MATEMATICAS COMPUTACIONALES.

EVALUCION NO.5 “EL ALGORITMO DE DIJKSTRA”.

LESLIE JARESSY MORALES ORTIZ 1661971

ALGORITMO DIJKSTRA

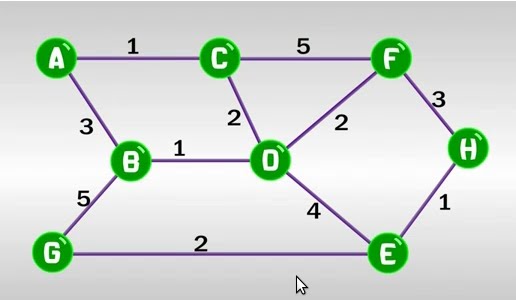
**Descripción**

El algoritmo de dijkstra 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](https://jariasf.wordpress.com/2012/02/27/algoritmo-de-busqueda-breadth-first-search/).
* 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.

**Como trabaja**

Primero marcamos todos los vértices como no utilizados. El algoritmo parte de un vértice origen que será ingresado, a partir de ese vértices evaluaremos sus adyacentes, como dijkstra usa una técnica greedy *– La técnica greedy utiliza el principio de que para que un camino sea óptimo, todos los caminos que contiene también deben ser óptimos-*entre todos los 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 (con las distancias ya actualizadas) 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 (*relaxation*).**

