

Problema del viajero Nuevo Leones

PROYECTO FINAL

Paulina Aguirre García | 1837503 | 26 de noviembre de 2017

Abstract.

¿Cuál es la ruta más corta posible que visita 50 municipios del estado de Nuevo León exactamente una vez y al finalizar regresa a la ciudad origen? Esta es la pregunta que responde la implementación del algoritmo Kruskal en nuestro grafo de cincuenta nodos y mil doscientos veinte y cinco aristas, en el cual también fue implementado el algoritmo de “el vecino más cercano”. Dando así el código del algoritmo en Python.

Introducción.

El problema del viajero consta (o se puede resumir) de dos partes, tener lugares que recorrer y minimizar gastos. Para la primera parte, se introduce un concepto matemático que es objeto de estudio en la Teoría de grafos, un grafo conexo ponderado, y para la segunda parte se hace uso de un algoritmo muy conocido por los informáticos, el algoritmo Kruskal. Es decir, este problema cae en el área de las matemáticas aplicadas.

En este proyecto lo que nos interesa recorrer es el estado de Nuevo León, un viaje por 50 de sus municipios y para esto, no solo usaremos el algoritmo Kruskal si no, otro algoritmo también muy conocido llamado Vecino más cercano.

Para este proyecto es necesario que se conozcan los conceptos de *grafo*, *grafo conexo* y *grafo ponderado*, conceptos matemáticos que tienen una aplicación en ciencias de la computación. Además de esto, saber que es un *algoritmo* y que función tienen el *algoritmo kruskal* y “*vecino más cercano*” en relación con un grafo.

Dato curioso. El primer artículo científico relativo a grafos fue escrito por el matemático suizo Leonhard Euler en 1736.

Definiciones para una mejor comprensión.

¹Def. Grafo. Es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten relaciones binarias entre elementos de un conjunto.

²Def. Grafo conexo. Un grafo se dice conexo si, para cualquier par de vértices u y v en G , existe al menos una trayectoria de u a v .

³Def. Grafo ponderado. Grafo al cual se le ha añadido un peso a las aristas. Generalmente suele ser un número natural.

⁴Def. Algoritmo. Conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos.

Muchos algoritmos son ideados para implementarse en un programa.

⁵Def. Algoritmo Kruskal. Algoritmo para encontrar un árbol de expansión mínima en un grafo ponderado.

⁶Def. Algoritmo Vecino más cercano. También llamado algoritmo voraz (greedy) permite al viajante elegir la ciudad no visitada más cercana como próximo movimiento. Este algoritmo retorna rápidamente una ruta corta.

Esto es,

- Se toma un vértice origen y se agrega a una lista.
- Se revisa cuáles son sus vecinos, elige el de menor peso revisando que no esté en la lista. Se agrega a la lista.
- Así sucesivamente hasta que se tenga en la lista todos los vértices del grafo.
- Se agrega el vértice origen pero al final de la lista.

Para N ciudades aleatoriamente distribuidas en un plano, el algoritmo en promedio retorna un camino de un 25% más largo que el menor camino posible

Implementación en Python de los Algoritmos Kruskal y Vecino más cercano.

