Reporte de algoritmo de Dijkstra

Yadira Yaneth Marroquín de la Peña

Yaz_marroquin@outlook.com

20 de octubre de 2017

En el siguiente reporte tiene como objetivo mostrar y analizar los algoritmos de Dijkstra donde tiene como función de conseguir la distancia o rutas más cercas. En documento mostraremos el algoritmo y las tablas de cada uno de los grafos.

1. Dijkstra

1.1 Que es el Dijkstra

Principalmente tiene como funcionamiento el algoritmo determina la ruta más corta desde un nodo origen hacia los demás nodos para ello es requerido como entrada un grafo cuyas aristas posean pesos. Algunas consideraciones:

- Si los pesos de mis aristas son de valor 1, entonces bastará con usar el algoritmo de BFS.
- Si los pesos de mis aristas son negativos no puedo usar el algoritmo de dijsktra, para pesos negativos tenemos otro algoritmo llamado Algoritmo de Bellmand-Ford.

1.2 Descripción

Este programa tiene como objetivo llegar a un destino por un camino más rápido, esto que nos sirve que a veces queremos ir una cierta parte pero no sabe qué destino o ruta tomar para llegar más rápido por ciertas circunstancias. Si lo observamos el programa de Grafo donde los aristas con pesos para poder saber con mayor facilidad la ruta más corta.

Para el Grafo es necesario iniciar con un nodo que tenga una distancia 0, para poder donde con un nodo inicial tomaremos una ruta más corta de ahí observar que ruta siguiente nos conviene tomar.

1.3 Funcionamiento del Algoritmo

El algoritmo forma parte de un vértice en el origen, a partir de ese vértices evaluaremos sus adyacentes, como dijkstra usa una técnica el principio para que un camino sea óptimo, todos los caminos que contiene también deben ser óptimos en sus vértices adyacentes, buscamos el que esté más cerca de nuestro punto origen, lo tomamos como punto intermedio y vemos si podemos llegar más rápido a través de este vértice a los demás.

Después escogemos al siguiente más cercano y repetimos el proceso. Esto lo hacemos hasta que el vértice no utilizado más cercano sea nuestro destino. Al proceso de actualizar las distancias tomando como punto intermedio al nuevo vértice se le conoce como relajación.

1.4 Características del Dijkstra

Trabaja por etapas, y toma en cada etapa la mejor solución sin considerar consecuencias futuras. El óptimo encontrado en una etapa puede modificarse posteriormente si surge una solución mejor.

Pasos del algoritmo:

Sea V un conjunto de vértices de un grafo.

Sea C una matriz de costos de las aristas del grafo, donde en C[u,v] se almacena el costo de la arista entre u y v.

Sea S un conjunto que contendrá los vértices para los cuales ya se tiene determinado el camino mínimo.

Sea D un arreglo unidimensional tal que D[v] es el costo del camino mínimo del vértice origen al vértice v.

1.5 Código de Pseudocódigo

```
método Dijkstra(Grafo,origen):
    creamos una cola de prioridad Q
    agregamos origen a la cola de prioridad Q
    mientras Q no este vacío:
        sacamos un elemento de la cola Q llamado u
        si u ya fue visitado continuo sacando elementos de Q
        marcamos como visitado u
        para cada vértice v adyacente a u en el Grafo:
            sea w el peso entre vértices ( u , v )
            si v no ah sido visitado:
               Relajacion( u , v , w )
```

```
método Relajacion( actual , adyacente , peso ):
    si distancia[ actual ] + peso < distancia[ adyacente ]
       distancia[ adyacente ] = distancia[ actual ] + peso
       agregamos adyacente a la cola de prioridad Q
```

1.6 Código del Grafo

```
def shortest(self, v):
    q = [(0, v, ())]
    dist = dict()
    visited = set()
    while len(q) > 0:
        (l, u, p) = heappop(q)
        if u not in visited:
            visited.add(u)
            dist[u] = (l,u,list(flatten(p))[::-1] + [u])
        p = (u, p)
        for n in self.vecinos[u]:
            if n not in visited:
                el = self.E[(u,n)]
                heappush(q, (l + el, n, p))
    return dist
```

2. Funcionamiento de grafos de Dijkstra

A continuación mostraremos los 5 grafos para probar el funcionamiento de cada grafo. Observaremos cuantos nodos y aristas se realizara en cada uno de las tablas.

a) Grafo 1

```
# primero 5 nodos con 10 aristas
print("primer grafo")
g=Grafo()
g.conecta('a','b', 2)
g.conecta('a','c', 3)
g.conecta('a','d', 10)
g.conecta('a','e', 6)
g.conecta('b','c', 5)
g.conecta('b','d', 9)
g.conecta('b','e', 14)
g.conecta('c','d', 7)
g.conecta('d','e', 1)
g.conecta('e','c', 16)
print(g.vecinos['a'])
print(g.shortest('a'))
```