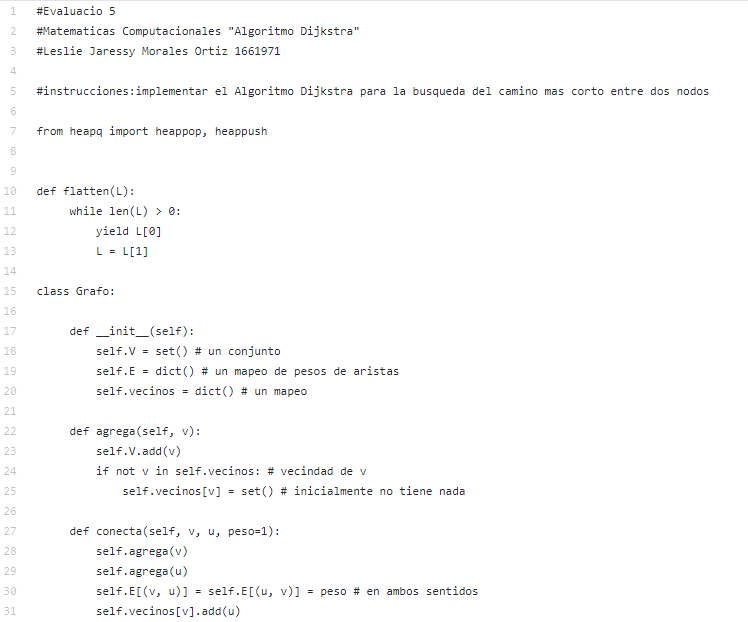


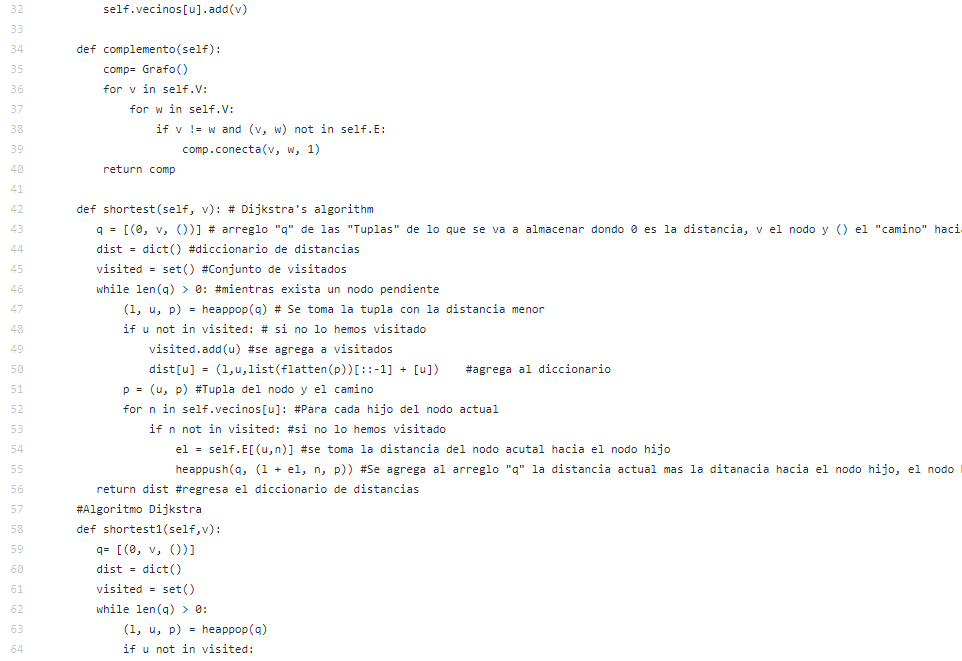
**ANALISIS.**

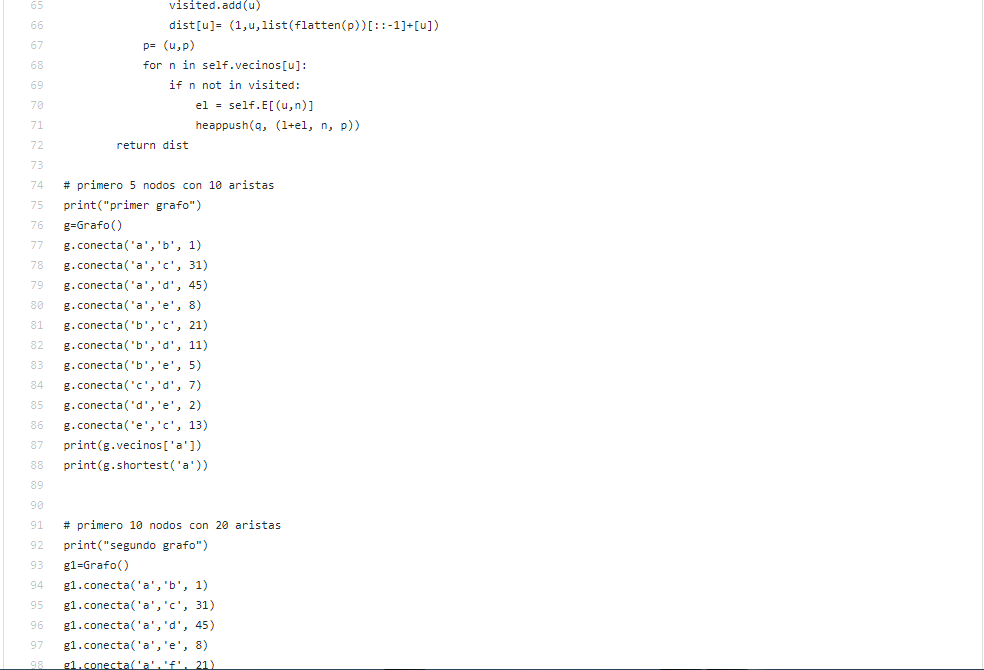
Aplicamos dicho algoritmo para encontrar el camino más corto entre dos nodos el cual tiene como funcionalidad analizar cuál de todos los caminos disponibles es el mejor dado que los vértices se conectan entre ellos a partir de aristas las cuales les hemos dado un valor diferente; todo esto es posible principalmente a la creación de un grafo ya que sin este no sería posible aplicar dicho algoritmo.