Kruskal:

```
107
108 def shortest(self, v): # Dijkstra's algorithm
109     q = [(0, v, ())] # arreglo "q" de las "Tuplas" de lo que se va a almacenar donde 0 es la distancia
110     dist = dict() # diccionario de distancias
111     visited = set() # Conjunto de visitados
112     while len(q) > 0: # mientras exista un nodo pendiente
113         (l, u, p) = heappop(q) # Se toma la tupla con la distancia menor
114         if u not in visited: # si no lo hemos visitado
115             visited.add(u) # se agrega a visitados
116             dist[u] = (l, u, list(flatten(p))[:-1] + [u]) # agrega al diccionario
117             p = (u, p) # Tupla del nodo y el camino
118             for n in self.vecinos[u]: # Para cada hijo del nodo actual
119                 if n not in visited: # si no lo hemos visitado
120                     el = self.E[(u, n)] # se toma la distancia del nodo actual hacia el nodo hijo
121                     heappush(q, (l + el, n, p)) # se agrega al arreglo "q" la distancia actual mas la ditan
122     return dist # regresa el diccionario de distancias
123
124 def kruskal(self):
125     e = deepcopy(self.E)
126     arbol = Grafo()
127     peso = 0
128     comp = dict()
129     t = sorted(e.keys(), key = lambda k: e[k], reverse=True)
130     nuevo = set()
131     while len(t) > 0 and len(nuevo) < len(self.V):
132         #print(len(t))
133         arista = t.pop()
134         w = e[arista]
135         del e[arista]
136         (u, v) = arista
137         c = comp.get(v, {v})
138         if u not in c:
139             #print('u ', u, ' v ', v, ' c ', c)
140             arbol.conecta(u, v, w)
141             peso += w
142             nuevo = c.union(comp.get(u, {u}))
143             for i in nuevo:
144                 comp[i] = nuevo
145     print('MST con peso', peso, ':', nuevo, '\n', arbol.E)
146     return arbol
147
```

Vecino más cercano:

```
147
148 def vecinoMasCercano(self):
149     ni = random.choice(list(self.V))
150     result = [ni]
151     while len(result) < len(self.V):
152         ln = set(self.vecinos[ni])
153         le = dict()
154         res = (ln - set(result))
155         for nv in res:
156             le[nv] = self.E[(ni, nv)]
157         menor = min(le, key = le.get)
158         result.append(menor)
159         ni = menor
160     return result
161
```

Problema del viajero Nuevo Leones.

Listado de los municipios seleccionados:

- | | |
|----------------------|-----------------------------|
| 1.Abasolo | 26.Higueras |
| 2.Agualeguas | 27.Hualahuises |
| 3.Allende | 28.Iturbide |
| 4.Anáhuac | 29.Juárez |
| 5.Apodaca | 30.Lampazos de Naranjo |
| 6.Aramberri | 31.Linares |
| 7.Bustamante | 32.Los Aldamas |
| 8.Cadereyta Jiménez | 33.Los Herreras |
| 9.Cerralvo | 34.Los Ramones |
| 10.China | 35.Marín |
| 11.Ciénega de Flores | 36.Melchor Ocampo |
| 12.Dr. Arroyo | 37.Mier y Noriega |
| 13.Dr. Coss | 39.Montemorelos |
| 14.Dr. González | 40.Monterrey |
| 15.El Carmen | 41.Parás |
| 16.Galeana | 42.Pesquería |
| 17.García | 43.Rayones |
| 18.Gral. Bravo | 44.Sabinas Hidalgo |
| 19.Gral. Escobedo | 45.Salinas Victoria |
| 20.Gral. Terán | 46.San Nicolás de los Garza |
| 21.Gral. Treviño | 47.San Pedro Garza García |
| 22.Gral. Zaragoza | 48.Santa Catarina |
| 23.Gral. Zuazua | 49.Santiago |
| 24.Guadalupe | 50.Villaldama |
| 25.Hida.Hidalgo | |

Grafo de los 50 municipios seleccionados del estado de Nuevo León:

<https://github.com/PaulinaAguirre/1837503MC/blob/master/Proyecto/Grafo>

Resultados de la mejor ruta.

