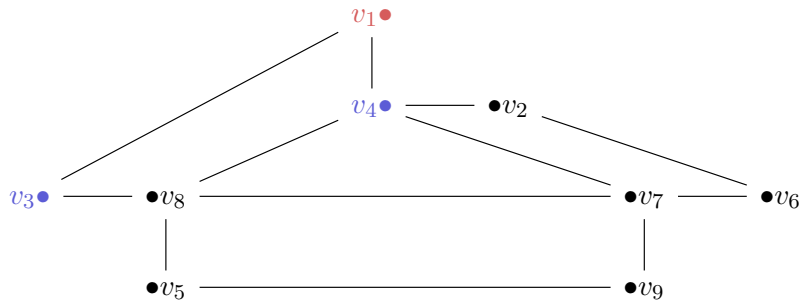
Utilicemos esta idea para decidir si el siguiente grafo es bipartito:



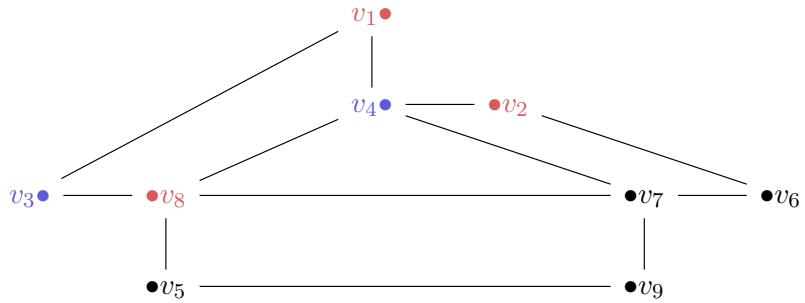
Comenzamos pintando cualquier vértice de un color.



En un segundo paso, pintamos todos sus vecinos de un color distinto:



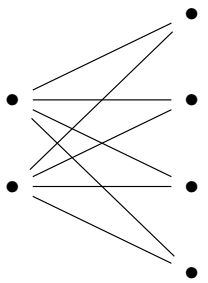
Continuando con este procedimiento, pintamos los vecinos de aquellos que acabamos de pintar del color original



¿Que ocurre con el vértice v_7? Tiene vecinos de ambos colores, por lo tanto, no podemos pintarlo de ningun color. Concluimos que el grafo no es bipartito

Definición El grafo bipartito completo de m y n, denotado $K_{m,n}$ vértices es el grafo simple donde el conjunto de vértices puede particionarse en V_1 con m vértices y V_2 con n vértices, y donde el conjutno de aristas consiste en todas las aristas de la forma (v_1, v_2) con $v_1 \in V_1$ y $v_2 \in V_2$.

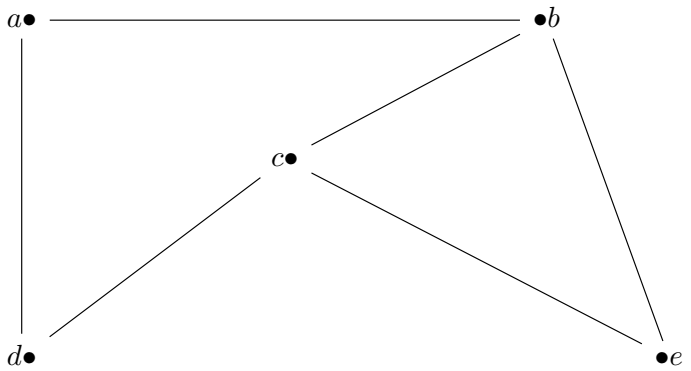
Se muestra como ejemplo el grafo $K_{2,4}$.



5 Representación de grafos

En las secciones anteriores se representó un grafo con un dibujo. En ocasiones, como por ejemplo al usar una computadora para analizar un grafo, se necesita una representación más formal. El primer método de representación de una gráfica usa la matriz de adyacencia.