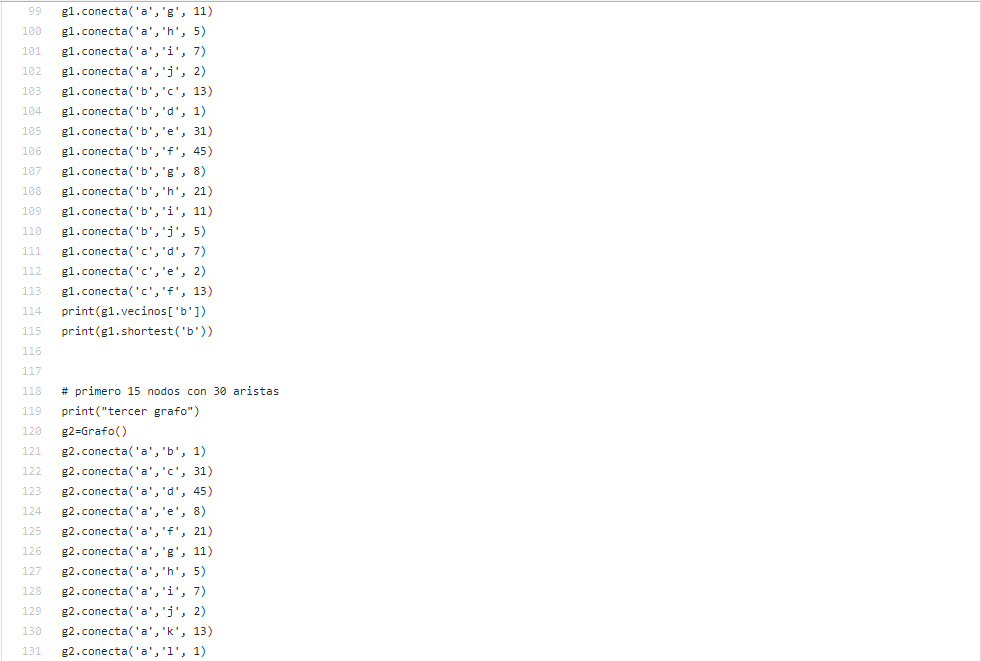
Así mismo después de crear nuestro código, posteriormente conectado los nodos y determinado el camino entre cada uno de ellos se nos mostrara en pantalla los resultados del grafo con el que estuvimos trabajando, sus respectivos caminos y pesos.

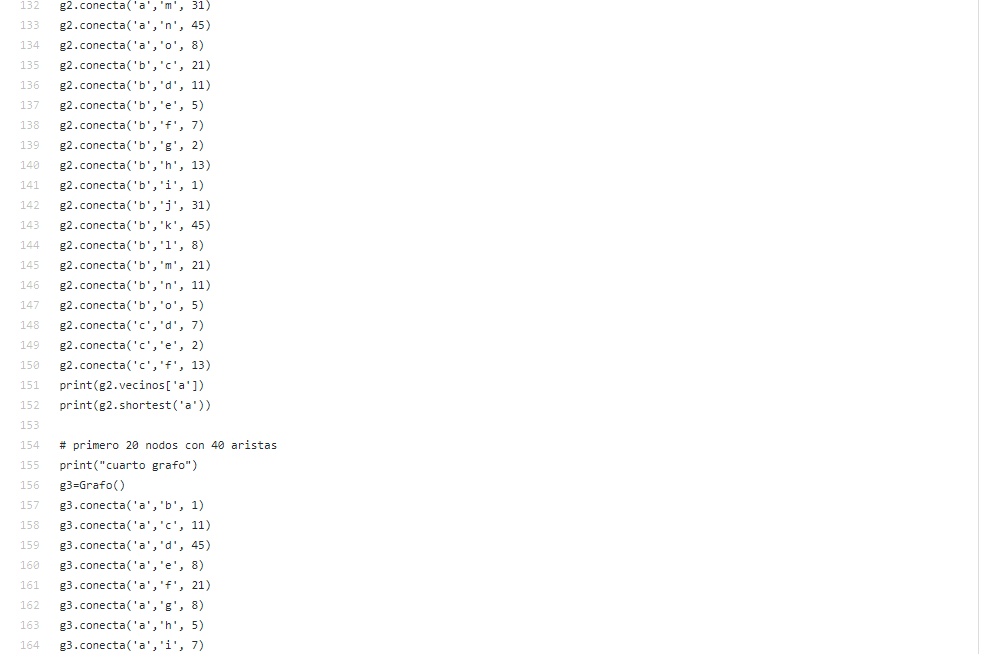
A continuación, mostramos el código con el cual llevamos a cabo nuestras prácticas