Con Kruskal:

Los Herreras, Melchor Ocampo, Gral. Treviño, Los Aldamas, Dr. Coss, Gral. Bravo, China, Gral. Terán, Linares, Hualahuises, Iturbide, Aramberri, Gral. Zaragoza, Mier y Noriega, Galeana, Rayones, Allende, Santiago, Monterrey, Guadalupe, Apodaca, San Nicolás, San Pedro, Escobedo, El Carmen, Abasolo, Hidalgo, García, Santa Catarina, Juárez, Pesquería, Marín, Zuazua, Ciénega de Flores, Higuera, Dr. González, Cerralvo, Agualeguas, Paras, Los Ramones, Cadereyta, Montemorelos, Sabinas, Villaldama, Bustamante, Mina, Lampazo del Naranjo, Dr. Arroyo, Anáhuac

Con un costo de: 1240

Con un tiempo de: 28.07

Con Vecino más cercano:

Marín, Higuera, Pesquería, Cadereyta, Dr. González, Cerralvo, Agualeguas, Melchor Ocampo, Gral. Treviño, Páras, Los Herreras, Los Aldamas, Dr. Coss, Gral. Bravo, China, Los Ramones, Gral. Zuazua, Apodaca, Guadalupe, Juárez, Monterrey, Santiago, Rayones, Galeana, Allende, Montemorelos, Hualahuises, Iturbide, Aramberri, Gral. Zaragoza, Mier y Noriega, Linares, Gral. Terán, San Nicolás, San Pedro, Santa Catarina, García, Escobedo, El Carmen, Villaldama, Lampazos del Naranjo, Dr. Arroyo, Anáhuac, Sabinas, Bustamante, Mina, Ciénega de Flores.

Con un costo de: 1322

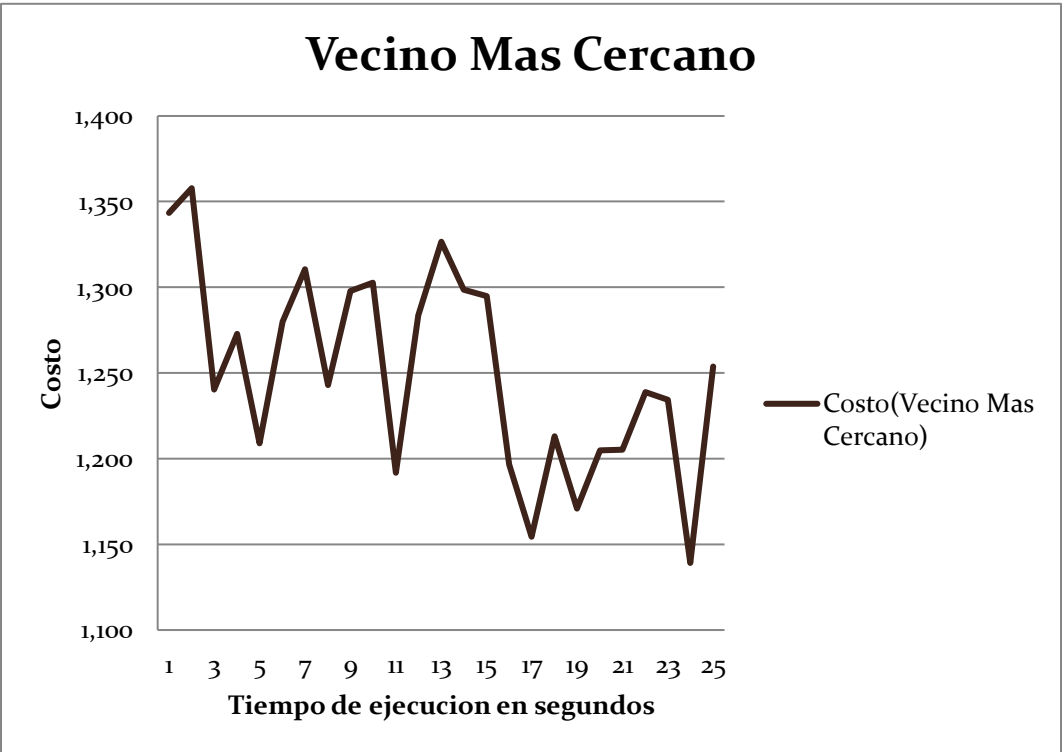
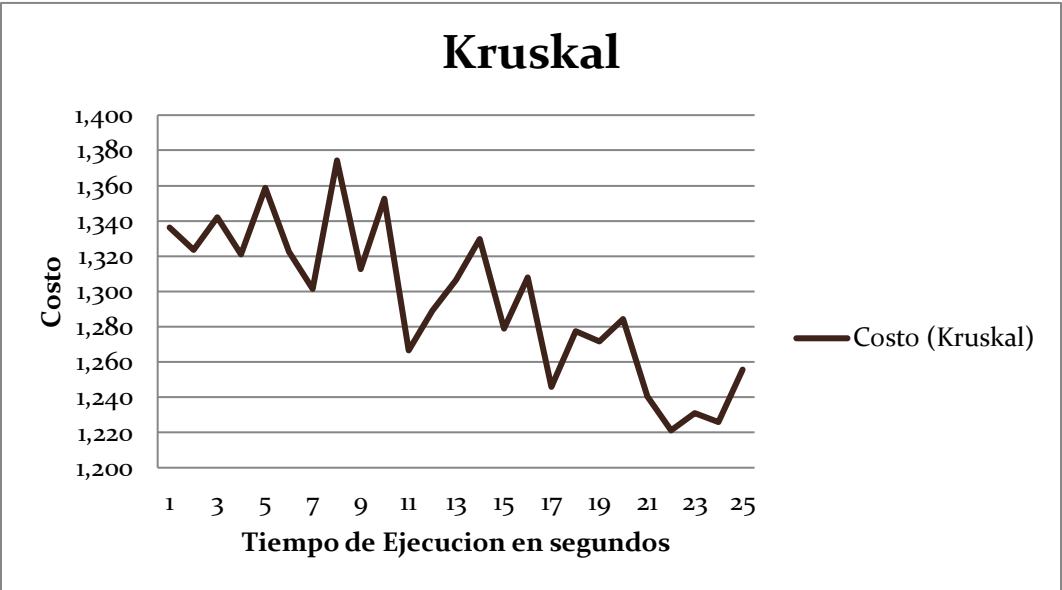
Con un tiempo de: 17.56

