Reporte de algoritmo de Kruskal y Problema del Agente Viajero (PAV).

María Luisa Borrego Riojas
Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas
Matemáticas Computacionales

En este reporte se buscará conocer y analizar el algoritmo de Kruskal aplicado en Python. Así como se implementará para mostrar el problema del agente viajero.

I. Problema del Agente Viajero (PAV).

¿Qué es?

El problema del Agente Viajero se refiere a, dado un conjunto finito de puntos de interés (ciudades, tiendas, etc.) en donde trasladarse de un punto a otro indica un gasto, se trata de buscar la forma más eficiente de visitar todas las ciudades solamente una vez, regresando al punto de partida inicial.

¿Qué es lo complicado de él?

La dificultad de resolverlo recae en que mientras más puntos de interés se tenga, más posibles rutas surgen, en donde la manera de solucionarlo es comparar cada una de las rutas para así poder obtener la más corta, o la que implique menos gasto. Esto, sin embargo, puede llegar a ser un proceso complicado e ineficiente cuando el número de puntos de interés es mayor.

II. Conceptos importantes.

A. Algoritmo de aproximación.

Existen algoritmos utilizados para encontrar soluciones óptimas, y lo interesante de ellos es que son exactos. No obstante, su tiempo de corrida es exponencial, por lo que no son necesariamente óptimos.

Los algoritmos de aproximación surgen en la década de los 70 buscando una aproximación para resolver estos problemas.

B. Árbol de Expansión.

Un árbol de expansión consiste en una especie de diagrama de árbol en donde se tienen todos los vértices y algunas, o todas las aristas de un conjunto o grafo A. Las características de este árbol es que no contiene ciclos, además de que cada par de vértices tiene una ruta que los conecta.

Árbol de Expansión Mínima.

En este tipo de árbol de expansión se muestra el “camino” o las aristas que conectan a todo el grafo A con una longitud mínima.

III. El algoritmo de Kruskal.

Este algoritmo busca construir un árbol de expansión mínima del grafo A. Esto significa que la suma de todas las magnitudes de las aristas sea la menor posible. La complejidad de éste es de Ȯ (*nlogn*), lo que representa un tiempo considerable. Para realizar este algoritmo se siguen los siguientes pasos:

1. Se tiene un grafo vacío B para construir en él el árbol de expansión mínima.
2. Se obtienen las magnitudes de las aristas del grafo A de menor a mayor.
3. Al considerar una arista del grafo A, se necesita verificar que al momento de agregarla al grafo B no se forme ningún ciclo. Una vez verificado se añade al grafo B.
4. El proceso sigue hasta considerar todas las aristas de A, o bien, hasta que las aristas de B sean igual a n-1, en donde n es la cantidad de vértices de B.

Esto en un formato para Python sería algo así:

```
def kruskal(self):
    ordenw =deepcopy(self.aristas)
    orden =sorted(ordenw.keys(), key=lambda k: peso,arbo,n=0,
    grafo (),len(self.vertices)
    dsu=DSU(self.vertices)

    while len(orden)>0 and len(t_aristas)<2*(n-1):
(x,y)= orden.pop ()
          if dsu.union(x,y):
               t_conecta(x,y,self.aristas [(x,y)])
               peso +=self.aristas [(x,y)]
     print ("Arbol de expansion minima: ",peso)
     return arbol
```

IV. El algoritmo en práctica.

Se aplicará el algoritmo a un conjunto de 10 países del mundo, en donde se buscará encontrar la ruta que recorra la menor distancia, tomando en cuenta de que se parte de México, y se debe regresar ahí.

Los países seleccionados son:

1. México
2. Italia
3. Francia
4. Turquía
5. Australia
6. Grecia
7. Portugal
8. Suráfrica
9. Taiwán
10. Marruecos

Asimismo, una tabla con la distancia aproximada de cada uno de los países hacia los 9 restantes (En miles de km).

NOTA. Se está tomando en cuenta una distancia absoluta, esto quiere decir, no se hace una distinción en cuál fue el punto de partida.

	1. México	2. Italia	3. Francia	4. Turquía	5. Australia	6. Grecia	7. Portugal	8. Suráfrica	9. Taiwán	10.Marruecos
México	---	10.1	9.2	11.8	14.4	10.9	8.7	14.6	12.9	9.1
Italia	10.1	---	0.95	1.9	14.3	0.84	1.78	8.1	9.69	2.1
Francia	9.2	0.95	---	2.8	15.2	1.78	1.2	8.75	10.1	1.79
Turquí	11.8	1.9	2.8	---	12.4	1.16	3.71	7.8	8.04	3.89

Australia	14.4	14.3	15.2	12.4	---	12.6	16.1	10.4	5.59	16.2
Grecia	10.9	0.84	1.78	1.16	12.6	---	2.58	7.71	9.12	2.73
Portugal	8.7	1.78	1.2	3.71	16.1	2.58	---	8.39	11.3	0.85
Suráfrica	14.6	8.1	8.75	7.8	10.4	7.71	8.39	---	12.1	7.59
Taiwán	12.9	9.69	10.1	8.04	5.59	9.12	11.3	12.1	---	11.7
Marruecos	9.1	2.1	1.79	3.89	16.2	2.73	0.85	7.59	11.7	---

Conclusiones.

Personalmente, el algoritmo de Kruskal junto con el PAV me ha resultado especialmente interesante, pues me pareció que tiene una implementación muy práctica y muy real. Me gustó la lógica que seguía y a pesar de que al momento de hacerlo aplicado en Python no lo vi tan claro como la teoría general, creo que es de los algoritmos más ingeniosos que se ha

visto en la clase. Es un acercamiento importante entre Python y sus herramientas con nuestras vidas cotidianas.