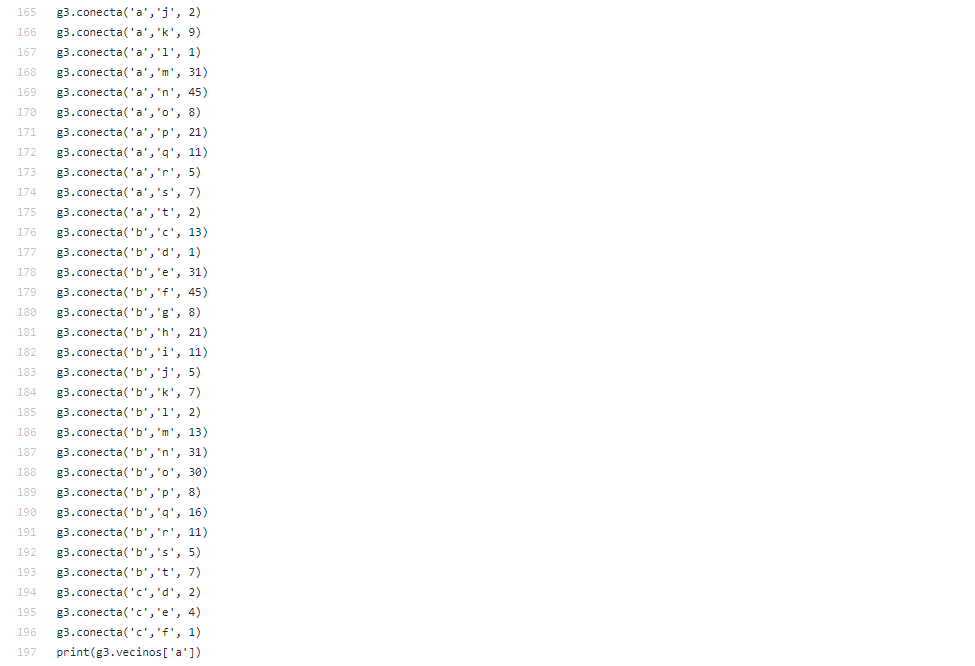


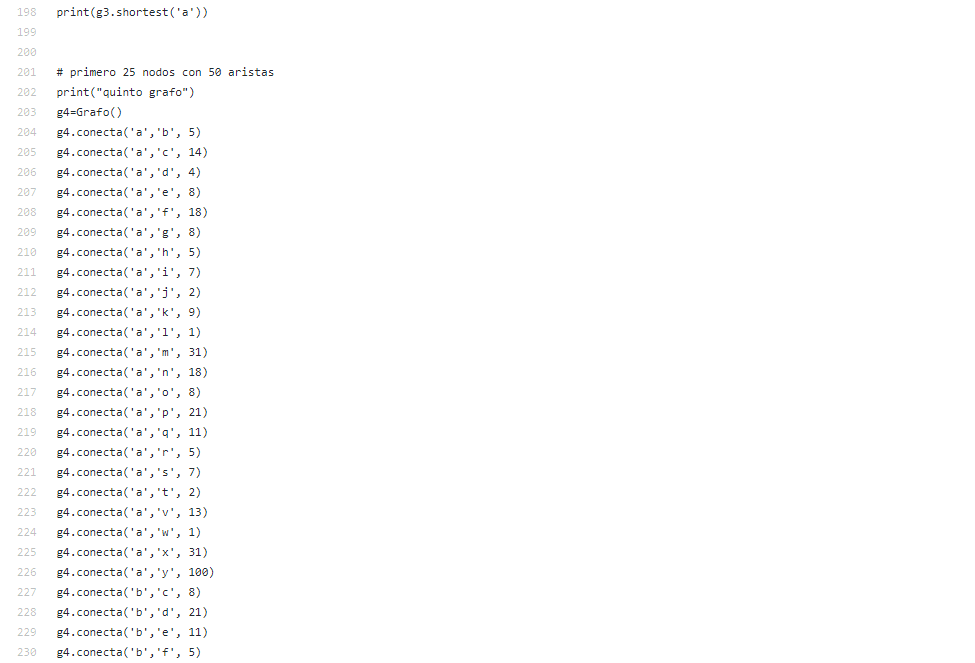














**RESULTADOS.**

* Primer grafo

Se muestra los resultados del primer grafo donde se le dan 5 nodos y 10 aristas.

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

{'a': (0, 'a', ['a']), 'b': (1, 'b', ['a', 'b']), 'e': (6, 'e', ['a', 'b', 'e']), 'd': (8, 'd', ['a', 'b', 'e', 'd']), 'c': (15, 'c', ['a', 'b', 'e', 'd', 'c'])}

|  |  |  |  |
| --- | --- | --- | --- |
| **NODO INC.** | **NODO FIN.** | **CAMINO** | **PESO** |
| a | a | (['a']), | 0 |
| a | b | ( ['a', 'b']) | 1 |
| a | c | (['a', 'b', 'e', 'd', 'c']) | 15 |
| a | d | (['a', 'b', 'e', 'd']) | 8 |
| a | e | (['a', 'b', 'e']) | 6 |

* Segundo grafo

Partimos de 10 nodos y 20 aristas.

{'i', 'e', 'c', 'h', 'g', 'f', 'j', 'a', 'd'}

{'b': (0, 'b', ['b']), 'a': (1, 'a', ['b', 'a']), 'd': (1, 'd', ['b', 'd']), 'j': (3, 'j', ['b', 'a', 'j']), 'h': (6, 'h', ['b', 'a', 'h']), 'c': (8, 'c', ['b', 'd', 'c']), 'g': (8, 'g', ['b', 'g']), 'i': (8, 'i', ['b', 'a', 'i']), 'e': (9, 'e', ['b', 'a', 'e']), 'f': (21, 'f', ['b', 'd', 'c', 'f'])}