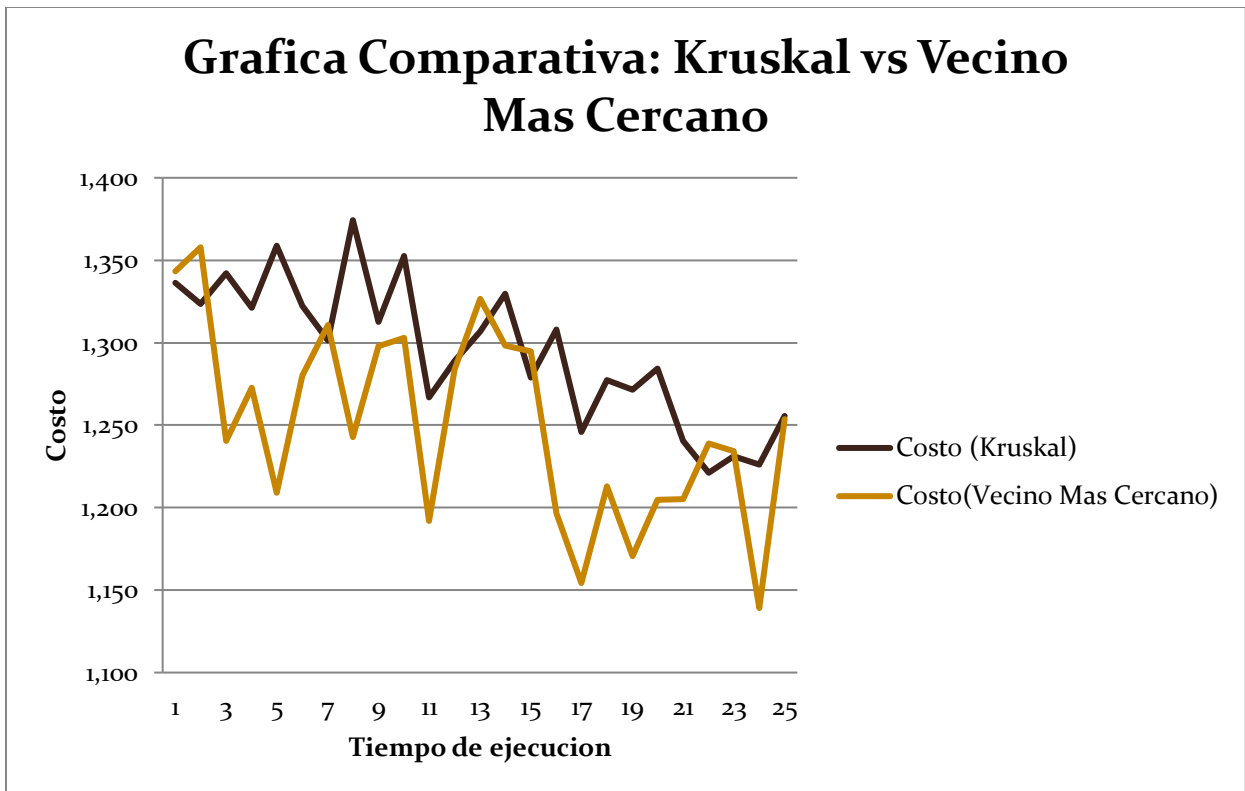
Además de obtener el camino más corto o con menor costo de nuestro grafo, es importante considerar el tiempo de ejecución de cada algoritmo, para esto se realizó una comparación en segundo entre los dos algoritmos.