Considere el siguiente grafo



Para obtener la **matriz de adyacencia**, primero se elige un orden para los vértices, por ejemplo (a, b, c, d, e). Después, se construye una matriz tal que el elemento en la posición i, j tiene un 1 si los vértices en las posiciones i, j son adyacentes y 0 si no. Si $i == j$, ponemos un 0. La matriz de adyacencia para este grafo es:

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{1}$$

Propiedades

- El grado de un vértice v se puede obtener sumando el renglón o la columna que corresponda al vértice v.
- Es una matriz simétrica respecto a la diagonal.
- El elemento i, j de la matriz A^n es igual al número de caminos desde el vértice que corresponde a la posición i al vértice que corresponde a la posición j.

Para obtener la **matriz de incidencia**, se etiquetan los renglones con los vértices y las columnas con las aristas en algún orden arbitrario. El elemento en el renglón v y la columna e tiene un 1 si e es incidente en v, o un 0 si no. La matriz de incidencia para el grafo de ejemplo es:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2}$$

6 El algoritmo de Dijkstra

Un problema muy recurrente es encontrar el camino mas corto entre dos vértices dados (es decir, un camino que tiene la longitud mínima entre todas los posibles caminos entre esos dos vértices.)

Edsger W. Dijkstra (1930-2002) fue un científico de la computación holandés que ideo un algoritmo que resuelve el problema de la ruta más corta de forma óptima. Además, fue de los primeros en proponer la programación como una ciencia. Ganador del premio Turing en 1972. El premio Turing es "el Nobel de la programación". Poco después de su muerte recibió también el premio de la ACM[6]. El premio pasó a llamarse Premio Dijkstra el siguiente año en su honor.

[6] Association for Computing Machinery

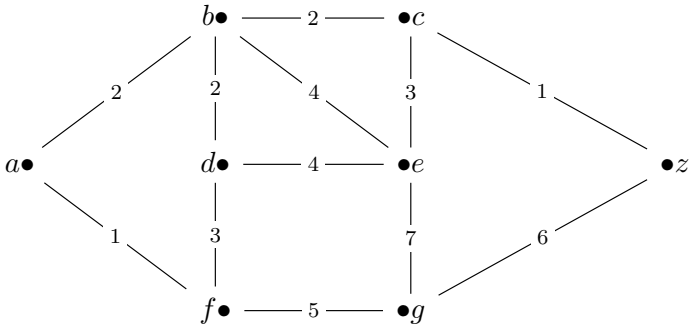
Este algoritmo encuentra la longitud de una ruta más corta del vértice a al vértice z en un grafo $G = (V, E)$ con pesos conexo. El peso de la arista (i, j) es $w(i, j) > 0$ (es decir, requerimos que los pesos sean siempre positivos). Es importante verificar que estas condiciones se cumplen antes de aplicar el algoritmo.

Figure 3: Edsger Wybe Dijkstra
??

El algoritmo Dijkstra funciona asignando **etiquetas** a los vértices. Notaremos a la etiqueta del vértice v como $L(v)$. Algunas etiquetas son temporales y otras son permanentes. Al ilustrar el algoritmo, por lo general se encierran en un círculo, o se pintan de un color distinto los vértices cuya etiqueta es permanente. Llamaremos T al conjunto de vértices que tienen un etiqueta temporal. Esos son los vértices con lo que "tenemos que seguir trabajando". La idea es que, una vez $L(v)$ sea la etiqueta definitiva de un vértice v, $L(v)$ sea la distancia más corta desde a hasta v.
Al inicio, todos los vértices tendrán etiquetas temporales. Cada iteración del algoritmo convierte una etiqueta temporal en una etiqueta definitiva. El algoritmo termina cuando z recibe una etiqueta definitiva. Cuando llega ese momento, $L(z)$ es el valor de la distancia del camino mas corto desde a hasta z. Los pasos detallados son los siguientes:

1. Primero, notemos que la ruta más corta del vértice a al vértice a, es la ruta de cero aristas, que tiene peso 0. Así, inicializamos $L(a) = 0$.
2. No sabemos aún el valor de una ruta más corta de a a los otros vértices, entonces para cada vértice $v \neq a$, inicializamos $L(v) = \infty$.
3. Inicializamos el conjunto T como el conjunto de todos los vértices, i.e. $T = V$.
4. Seleccionamos un vértice $v \in T$ tal que $L(v)$ sea mínimo.
5. Quitamos el vértice v del conjunto T: $T = T - v$
6. Para cada $w \in T$ adyacente a w, actualizamos su etiqueta: $L(w) = \text{MIN}\{L(w), L(v) + w(v, w)\}$
7. Si $z \in T$, repetimos desde el paso 4, si no, hemos terminado y $L(z)$ es el valor de la ruta mas corta entre a y z.

