X

Algoritmos con grafos

May 19, 2023

Contenido

1 Recapitulando

2 Un algoritmo para la ruta más corta

Recapitulando

Definición Un grafo $G = (V, E)$ consiste de un conjunto V de vértices o nodos y un conjunto E de aristas tal que cada arista $e \in E$ se asocia con un par no ordenado de vértices. Normalmente, si e es una arista asociada a los nodos v y w, escribimos simplemente $e = (v, w)$ o $e = (w, v)$, o más simple aún, (v, w). En este contexto, es importante notar que (v, w) y (w, v) son exactamente la misma arista.

Recapitulando

Si $e = (v, w)$ decimos que la arista e es *incidente* sobre v y w, que v y w son incidentes en e y además diremos que los vértices v y w son *adyacentes*.

Recapitulando

Si $e = (v, w)$ decimos que la arista e es *incidente* sobre v y w, que v y w son incidentes en e y además diremos que los vértices v y w son *adyacentes*.

Ayudamemoria Siempre que hablamos de incidencia estamos hablando de una relación entre un vértice y una arista, y siempre que hablamos de una relación de adyacencia, estamos hablando de una relación entre dos vértices. Cuando dos vértices son adyacentes, podemos decir coloquialmente que son **vecinos**.

Recapitulando

Definición Un **grafo con pesos** $G = (V, E)$ consiste de un conjunto V de vértices o nodos y un conjunto E de aristas tal que cada arista $e \in E$ se asocia con un par no ordenado de vértices. Adicionalmente, a cada arista le asignamos un número, que conocemos como el **peso** de la arista.

Recapitulando

Definición El grafo completo de n vértices, denotado por K_n es el grafo con n vértices en la que hay una arista entre cada para de vértices distintos.

Recapitulando

Se dice que un grafo es *bipartito* si existen subconjuntos V_1 y V_2 de V tales que $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ y cada arista $e \in E$ es incidente sobre un vértice de V_1 y sobre un vértice en V_2.

Recapitulando

Se dice que un grafo es *bipartito* si existen subconjuntos V_1 y V_2 de V tales que $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ y cada arista $e \in E$ es incidente sobre un vértice de V_1 y sobre un vértice en V_2.
El grafo bipartito completo de m y n vértices, que notamos $K_{m,n}$ es el grafo donde el conjunto de vértices tiene una partición V_1 con m vértices y V_2 con n vértices y donde el conjunto de aristas consiste de **todas** las aristas de la forma (v_1, v_2) con $v_1 \in V_1$ y $v_2 \in V_2$.

Recapitulando

Un grafo conexo es un grafo del cual podemos ir de cualquier vértice a cualquier otro por un camino, es decir, si dado cualquier par de vértices v y w en G, siempre existe un camino de v a w.

Recapitulando

Un grafo conexo es un grafo del cual podemos ir de cualquier vértice a cualquier otro por un camino, es decir, si dado cualquier par de vértices v y w en G, siempre existe un camino de v a w.
Intuitivamente, un grafo conexo es "de una sola pieza", mientras que un grafo no conexo tiene "varias partes". Cada una de estas partes se llama **componente conexo** del grafo.

Recapitulando

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

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

Recapitulando

Un camino es una sucesión de vértices y aristas dentro de un grafo, que empieza y termina en vértices, tal que cada vértice es incidente en las aristas que le siguen y que le preceden en la secuencia.
Dos vértices están conectados o son accesibles si existe un camino que forma una trayectoria para llegar de uno al otro; en caso contrario, los vértices están desconectados o bien son inaccesible
Un ciclo es un camino de longitud diferente de cero que empieza y termina en el mismo vértice.
Nota Por lo general, estamos interesados en caminos y ciclos **simples** es decir, que no repiten aristas.
En un grafo sin pesos, la longitud de un camino es la cantidad de aristas por las que pasamos. En un grafo con pesos, la longitud del camino es la suma de los pesos de las aristas por las que pasamos.

Recapitulando

El **grado de un vértice** v, que denotamos como $\delta(v)$ es el número de aristas que inciden en v.

Contenido

1 Recapitulando

2 Un algoritmo para la ruta más corta

Un algoritmo para la ruta mas corta

Un grafo con pesos o **grafo ponderado** es un grafo donde asignamos a cada arista un valor numérico.

Un algoritmo para la ruta mas corta

Un grafo con pesos o **grafo ponderado** es un grafo donde asignamos a cada arista un valor numérico.
En un grafo con pesos que la longitud de un camino es la suma de los pesos de las aristas que componen el camino.

Un algoritmo para la ruta mas corta

Un grafo con pesos o **grafo ponderado** es un grafo donde asignamos a cada arista un valor numérico.
En un grafo con pesos que la longitud de un camino es la suma de los pesos de las aristas que componen el camino.
Notaremos $w(i, j)$ el peso de la arista (i, j).

Un algoritmo para la ruta mas corta

Un grafo con pesos o **grafo ponderado** es un grafo donde asignamos a cada arista un valor numérico.
En un grafo con pesos que la longitud de un camino es la suma de los pesos de las aristas que componen el camino.
Notaremos $w(i, j)$ el peso de la arista (i, j). En un grafo con pesos un problema muy común es encontrar el camino más corto entre dos vértices (es decir, un camino que tiene la longitud mínima entre todas los posibles caminos entre esos dos vértices.)

Edsger Dijkstra

Edsger W. Dijkstra (1930-2002) fue un programador 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".

El algoritmo de Dijkstra

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,

El algoritmo de 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 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.

