UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS



FACULTAD DE INGENIERÍA

Trabajo Parcial

**INTEGRANTES**: RODRIGO GUADALUPE MEDINA

ANDRÉS LÓPEZ HURTADO

DIEGO SALAS NOAIN

LIMA – PERÚ 2017

El problema del agente viajero aplicado en Perú

**Introducción:**

El problema del agente viajero es uno de los problemas más conocidos por el mundo de la informática y la matemática computacional. Este problema trata sobre una persona que desea visitar varios puntos de una ciudad, pero para eso necesita hacer el recorrido de cada punto a otro en la menor distancia posible. Este problema es de difícil solución y, según lo investigado, no existe un algoritmo de resuelva este problema de manera óptima. Existen varios caminos para llegar de un punto a otro y, en general ”no se retorna la ruta más corta” Schrijver (2005). Sin embargo, en el mundo de las ciencias de computación existen muchos caminos para resolverlo, cada uno siendo más eficiente que otro. El problema del agente viajero tiene diversas aplicaciones, tales como: la planificación, logística y fabricación de circuitos electrónicos, secuencia de ADN, etc[ref]. Asimismo, en las aplicaciones de ahora como “waze” sabemos que utiliza este problema como modelo y lo trata de solucionar, ya que ellos implementan un algoritmo que busca las rutas más cortas y con menor tráfico; por esa razón, implementaremos un algoritmo para solucionarlo y a la vez ser el más eficiente, aplicando los temas que aprendimos durante el ciclo.

**Motivación:**

Deseamos resolver el problema del vendedor viajero puesto que es uno de los problemas más conocidos en el mundo de la informática y matemática, y representa un reto, el cual nos obliga a investigar sobre los algoritmos más convenientes a utilizar.

**Objetivos:**

* Implementar un algoritmo el cual resuelva de manera eficiente el problema de las conexiones de los centros poblados en el país.
* Para ciertas implementaciones, modificar el dataset.
* Analizar la conveniencia de usar las técnicas de “divide y vencerás”, “fuerza bruta” y “backtracking”
* Comparar las 3 diferentes implementaciones de los algoritmos que resuelvan el problema.
* Analizar la complejidad del algoritmo que se implementará.

**Marco teórico:**

Para el desarrollo del trabajo hemos utilizado el lenguaje de programación python para implementar los algoritmos de búsqueda. El algoritmo usado fue Dijsktra y Kruskal (para el árbol de expansión mínima) con diferentes implementaciones. Entre las estrategias usadas tenemos las siguientes:

1. Divide y vencerás:

Es una paradigma que divide a un problema en subproblemas parecidos al original que son resueltos por separado (cuando llegan a una condición de parada o caso base) de manera recursiva, y que finalmente son combinados para poder resolver el problema original.

Divide y vencerás fue usado para...

1. Fuerza Bruta:

También llamado búsqueda combinatoria o fuerza exhaustiva es una técnica sencilla, pero muy usada para casos de prueba rápidos. En este paradigma, se recorre toda la cantidad de elementos que se asignen para una iteración, por lo cual, para problemas con una gran cantidad de datos, posiblemente, esta no sea la técnica más conveniente a usar. Siempre que exista una solución al problema, esta técnica la en encontrará.

Utilizamos fuerza bruta para recorrer el grafo de nodos representados en un diccionario y establecer comparaciones entre los nodos adyacentes al nodo que se ha terminado de visitar.

Asimismo, no hemos considerado BFS, puesto que este algoritmo, a pesar de buscar el camino màs corto, solo lo hace en base a la cantidad de aristas recorridas. Es decir, el peso por arista en BFS es 1. Por otro lado, Dijsktra nos permite llevar un conteo de las rutas recorridas y sumar los pesos para llegar de un punto origen a un destino.

**Análisis de la complejidad:**

La complejidad del algoritmo depende mucho de la implementación del mismo, por lo que pasaremos a detallar la complejidad teórica, y la complejidad en la práctica. Teóricamente, la complejidad del algoritmo de Dijkstra es la siguiente:

* Si se usan pilas, la complejidad del algoritmo es de O(|E|+|V|log| V|)
* Si se usan pilas binarias O((|E| + |V|) log |V|)
* Si se usan arreglos no ordenados la complejidad será mucho mayor O(|V|^2)[REF]

**Conclusiones:**

* El programa no llega a cargar todos los datos del archivo csv de donde se extrae la información.
* Resulta difícil hacer que uno de los algoritmos implementados retorne el camino más corto, puesto que al crear las conexiones entre los centros poblados se hace de forma aleatoria.
* Es necesario utilizar algoritmos de backtracking para solucionar este problema cuando todas las ciudades están conectadas entre sí, pero esto puede consumir tanta memoria que podría no llegarse a ejecutar el programa.
* Aplicando el algoritmo de Kruskal para obtener el árbol de expansión mínima resulta ser costo en términos de tiempo puesto que asumimos que todos los centros poblados están conectados entre sí para, de esta forma, obtener las rutas más cortas posibles entre todos estos nodos.

BIBLIOGRAFIA

<https://www.quora.com/What-is-the-complexity-of-Dijkstras-algorithm>

<https://es.wikipedia.org/wiki/Problema_del_viajante>

https://medium.com/@andreaiacono/backtracking-explained-7450d6ef9e1a