Resultados de programa

primer grafo

{'b', 'c', 'd', 'e'}

{'a': (0, 'a', ['a']), 'b': (2, 'b', ['a', 'b']), 'c': (3, 'c', ['a', 'c']), 'e': (6, 'e', ['a', 'e']), 'd': (7, 'd', ['a', 'e', 'd'])}

Nodo inicial	Nodo final	Pesos	Distancias
A	A	0	[´a´]
A	B	2	['a', 'b']
A	C	3	['a', 'c']
A	E	6	['a', 'e']
A	D	7	['a', 'e', 'd']

b) Grafo 2

```
# primero 10 nodos con 20 aristas
print("segundo grafo")
g1=Grafo()
g1.conecta('a','b', 5)
g1.conecta('a','c', 12)
g1.conecta('a','d', 8)
g1.conecta('a','e', 2)
g1.conecta('a','f', 15)
g1.conecta('a','g', 6)
g1.conecta('a','h', 4)
g1.conecta('a','i', 7)
g1.conecta('a','j', 9)
g1.conecta('b','c', 16)
g1.conecta('b','d', 1)
g1.conecta('b','e', 14)
g1.conecta('b','f', 10)
g1.conecta('b','g', 7)
g1.conecta('b','h', 13)
g1.conecta('b','i', 9)
g1.conecta('b','j', 17)
g1.conecta('c','d', 7)
g1.conecta('c','e', 11)
g1.conecta('c','f', 5)
print(g1.vecinos['b'])
print(g1.shortest('b'))
```

Tabla-

Nodo inicial	Nodo final Pesos		Distancias
A A	B D	0 1	['b'] ['b', 'd']
A	A	5	['b', 'a']
A	E	7	['b', 'a', 'e']
A	G	7	['b', 'g']
A	C	8	['b', 'd', 'c']
A	H	9	['b', 'a', 'h']
A	I	10	['b', 'i']
A	F	10	['b', 'f']
A	**J**		['b', 'a', 'j']

Grafica 3

```
# primero 15 nodos con 30 aristas
print("tercer grafo")
g2=Grafo()
g2.conecta('a','b', 2)
g2.conecta('a','c', 3)
g2.conecta('a','d', 10)
g2.conecta('a','e', 6)
g2.conecta('a','f', 5)
g2.conecta('a','g', 9)
g2.conecta('a','h', 14)
g2.conecta('a','i', 7)
g2.conecta('a','j', 1)
g2.conecta('a','k', 16)
g2.conecta('a','l', 2)
g2.conecta('a','m', 3)
g2.conecta('a','n', 10)
g2.conecta('a','o', 6)
g2.conecta('b','c', 5)
g2.conecta('b','d', 9)
g2.conecta('b','e', 14)
g2.conecta('b','f', 7)
g2.conecta('b','g', 1)
g2.conecta('b','h', 16)
g2.conecta('b','i', 2)
g2.conecta('b','j', 3)
g2.conecta('b','k', 10)
g2.conecta('b','l', 6)
g2.conecta('b','m', 5)
g2.conecta('b','n', 9)
g2.conecta('b','o', 14)
g2.conecta('c','d', 7)
g2.conecta('c','e', 1)
g2.conecta('c','f', 16)
print(g2.vecinos['a'])
print(g2.shortest('a'))
```

Tabla

Nodo inicial	Nodo final	Pesos	Distancias
A	J	1	['a', 'j']
A	B	2	['a', 'b']
A	L	2	['a', 'l']
A	C	3	['a', 'c']
A	G	3	['a', 'b', 'g']
A	M	3	['a', 'm']
A	E	4	['a', 'c', 'e']
A	I	4	['a', 'b', 'i']
A	F	5	['a', 'f']
A	O	6	['a', 'o']
A	D	10	['a', 'd']
A	N	10	['a', 'n']
A	K	12	['a', 'b', 'k']

A	H	14	['a', 'h']

C) Grafo 4

```
# primero 20 nodos con 40 aristas
print("cuarto grafo")
g3=Grafo()
g3.conecta('a','b', 2)
g3.conecta('a','c', 4)
g3.conecta('a','d', 12)
g3.conecta('a','e', 23)
g3.conecta('a','f', 17)
g3.conecta('a','g', 9)
g3.conecta('a','h', 7)
g3.conecta('a','i', 8)
g3.conecta('a','j', 19)
g3.conecta('a','k', 14)
g3.conecta('a','l', 11)
g3.conecta('a','m', 16)
g3.conecta('a','n', 6)
g3.conecta('a','o', 3)
g3.conecta('a','p', 5)
g3.conecta('a','q', 26)
g3.conecta('a','r', 15)
g3.conecta('a','s', 22)
g3.conecta('a','t', 10)
g3.conecta('b','c', 21)
g3.conecta('b','d', 29)
g3.conecta('b','e', 1)
g3.conecta('b','f', 18)
g3.conecta('b','g', 25)
g3.conecta('b','h', 31)
g3.conecta('b','i', 9)
g3.conecta('b','j', 34)
g3.conecta('b','k', 36)
g3.conecta('b','l', 40)
g3.conecta('b','m', 46)
g3.conecta('b','n', 20)
g3.conecta('b','o', 33)
g3.conecta('b','p', 10)
g3.conecta('b','q', 6)
g3.conecta('b','r', 25)
g3.conecta('b','s', 9)
g3.conecta('b','t', 24)
```

```
g3.conecta('b','s', 9)
g3.conecta('b','t', 24)
g3.conecta('c','d', 17)
g3.conecta('c','e', 13)
g3.conecta('c','f', 18)
print(g3.vecinos['a'])
print(g3.shortest('a'))
```