El algoritmo de Dijkstra

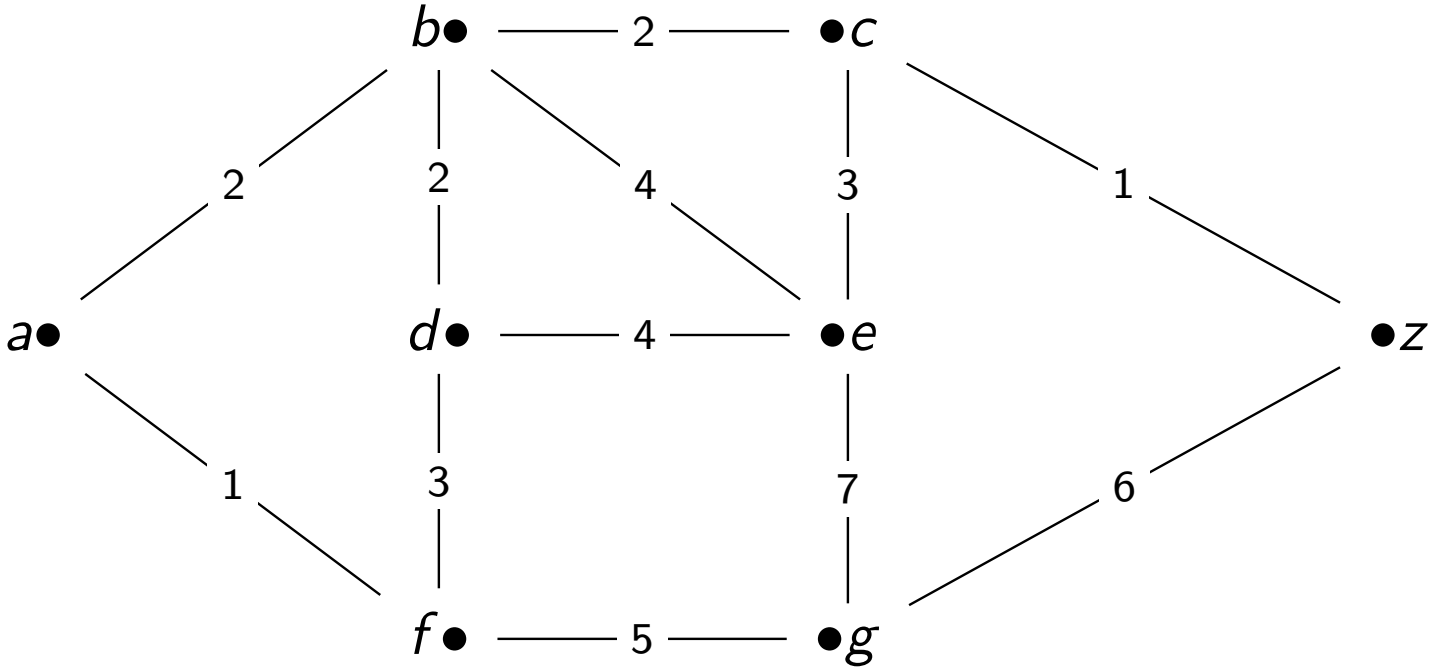
1. Primero, notemos que la ruta más corta del vértice a al vértice a, es la ruta de cero aristas, que tiene peso 0. Así, inicializamos $L(a) = 0$.
2. No sabemos aún el valor de una ruta más corta de a a los otros vértices, entonces para cada vértice $v \neq a$, inicializamos $L(v) = \infty$.
3. Inicializamos 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.

El algoritmo de Dijkstra

El algoritmo de Dijkstra para la ruta mas corta del vértice a al vértice z implica asignar etiquetas a los vértices. Sea $L(v)$ la etiqueta del vértice v. En cualquier punto, algunos vértices tienen etiquetas temporales y el resto son permanentes. Sea T el conjunto de vértices que tienen etiquetas temporales. Al ilustrar el algoritmo, normalmente remarcamos con un circulo los vértices que tienen etiquetas temporales. Si $L(v)$ es la etiqueta permanente del vértice v, entonces $L(v)$ es la longitud de una ruta más corta de a a v. Al inicio, todos los vértices tienen etiquetas temporales. Cada iteración del algoritmo cambia el estado de una etiqueta de temporal a permanente; entonces el algoritmo puede terminar cuando z recibe una etiqueta permanente.

El algoritmo Dijkstra - 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

Algoritmo de Dijkstra - ejemplo

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	∞	∞	∞	∞	∞	∞	∞

Algoritmo de Dijkstra - ejemplo

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	/	???	∞	∞	∞	???	∞	∞

Algoritmo de Dijkstra - ejemplo

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	/	$\min\{\infty, 0 + 2\}$	∞	∞	∞	???	∞	∞

Algoritmo de Dijkstra - ejemplo

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	∞	∞	∞	???	∞	∞

Algoritmo de Dijkstra - ejemplo

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	∞	∞	∞	$\min\{ \infty, 0+1 \}$	∞	∞

Algoritmo de Dijkstra - ejemplo

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	∞	∞

Algoritmo de Dijkstra - ejemplo

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	∞	???	∞	/	???	∞

Algoritmo de Dijkstra - ejemplo

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	∞	$\min\{ \infty, 1 + 3 \}$	∞	/	???	∞

Algoritmo de Dijkstra - ejemplo

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	∞	/	???	∞

Algoritmo de Dijkstra - ejemplo

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	∞	/	$\min\{ 1 + 5 \}$	∞

Algoritmo de Dijkstra - ejemplo

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	∞

El algoritmo Dijkstra - ejemplo

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.

El algoritmo Dijkstra - modificado

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.