Algoritmo de Dijkstra

Luis Enrique López Nerio Universidad Autónoma de Nuevo León
luiselopeznerio@gmail.com

19 de Octubre del 2017

Abstract

En este reporte se ahondara en el tema de teoría de grafos introducido en el reporte 3, se tomaran las mismas bases y los algoritmos ya creados para crear grafos y a partir de eso se elaborara un algoritmo utilizado para encontrar los caminos más cortos de un nodo dado a cada uno de los otros nodos del grafo.

I. La pregunta del millon de dolares

Como se mencionó antes un grafo se puede interpretar matemáticamente como como dos conjuntos, uno de vértices y otro de aristas, el de vértices contiene todos los puntos del grafo y el de aristas nos dice que vértices de nuestro grafo están conectados. Este objeto matemático es útil ya que muchos problemas de la vida real pueden ser resueltos si se abstrae su idea principal como un grafo, este objeto se puede hacer más complejo si consideramos que las aristas tienen un peso, esto es a lo que llamamos un grafo ponderado

Un ejemplo concreto seria el Grafo que representa a México, sus vértices podríamos considerarlos cada una de las ciudades más importantes del país, sus aristas podrían ser las carreteras que conecten a cada una de esas ciudades; es conveniente pensar que el peso de las aristas sea la longitud entre ciudad y ciudad.

Una vez hecho esto nos podemos hacer muchas preguntas, cual es la distancia más larga entre cualquiera de los nodos del grafo, cuales son nodos que minimizan la distancia más larga a los otros nodos, etc. La pregunta que responderemos en este reporte será:

Dfddd Dado un vértice del grafo, Cual es la distancia más corta a cada uno de los otros vértices del grafo DFD, en nuestro ejemplo del grafo de México podríamos interpretarla como fdfdSi estoy en una ciudad, cual es la distancia más corta a cada uno de las otras ciudades

II. Algoritmo de Dijkstra

El algoritmo de Dijkstra es un algortimos que nos ayudara a reponder la pregunta, dado un verticeFDF Cual es la distancia mas corta a cada uno de los vertices del grafoDFSDF. Este algoritmo fue desarrollado alredodor de los 1950's por el cientifico de computación Edsger W. Dijkstra.

Originalmente el algoritmo se utilizaba para encontrar el camino más corto entre dos nodos, versiones posteriores fueron desarrolladas para encontrar el camino más corto entre un vértice o nodo específico a cada uno de los otros nodos del grafo.

La idea principal del algoritmo es la siguiente:

1 Se toma un nodo inicial
2 Asignemos al nodo inicial consigo mismo la distancia de cero e infinito a la distancia con cualquier otro nodo.
- Marqueos el nodo inicial como visitado y creemos un conjunto de nodos no visitados con todos los otros vértices del grafo

3 Recorremos todos los vértices vecinos del vértice actual y calculemos su distancia con el nodo actual, si la distancia tentativa más la distancia al nodo actual es menor que la distancia almacenada entonces guardamos la distancia tentativa como la distancia adecuada.
4 Marcamos el nodo como visitado
5 Tomamos como siguiente nodo actual al nodo o vértice con la distancia menor que no este visitado.

A continuación se presenta el codigo en python para el algoritmo de Dijkstra

```
def dijkstra(self, actual):
    distancia = dict()
    previo = dict()
    vertices = self.V.copy()
    for v in vertices:
        distancia[v] = math.inf
        previo[v]="indefinido"

    distancia[actual]= 0

    while len(vertices)!= 0:
        minimo = math.inf
        for e in vertices:
            aux = distancia[e]
            if aux < minimo:
                u = e
                minimo=distancia[e]
        vertices.remove(u)

        for v in self.vecinos[u]:
            alternativa = distancia[u]+self.E[(u,v)]
            if alternativa < distancia[v]:
                distancia[v]=alternativa
                previo[v]=u
    return (distancia,previo)
```

Codigo 1. Algoritmo de Dijkstra

Como podemos ver este algoritmo regresa como resultado dos listas, una es la lista de distancias más cortas al vertice dado y otra del vertice previo, esto es para cada vertice la columna te dice desde que lugar viene el camino para llegar a ese vertice, con estas dos listas podemos encontrar los caminos mas cortos desde el vertice dado a cada vertice.

