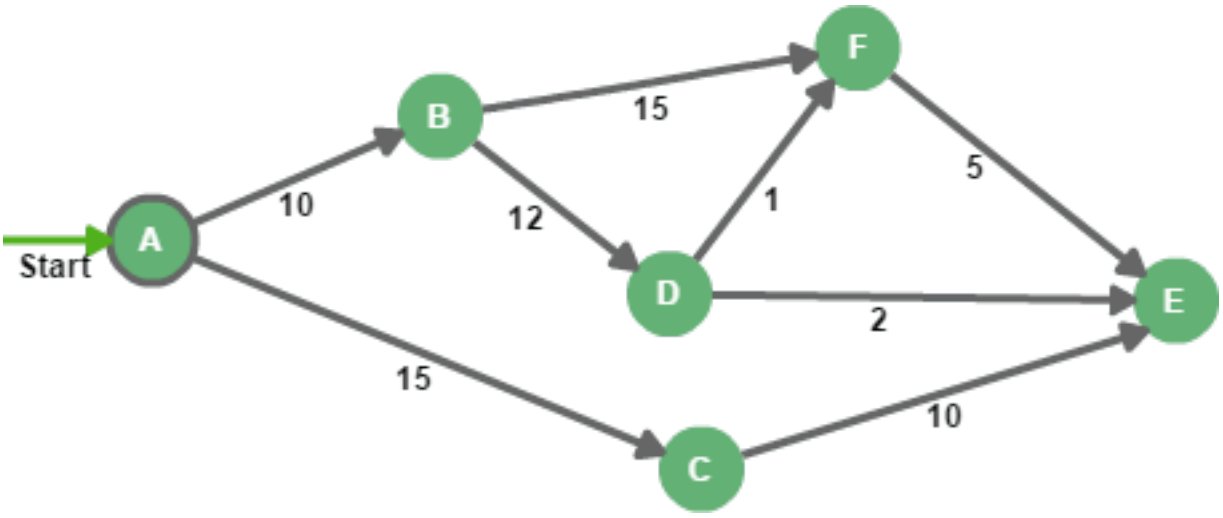
Extraordinario de Inteligencia Artificial - Grupo: ED61
Mauricio Garduño Gonzalez - No. Cuenta: 311234543

Algoritmo de Dijkstra

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino mas corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Su nombre alude a Edsger Dijkstra, científico de la computación de los Países Bajos que lo describió por primera vez en 1959.



Algoritmo

Teniendo un grafo dirigido ponderado de N nodos no aislados, sea x el nodo inicial. Un vector D de tamaño N guardará al final del algoritmo las distancias desde x hasta el resto de los nodos.

1) Inicializar todas las distancias en D con un valor infinito relativo, ya que son desconocidas al principio, exceptuando la de x, que se debe colocar en 0, debido a que la distancia de x a x sería 0.

2) Sea a = x (Se toma a como nodo actual.)

3)Se recorren todos los nodos adyacentes de a, excepto los nodos marcados. Se les llamará nodos no marcados vi.

4) Para el nodo actual, se calcula la distancia tentativa desde dicho nodo hasta sus vecinos con la siguiente fórmula: dt(vi) = Da + d(a,vi). Es decir, la distancia tentativa del nodo ‘vi’ es la distancia que actualmente tiene el nodo en el vector D más la distancia desde dicho nodo ‘a’ (el actual) hasta el nodo vi. Si la distancia tentativa es menor que la distancia almacenada en el vector, entonces se actualiza el vector con esta distancia tentativa. Es decir, si dt(vi) < Dvi → Dvi = dt(vi)

5) Se marca como completo el nodo a.

6) Se toma como próximo nodo actual el de menor valor en D (puede hacerse almacenando los valores en una cola de prioridad) y se regresa al paso 3, mientras existan nodos no marcados.

7) Una vez terminado al algoritmo, D estará completamente lleno.

Pseudocódigo

```
DIJKSTRA (Grafo G, nodo_fuente s)
    para u ∈ V[G] hacer
        distancia[u] = INFINITO
        padre[u] = NULL
        visto[u] = false
    distancia[s] = 0
    adicionar (cola, (s, distancia[s]))
    mientras que cola no es vacía hacer
        u = extraer_mínimo(cola)
        visto[u] = true
        para todos v ∈ adyacencia[u] && visto[v]==False hacer
            si distancia[v] > distancia[u] + peso (u, v) hacer
                distancia[v] = distancia[u] + peso (u, v)
                padre[v] = u
                adicionar(cola,(v, distancia[v]))
```

Aplicaciones

Sistemas de navegación GPS (Tom-Tom).
Los sistemas de navegación están diseñados para proporcionar al usuario el trayecto óptimo de la ruta que pretende realizar, apoyándose del algoritmo de Dijkstra a la hora de elegir el camino mínimo. Son comúnmente utilizados en el envío de correspondencia tanto vía aérea, terrestre y marítimo. Se establecen una series de lugares (nodos) interconectados entre sí (arista) y se calcula cuál sería el camino más económico para realizar ese servicio.