Ejemplo Utilizaremos el algoritmo de Dijkstra para encontrar el tamaño de la ruta más corta de a a z en el siguiente grafo:



Cuando corremos el algoritmo a mano, conviene utilizar una tabla para ir llevando como evolucionan los valores de las etiquetas L a medida que ocurren las iteraciones del algoritmo.

Pondremos en la tabla una columna para llevar el número de iteración en el que nos encontramos y luego una columna por cada vértice. Un valor en la posición (i, j) en la tabla denotara

el valor de la etiqueta L del vértice en la posición j, en la iteración i. Comenzamos escribiendo la primera fila, los valores con los que inicializamos L, para el vértice a, la etiqueta vale 0, y para todos los demás, infinito.

it	$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$	$L(g)$	$L(z)$
	0	∞	∞	∞	∞	∞	∞	∞

Completamos la primera iteración del algoritmo, hay que elegir el vértice que tenga la etiqueta mínima y removerla del conjunto T. En la tabla consideraremos esto con una línea diagonal tachando ese casilla. En este paso, obviamente el valor menor es $L(a) == 0$. Luego actualizamos los valores de las etiquetas para los vértices adyacentes a a, que son los vértices b y f. Para el vértice b, el nuevo valor de la etiqueta es el menor entre la etiqueta que tenia, y lo que resultaría de sumar la etiqueta de a con el valor de la arista (a, b). Del mismo modo, procedemos con la etiqueta del vértice f.

it	$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$	$L(g)$	$L(z)$
	0	∞	∞	∞	∞	∞	∞	∞
1		2	∞	∞	∞	1	∞	∞

En la segunda iteración, elegimos el vértice f. Actualizamos los valores de los vértices d y g (no actualizamos el nodo a porque ya esta "tachado").

it	$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$	$L(g)$	$L(z)$
	0	∞	∞	∞	∞	∞	∞	∞
1		2	∞	∞	∞	1	∞	∞
2		2	∞	4	∞		6	∞

Continuamos realizando las iteraciones del algoritmo hasta que conseguimos tachar el vértice z.

it	$L(a)$	$L(b)$	$L(c)$	$L(d)$	$L(e)$	$L(f)$	$L(g)$	$L(z)$
	0	∞	∞	∞	∞	∞	∞	∞
1		2	∞	∞	∞	1	∞	∞
2		2	∞	4	∞		6	∞
3			4	4	6		6	∞
4				4	6		6	5
5					6		6	5
6					6		6	

La longitud del camino más corto de a a z es 5.

Se presenta el problema de que ahora conocemos el camino mas corto desde a hasta z, pero no un camino de tal longitud. Para conseguir la longitud del camino mas corto junto con un camino de esa longitud puede conseguirse con una versión modificada del algoritmo, donde, además de la distancia del camino mas corto, guardamos en la etiqueta el nodo del cual provenimos.

7 Arboles

Los árboles son un tipo especial de grafo. La definición que dimos en unidades anteriores se puede expresar ahora en términos de teoría de grafos:

Definición Un **árbol** (libre) T es una grafo simple que satisface lo siguiente: si v y w son vértices en T existe un camino único de v a w.

Recordemos también que llamamos **árbol con raíz** a un árbol donde se ha designado un nodo especial, llamado raíz.

Ahora que sabemos teoría de grafos podemos, además, dar algunas propiedades adicionales de árboles.

Propiedad Un grafo conexo y sin ciclos es un árbol. En efecto, como el grafo es conexo entre dos vértices cualesquiera siempre hay un camino.

Propiedad Un grafo conexo con n vértices y $n-1$ aristas es necesariamente un árbol.

Propiedad Si un grafo con n vértices y $n-1$ aristas, no contiene ciclos entonces necesariamente es un árbol.