III. Experimento

Nuestra siguiente tarea sera probrar nuestro algoritmo, para esto crearemos 5 diferentes grafos y aplicaremos el algoritmo sobre ellos, los grafos que crearemos tienen que tener las siguientes caracteristicas:

- 5 vertices y 10 aristas
- 10 vertices y 20 aristas
- 15 vertices y 30 aristas

- 20 vertices y 40 aristas
- 25 vertices y 50 aristas

Sin embargo a cada grafo le reduci 10 aristas porque de esta manera los grafos quedan demasiado densos, si esto pasa al tomar un vertice, este sera vecino con la mayoria de cada uno de los vertices, entonces reduci 10 aristas a cada grago para que esto no pasara.

Para realizar esta tarea escribi un codigo que ayuda a hacer esta tarea un poco mas rapido utilizando la libreria para generar números aleatorios, a continuación se presenta el codigo:

```
import random

def g_al(V, E):
    if V > 26:
        return 0
    al = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
     'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
    vertices = al[0:V]
    g=grafo()
    for i in range(E):
        while True:
            v1 = vertices[random.randint(0,V-1)]
            v2 = vertices[random.randint(0,V-1)]
            if (v1 != v2 and (v1,v2) not in g.E) :
                break
        g.conecta(v1,v2,peso = random.randint(1,100))

    return g

g1 = g_al(5, 5)
g2 = g_al(10, 10)
g3 = g_al(15, 20)
g4 = g_al(20, 30)
g5 = g_al(25,40)

res1= g1.dijkstra('a')
res2=g2.dijkstra('a')
res3=g3.dijkstra('a')
res4=g4.dijkstra('a')
res5=g5.dijkstra('a')
```

Codigo 2. Codigo para generar grafos

A continuación se presentan los resultados

i. 5 vertices 5 aristas

V	Distancia	Vertice previo
a	0	indefinido
b	134	c
c	35	a
d	84	a
e	67	c

ii. 10 vertices 15 aristas

V	Distancia	Vertice previo
a	0	indefinido
b	43	h
c	96	i
d	10	g
e	47	d
f	89	h
g	2	a
h	17	a
i	53	d
i	14	g

iii. 15 vertices 20 aristas

V	Distancia	Vertice previo
a	0	indefinido
b	277	f
c	49	a
d	219	k
e	230	m
f	238	i
g	244	m
h	184	o
i	211	h
i	174	o
k	217	i
l	142	c
m	191	h
n	218	o
o	171	l

iv. 20 vertices 25 aristas

V	Distancia		Vertice previo
a	0		indefinido
c	3		a
d	34		a
e	149	s	
f	200		l
g	97		q
h	86		a
i	157		l
i	208		m
k	77		c
l	115		r
m	148		s
n	149		r
o	123		r
p	141		h
q	73		c
r	84		t
s	95		a
t	83		q

v. 25 vertices 30 aristas

V	Distancia	Vertice previo
a	0	indefinido
b	158	v
c	167	s
d	206	e
e	157	u
g	118	u
i	34	a
i	292	r
k	150	s
l	194	e
m	218	r
n	221	b
o	227	v
p	240	c
q	48	i
r	211	l
s	97	x
u	100	q
v	132	g
w	221	d
x	54	q
y	143	x

IV. Conclusión

Mi conclusión es que el algoritmo de Dijkstra incrementa exponencialmente nuestro potencial para abordar problemas relacionados con teoría de grafos, solo basta encontrar un problema que se puede interpretar de esta manera y listo. Valdria la pena tratar de optimizar el algoritmo con algunas mejoras, mas esto queda como trabajo futuro.