|  |  |  |  |
| --- | --- | --- | --- |
| **Columna1** | **Columna2** | **Columna3** | **Columna4** |
| b | b | ( ['b']) | 0 |
| b | a | (['b', 'a']) | 1 |
| b | c | (['b', 'd', 'c']) | 8 |
| b | d | (['b', 'd']) | 1 |
| b | e | (['b', 'a', 'e']) | 9 |
| b | f | (['b', 'd', 'c', 'f']) | 21 |
| b | g | (['b', 'g']) | 8 |
| b | h | (['b', 'a','h']) | 6 |
| b | i | (['b', 'a', 'i']) | 8 |
| b | j | (['b', 'a', 'j']) | 3 |

* Tercer grafo

Para el tercero incrementamos y trabajamos con 15 nodos y 30 aristas.

{'i', 'o', 'e', 'c', 'b', 'g', 'h', 'f', 'k', 'n', 'j', 'm', 'l', 'd'}

{'a': (0, 'a', ['a']), 'b': (1, 'b', ['a', 'b']), 'l': (1, 'l', ['a', 'l']), 'i': (2, 'i', ['a', 'b', 'i']), 'j': (2, 'j', ['a', 'j']), 'g': (3, 'g', ['a', 'b', 'g']), 'h': (5, 'h', ['a', 'h']), 'e': (6, 'e', ['a', 'b', 'e']), 'o': (6, 'o', ['a', 'b', 'o']), 'c': (8, 'c', ['a', 'b', 'e', 'c']), 'f': (8, 'f', ['a', 'b', 'f']), 'd': (12, 'd', ['a', 'b', 'd']), 'n': (12, 'n', ['a', 'b', 'n']), 'k': (13, 'k', ['a', 'k']), 'm': (22, 'm', ['a', 'b', 'm'])}