8 Arboles de expansión

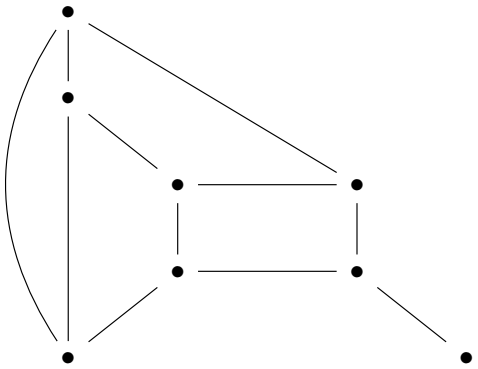
Se dice que árbol T es un **árbol de expansión** de un grafo G si:

1. T es un subrafo de G.

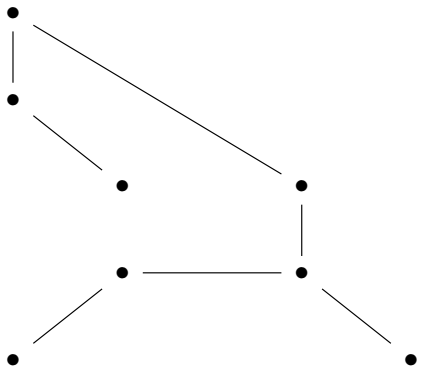
2. T es un árbol.

3. T contiene todos los vértices de G.

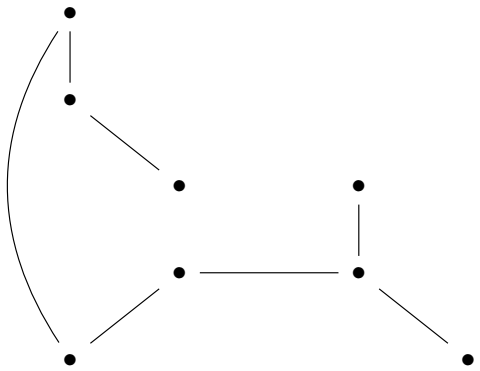
Por ejemplo, dado este grafo



Este es un árbol de expansión de ese grafo:



Nota El árbol de expansión de un grafo en general no es único, por ejemplo, acá mostramos otro árbol de expansión distinto del mismo grafo.



- Un árbol de expansión para un grafo existe si y solo si el grafo es conexo.

En efecto, todos los arboles pueden pensarse como grafos conexos sin ciclos. Si un grafo G tiene un árbol de expansión T, entonces entre dos vértices cualesquiera existe un camino en el árbol que los une (pues todos los arboles son conexos). Ahora bien, como T es subgrafo de G, necesariamente debe existir ese mismo camino en el grafo G.
El algoritmo de búsqueda en profundidad, normalmente llamado DFS por sus siglas en inglés *Depth First Search* permite encontrar el árbol de expansión de un grafo conexo. Este algoritmo puede verse como versión generalizada del recorrido en pre-orden. La estrategia consiste en partir de un vértice determinado v y a partir de allí, cuando se visita un nuevo vértice, explorar cada camino que salga de él. Un camino deja de explorarse cuando se llega a un vértice ya visitado. Si existen vértices no alcanzables, el recorrido queda incompleto; entonces, se debe "volver hacia atrás" hasta encontrar un vértice donde hayamos dejado un camino sin explorar, y repetir el proceso.

Se utiliza, además, un orden sobre los vértices para desempatar en casos donde tengamos más de una opción. Normalmente, como convención interna, utilizaremos el orden alfabético cuando los nodos lleven letras por nombres, o el orden natural cuando los vértices sean numerados. Dado un orden $v_1, v_2, ..., v_n$ de los vértices, el algoritmo puede expresarse formalmente como sigue:

1. Dado el grafo $G = (V, E)$, inicializamos el árbol $T = (V', E')$ con $V' = \{v_1\}$ y $E' = \emptyset$. La idea sera ir agregando aristas a E' a medida que lo necesitemos. Definimos además una variable $w = v_1$ que llevará el nodo donde estamos parados actualmente.

2. Mientras exista v tal que (w, v) es una arista que al agregarla a T no genera un ciclo, realizamos lo siguiente:

   (a) Elegimos la arista (w, v_k) con k mínimo tal que al agregarla a T no genera un ciclo.