Por ejemplo, una empresa de paquetería en Japón tiene su base en Komatsujima y tiene que enviar cajas a Sawatawi y a Waki. ¿Qué ruta deben escoger?

También se basa en este algoritmo el motor de búsqueda de Google Maps, eligiendo el camino más corto a recorrer entre dos puntos distintos.

Reconocimiento de lenguaje hablado.
Aplicaciones como el reconocimiento de voz en los smartphones, o a la hora de transcribir al ordenador el lenguaje hablado por ciegos presentan problemas a la hora de distinguir palabras que suenan de manera similar. Como solución, se puede construir un grafo cuyos vértices sean las posibles palabras y cuyos arcos unan palabras que puedan ir colocadas con coherencia. Si tomamos como peso del arco la probabilidad de que estén colocadas de esa manera, el camino más corto será la mejor interpretación de la frase.

Reconstrucción de bordes de imágenes.
El algoritmo de Dijkstra se utiliza también en técnicas de procesamiento de imágenes, concretamente en la reconstrucción de bordes cuando se presentan discontinuidades por exceso de ruido en el archivo procesado. Se crea un grafo en el que cada nodo corresponde a un elemento de borde, unidos por aristas en caso de que se pueda formar un borde entre ellos. A cada arista se le asigna un valor (peso) que se calcula con la siguiente fórmula: $c(p, q) = H - [f(p) - f(q)]$

donde H es el valor máximo de la intensidad luminosa de la imagen y f(·) es el valor de intensidad de un píxel, por lo que f(p) - f (q) es la intensidad de ese elemento del borde. El proceso termina mediante el cálculo del camino de mínimo coste. Este camino determina el borde final.

Transmisión de datos por la red.
Toda la información de Internet se transmite a través de paquetes que se intercambian entre servidor y router. La dirección que tome varía según el tráfico de la red, el cuál se guía a través del algoritmo de Dijkstra que se encarga de buscar un camino entre todos los posibles. A este proceso se le conoce como encaminamiento.

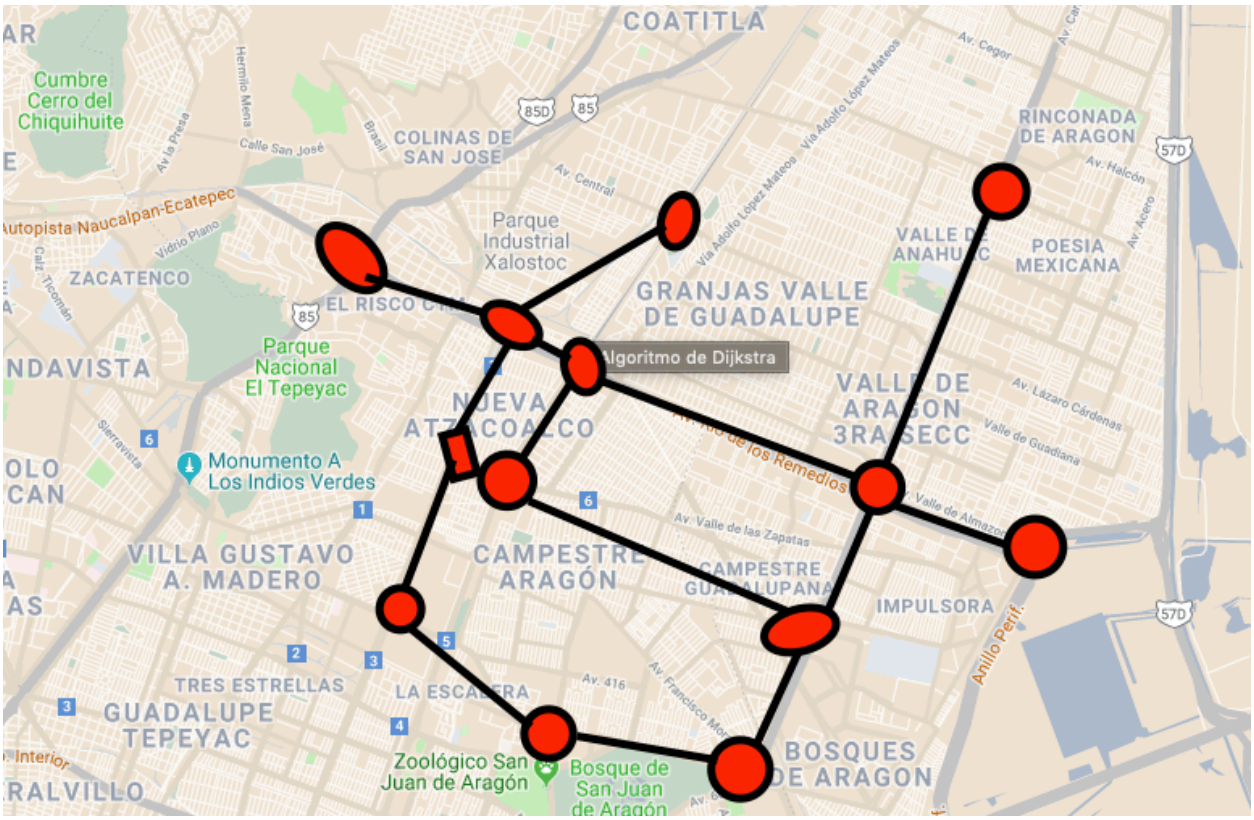
Empresas que aplican Algoritmo de Dijkstra Como Google