|  |  |  |  |
| --- | --- | --- | --- |
| **Columna1** | **Columna2** | **Columna3** | **Columna4** |
| a | a | (['a']) | 0 |
| a | b | (['a', 'b']) | 1 |
| a | c | (['a', 'b', 'e', 'c']) | 8 |
| a | d | (['a', 'b', 'd']) | 12 |
| a | e | (['a', 'b', 'e']) | 6 |
| a | f | (['a', 'b', 'f']) | 8 |
| a | g | (['a', 'b', 'g']) | 3 |
| a | h | (['a', 'h']) | 5 |
| a | i | (['a', 'b', 'i']) | 2 |
| a | j | (['a', 'j']) | 2 |
| a | k | (['a', 'k']) | 13 |
| a | l | (['a', 'l']) | 1 |
| a | m | (['a', 'b', 'm']) | 22 |
| a | n | (['a', 'b', 'n']) | 12 |
| a | o | ( ['a', 'b', 'o']) | 6 |

* Cuarto grafo

20 nodos-40 aristas

{'s', 'i', 'r', 'o', 'e', 'p', 'c', 'b', 'g', 'h', 'f', 'k', 'n', 'q', 'j', 'm', 't', 'l', 'd'}

{'a': (0, 'a', ['a']), 'b': (1, 'b', ['a', 'b']), 'l': (1, 'l', ['a', 'l']), 'd': (2, 'd', ['a', 'b', 'd']), 'j': (2, 'j', ['a', 'j']), 't': (2, 't', ['a', 't']), 'c': (4, 'c', ['a', 'b', 'd', 'c']), 'f': (5, 'f', ['a', 'b', 'd', 'c', 'f']), 'h': (5, 'h', ['a', 'h']), 'r': (5, 'r', ['a', 'r']), 's': (6, 's', ['a', 'b', 's']), 'i': (7, 'i', ['a', 'i']), 'e': (8, 'e', ['a', 'e']), 'g': (8, 'g', ['a', 'g']), 'k': (8, 'k', ['a', 'b', 'k']), 'o': (8, 'o', ['a', 'o']), 'p': (9, 'p', ['a', 'b', 'p']), 'q': (11, 'q', ['a', 'q']), 'm': (14, 'm', ['a', 'b', 'm']), 'n': (32, 'n', ['a', 'b', 'n'])}

|  |  |  |  |
| --- | --- | --- | --- |
| **Columna1** | **Columna2** | **Columna3** | **Columna4** |
| a | a | (['a']) | 0 |
| a | b | (['a', 'b']) | 1 |
| a | c | (['a', 'b', 'd', 'c']) | 4 |
| a | d | (['a', 'b', 'd']) | 2 |
| a | e | (['a', 'e']) | 8 |
| a | f | (['a', 'b', 'd', 'c', 'f']) | 5 |
| a | g | (['a', 'g']) | 8 |
| a | h | (['a', 'h']) | 5 |
| a | i | (['a', 'i']) | 7 |
| a | j | (['a', 'j']) | 2 |
| a | k | (['a', 'b', 'k']) | 8 |
| a | l | (['a', 'l']) | 1 |
| a | m | ['a', 'b', 'm']), | 14 |
| a | n | (['a', 'b', 'n']) | 32 |
| a | o | (['a', 'o']) | 8 |
| a | p | (['a', 'b', 'p']) | 9 |
| a | q | (['a', 'q']) | 11 |
| a | r | (['a', 'r']) | 5 |
| a | s | (['a', 'b', 's']) | 6 |
| a | t | ['a', 't']), | 2 |

Quinto grafo

25 nodos-50 aristas

{'i', 'k', 'n', 'j', 't', 'o', 'f', 'y', 'q', 'e', 'p', 'c', 'v', 's', 'l', 'r', 'b', 'h', 'g', 'w', 'm', 'x', 'd'}