   (b) Agregamos la arista (w, v_k) a E'.

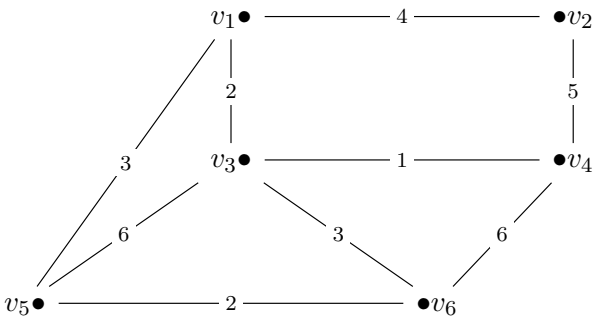
   (c) Agregamos v_k a V'

   (d) Actualizamos $w = v_k$

3. Si $V' = V$ hemos terminado y T y T es un árbol de expansión del grafo G. Si $w = v_1$, el grafo es disconexo, y por lo tanto jamás podremos encontrar un árbol de expansión para el mismo. Si no se da ninguna de las dos situaciones, actualizamos el valor de w para que sea el padre de w en el árbol T, y repetimos desde el paso 2. Dar este paso hacia atrás nos obligará a explorar otros caminos.

9 Árbol de expansión mínima

Ejemplo El grafo con pesos de la figura muestra seis ciudades y los costos de construir carreteras entre ellas. Se desea construir el sistema de carreteras de menor costo que conecte a las seis ciudades. La solución debe necesariamente ser un árbol de expansión ya que debe contener a todos los vértices y para ser de costo mínimo, seria redundante tener dos caminos entre ciudades. Entonces lo que necesitamos es el árbol de expansión del grafo que sea de peso mínimo.



Definición Sea G un grafo con pesos. Un árbol de expansión mínima de G es un árbol de expansión de G que tiene peso mínimo entre todos los posibles.

Nota El algoritmo DFS no asegura que el árbol encontrado sea de peso mínimo.

El algoritmo de Prim permite encontrar un árbol de expansión mínimo para un grafo con pesos conexo de vértices $v_1, v_2, ...v_n$. El algoritmo incrementa continuamente el tamaño de un árbol, de manera similar al algoritmo DFS, comenzando por un vértice inicial al que se le van agregando sucesivamente vértices cuya distancia a los anteriores es mínima. Esto significa que en cada paso, las aristas a considerar son aquellas que inciden en vértices que ya pertenecen al árbol. El árbol está completamente construido cuando no quedan más vértices por agregar.

Definimos $w(i, j)$ como el peso de la arista que une los vértices i, j si existe, o como ∞ si la misma no existe. Además, llevamos la cuenta en un diccionario *agregado* cuyas claves son los vértices y cuyas valores son $True$ si el vértice fue agregado al árbol de expansión mínima, y $False$ si aún no ha sido agregado. También iremos actualizando el diccionario E' de las aristas del árbol. El algoritmo puede ser descrito formalmente como sigue:

1. Inicializamos el diccionario *agregado*, seteando todos los vértices a $False$ (es decir, ningún vértice ha sido agregado aún.)
2. Agregamos el primer vértice al árbol $agregado[v_1] = True$.
3. Inicializamos la lista de aristas que compondrán el árbol como un conjunto vacio: $E = \emptyset$.
4. Para cada i en el rango $1, ..., n-1$, agregamos la arista de peso mínimo que tiene un vértice que ya fue agregado y un vértice que aún no fue agregado, esto lo hacemos del siguiente modo:
   (a) Definimos la variable temporal $min = \infty$.
   (b) Para cada j en el rango $(1, ...n)$:
      i. Si $agregado[v_j] == True$, el vértice v_j ya esta en el árbol:
      ii. Para cada k en el rango de $(1, ..., n)$:
         Si $agregado[v_k] == False$ y además $w(j, k) < min$, el vértice v_k será el *candidato* a ser agregado al árbol, y la arista (j, k) sera la arista candidata a agregar al árbol.
   (c) Al finalizar el for, agregamos el vértice candidato al árbol actualizando el diccionario agregado, y además agregamos la arista candidata al conjunto de aristas.