Costo (Kruskal)	Tiempo de ejecucion
1,336	4
1,323	5
1,342	5
1,321	5
1,359	5
1,322	11
1,301	4.614880697
1374.326301	12
1,312	12.43095247
1,352	13
1266.722214	14
1289.087919	14.89325682
1306.632793	15.04399889
1329.555657	15
1,279	15.69292562
1307.860339	15.4115903
1245.740593	16
1277.448686	17
1271.452951	16.93546379
1284.312329	17.26813024
1240.547813	18.62541237
1221.159752	18.71496038
1230.960639	18.67018637
1226.060196	18.72125193
1255.53	18.72

Costo(Vecino Mas Cercano)	Tiempo de ejecucion
1,343	4
1,358	4
1,240	5
1,273	5
1,209	4
1280.04894	12
1,311	9
1,243	11
1,298	10
1,303	9
1191.774666	16.90452299
1283.81021	15.1003409
1326.632758	16.78426519
1298.474129	14.7961582
1294.718964	16.79461382
1196.762222	17
1154.469479	15
1212.961215	17
1170.796126	18
1204.793687	17
1,205	25.36
1239.02149	20.27782253
1234.250258	21.4791254
1139.145871	20.14820367
1253.712548	24.71290163

Obteniendo así las siguientes gráficas:





En base a las gráficas es posible observar que entre más tiempo se tarde se prueban más distancias, dando así una mayor probabilidad de que se encuentre un mejor camino.

No hay una notoria diferencia entre ambos algoritmos, ya que funcionan casi igual. Es decir, el tiempo no es un factor decisivo al momento de elegir que algoritmo usar.

Conclusiones.

Ya que los algoritmos probados difieren en una mínima cantidad de tiempo lo más óptimo y coherente sería utilizar el algoritmo que te designe el camino más corto, ya que ese es el objetivo de El Problema del viajero. En nuestro problema nuevo leones fue el Algoritmo Kruskal, una vez más es el ganador.

La parte enriquecedora de este proyecto fue el conocer los municipios de Nuevo León ya que para algunos de nosotros era de completa ignorancia, incluso el hecho de que fueran más de 50 municipios.