{'a': (0, 'a', ['a']), 'l': (1, 'l', ['a', 'l']), 'w': (1, 'w', ['a', 'w']), 'j': (2, 'j', ['a', 'j']), 't': (2, 't', ['a', 't']), 'd': (4, 'd', ['a', 'd']), 'b': (5, 'b', ['a', 'b']), 'h': (5, 'h', ['a', 'h']), 'r': (5, 'r', ['a', 'r']), 's': (6, 's', ['a', 'b', 's']), 'i': (7, 'i', ['a', 'i']), 'm': (7, 'm', ['a', 'b', 'm']), 'q': (7, 'q', ['a', 'b', 'q']), 'u': (7, 'u', ['a', 'b', 'u']), 'e': (8, 'e', ['a', 'e']), 'g': (8, 'g', ['a', 'g']), 'o': (8, 'o', ['a', 'o']), 'c': (9, 'c', ['a', 'g', 'c']), 'k': (9, 'k', ['a', 'k']), 'f': (10, 'f', ['a', 'b', 'f']), 'y': (10, 'y', ['a', 'b', 'y']), 'p': (12, 'p', ['a', 'b', 'p']), 'v': (13, 'v', ['a', 'v']), 'n': (16, 'n', ['a', 'b', 'n']), 'x': (16, 'x', ['a', 'b', 'x'])}

|  |  |  |  |
| --- | --- | --- | --- |
| **Columna1** | **Columna2** | **Columna3** | **Columna4** |
| a | a | (['a']) | 0 |
| a | b | (['a', 'b']) | 5 |
| a | c | (['a', 'g', 'c']) | 9 |
| a | d | (['a', 'd']) | 4 |
| a | e | (['a', 'e']) | 8 |
| a | f | (['a', 'b', 'f']) | 10 |
| a | g | (['a', 'g']) | 8 |
| a | h | (['a', 'h']) | 5 |
| a | i | (['a', 'i']) | 7 |
| a | j | (['a', 'j'])) | 2 |
| a | k | (['a', 'k']) | 9 |
| a | l | (['a', 'l']) | 1 |
| a | m | (['a', 'b', 'm']) | 7 |
| a | n | (['a', 'b', 'n']) | 16 |
| a | o | (['a', 'o']), | 8 |
| a | p | (['a', 'b', 'p']) | 12 |
| a | q | (['a', 'b', 'q']) | 7 |
| a | r | (['a', 'r']) | 5 |
| a | s | (['a', 'b', 's']) | 6 |
| a | t | (['a', 't']) | 2 |
| a | u | (['a', 'b', 'u']) | 7 |
| a | v | (['a', 'v']) | 13 |
| a | w | (['a', 'w']) | 1 |
| a | x | (['a', 'b', 'x']) | 16 |
| a | y | (['a', 'b', 'y']) | 10 |

**CONCLUSION.**

Desde mi punto de vista considero que la implementación de dijkstra es una excelente opción para determinar la distancia desde un lugar a otro ya que eso lo podemos aplicar para las grandes empresas en la búsqueda de rutas más favorables para repartir su mercancía ya que ellos siempre están buscando tener los mejores resultados tanto para sus clientes como para ellos.

El algoritmo de dijkstra es una herramienta que podemos seguir implementando debido a que su interpretación es muy sencilla ;en esta ocasión trabajamos con un grafo donde las aristas no estaban dirigidas , ya que si lo hubiéramos hecho el grado de complejidad habría aumentado , por una parte creo que el enfoque que se nos está dando a conocer más sobre otros campos laborales donde podemos poner en práctica las matemáticas puede dar pie a que cada vez sea más lo estudiantes que deseen estudiar una maestría o un doctorado.

Para finalizar solo quiero resaltar que cada vez puedo entender como están funcionando los programas, es decir me detengo analizar en que otros lugares podrían usarlo para que de esta manera me resulte más intrigante encontrar la estructura para después usarlo en un momento dado.