Google utiliza algoritmos para determinar nuestras mejores rutas. Para que estos algoritmos funcionen correctamente, necesitan la forma correcta de entrada. Consideremos primero qué concepto tiene más sentido aplicar al problema de encontrar el camino más corto. Una red de carreteras se muestra mejor en una vista de arriba hacia abajo, como en una imagen de satélite. En la siguiente imagen, vemos cómo se ve los alrededores de Fes Aragon.

Desafortunadamente, el algoritmo de búsqueda de rutas utilizado por Google Maps no funciona con imágenes. Funciona con una estructura de datos diferente. Siempre podemos considerar la red de carreteras como un gráfico dirigido.

Consideremos cómo se vería la Figura 1 si la convirtiéramos en una gráfica, cf. Figura 2. Cada intersección y cada ubicación de interés pueden considerarse como un vértice del gráfico y las carreteras que conectan los vértices son los bordes.

Las líneas en la Figura 2 representan los bordes del gráfico y los círculos son los nodos (para un curso básico básico en teoría de gráficos, lea la Paradoja de la Amistad).
Denote el conjunto de bordes por \ (E \) y el conjunto de vértices por \ (V. \) Para cada borde \ (e \ en E \) denotamos su peso de borde por \ (c_e. \)
En el caso de una red de carreteras, los pesos de borde representan el tiempo que tarda en ir desde un vértice \ (u \) a su vecino \ (v \) a través del borde \ (\ {u, v \}, \) que representa una carretera que va de \ (u \) a \ (v. \)
.La tarea de encontrar el camino más corto desde el punto A hasta el punto B se puede reducir a encontrar el camino más corto en un gráfico ponderado. Hay muchos algoritmos diferentes que pueden hacer esto, pero solo queremos discutir el que introdujo Dijkstra. (Lo más probable es que Google Maps utilice la búsqueda \ (A ^ * \)).



Problemas

En la práctica, por supuesto, hay muchas más complicaciones involucradas en el proceso de encontrar el camino más corto, como:
No es trivial crear una gráfica a partir del conocimiento de dónde están los caminos y las ubicaciones.
Los atascos de tráfico afectan el tiempo de viaje y no es posible predecirlos todos por adelantado.
Algunas carreteras pueden estar cerradas durante ciertas horas durante el día, lo que hace que el gráfico dependa del tiempo.
Pero, como todos sabemos, el proceso de búsqueda de información está mejorando cada año y esperamos poder ver nuevos descubrimientos en esta área de investigación.

Código en JAVA

```
1
```

```
/**
 * Realizar el algoritmo de Dijkstra sobre el grafo
```

```
2
        * @param origen nodo inicial
3
        * @param destino nodo destino
4
        * @return camino ArrayList con el camino a seguir.
5
        */
6
        public ArrayList<Integer> dijkstra(int origen, int destino) {
7
                ArrayList<Integer> camino= new ArrayList<Integer>();
8
                int distancia=Grafo.INFINITO;
9
                int nodo=origen;
10
                boolean fin=true;
11
                camino.add(nodo);
12
                while(fin) {
13
                        if(this.floydC[nodo][destino]<distancia) {
14
                          /*El metodo siguiente(nodo, destino), nos devuelve
15
                          el siguiente nodo a visitar*/
16
                                nodo=this.siguiente(nodo, destino);
17
                                camino.add(nodo);
18
                        }
19
```

```
20                      if(nodo==destino) {
21                          fin=false
22                      }
23                  }
24
25                  return camino;
26          }
```

Proyecto

Consiste en usar la API de Google Maps para saber la ruta mas corta a una lugar de la CDMX y el precio aproximado de lo que nos cobraría un TAXI., el objetivo es crear un sitio web informativo del algoritmo de Dijkstra.

Repositorio y Documentación:
https://github.com/maubuntos/Dijkstra

Sitio web en producción:
https://maubuntos.github.io/Dijkstra/