Tabla

Nodo inicial	Nodo final	Pesos	Distancias
A	B	2	['a', 'b']
A	E	3	['a', 'b', 'e']
A	O	3	['a', 'o']
A	C	4	['a', 'c']
A	P	5	['a', 'p']
A	N	6	['a', 'n']
A	H	7	['a', 'h']
A	I	8	['a', 'i']
A	Q	8	['a', 'b', 'q']
A	G	9	['a', 'g']
A	T	10	['a', 't']
A	L	11	['a', 'l']
A	S	11	['a', 'b', 's']
A	D	12	['a', 'd']
A	K	14	['a', 'k']
A	R	15	['a', 'k']
A	M	16	['a', 'm']
A	F	17	['a', 'f']
A	J	19	['a', 'j']

Grafo 5

```
# primero 25 nodos con 50 aristas
print("quinto grafo")
g4=Grafo()
g4.conecta('a','b', 2)
g4.conecta('a','c', 4)
g4.conecta('a','d', 10)
g4.conecta('a','e', 13)
g4.conecta('a','f', 17)
g4.conecta('a','g', 11)
g4.conecta('a','h', 27)
g4.conecta('a','i', 8)
g4.conecta('a','j', 9)
g4.conecta('a','k', 24)
g4.conecta('a','l', 31)
g4.conecta('a','m', 26)
g4.conecta('a','n', 5)
g4.conecta('a','o', 21)
g4.conecta('a','p', 34)
g4.conecta('a','q', 16)
g4.conecta('a','r', 35)
g4.conecta('a','s', 32)
g4.conecta('a','t', 28)
g4.conecta('a','v', 17)
g4.conecta('a','w', 3)
g4.conecta('a','x', 7)
g4.conecta('a','y', 8)
g4.conecta('b','c', 23)
g4.conecta('b','d', 29)
g4.conecta('b','e', 33)
g4.conecta('b','f', 18)
g4.conecta('b','g', 35)
g4.conecta('b','h', 36)
g4.conecta('b','i', 42)
g4.conecta('b','j', 28)
g4.conecta('b','k', 13)
g4.conecta('b','l', 32)
g4.conecta('b','m', 11)
g4.conecta('b','n', 34)
g4.conecta('b','o', 21)
g4.conecta('b','p', 4)
g4.conecta('b','q', 9)
```

```
g4.conecta('b','p', 4)
g4.conecta('b','q', 9)
g4.conecta('b','r', 7)
g4.conecta('b','s', 12)
g4.conecta('b','t', 19)
g4.conecta('b','v', 20)
g4.conecta('b','w', 30)
g4.conecta('b','x', 15)
g4.conecta('b','y', 5)
g4.conecta('c','d', 8)
g4.conecta('c','e', 19)
g4.conecta('c','f', 37)
g4.conecta('c','g', 39)
g4.conecta('c','h', 13)
print(g4.vecinos['a'])
print(g4.shortest('a'))
```

Tabla

Nodo inicial	Nodo final	Pesos	Distancias
A	B	2	['a', 'b']
A	W	3	['a', 'w']
A	C	4	['a', 'c']
A	N	5	['a', 'n']
A	P	6	['a', 'b', 'p']
A	X	7	['a', 'x']
A	G	7	'a', 'b', 'y']
A	Q	8	['a', 'i']
A	E	9	['a', 'j']
A	M	10	['a', 'b', 'r']
A	S	11	['a', 'd']
A	K	11	['a', 'd']
A	F	13	['a', 'b', 'q']
A	H	13	['a', 'e']
A	V	14	['a', 'b', 'm']
A	O	15	['a', 'b', 's']
A	T	17	['a', 'b', 'k']
A	L	17	['a', 'f']
A	L	21	['a', 'c', 'h']
A	H	21	['a', 'v']
A	S	31	['a', 'o']
A	O	4	['a', 'b', 't']
A	T	2	['a', 'b']
A	L	21	['a', 'l']

Conclusión

Este algoritmo nos ayuda mucho ya que en la vida cotidiana las personas vivimos a prisa por llegar al lugar que vallamos entonces programa lo que hace es que te da la ruta más corta por decir desde tu punto inicial hacia ruta más corta a donde quieres llegar, además es un extra porque ya habíamos visto grafo entonces lo de grafo mas algoritmo de Dijkstra. Ente proyecto aprendimos a resolver nodo, aristas y distancias.