

**Complejidad Algorítmica (CC184)**

**Informe del Trabajo Parcial**

**Integrantes:**

U201623394 - Victor Fernando Camones Ascurra

U201822752 - Carlos Marcelo Ronaldo Torre Calvo

**Profesor:** Canaval Sánchez, Luis Martín

**Sección:** CC42

**Ciclo:** 2021-01

Mayo - 2021

ÍNDICE

[Introducción 1](#_Toc71378996)

[Objetivos 1](#_Toc71378997)

[Marco teórico 2](#_Toc71378998)

[Análisis de la complejidad algorítmica 3](#_Toc71378999)

[Conclusiones 3](#_Toc71379000)

# Introducción

El presente proyecto realizado como trabajo parcial del curso de Complejidad Algorítmica, consiste en encontrar la ruta más corta que recorra todos los lugares que se nos lista. Contamos con un conjunto de datos clasificados que listan centros poblados del territorio peruano que se encuentran expuestos con un alto riesgo de inundación, y cada uno de estos cuentan con el departamento de origen, provincia a la que pertenecen, distrito en donde están ubicados y posición geográfica (en forma de coordenadas de latitud y longitud). A primera vista, se puede observar que el problema consiste en una adaptación del clásico problema del vendedor viajero, también conocido como Travelling Salesman Problem. El problema radica en encontrar la ruta más eficiente para que una persona, en este caso el vendedor que se encuentra viajando, pueda recorrer una lista específica de destinos. Este ejercicio ha posado un verdadero desafío para la comunidad científica de investigación de operaciones. Obviamente hay muchas rutas de las cuales elegir, pero encontrar la mejor o, en otras palabras, la que necesitará la menor distancia o costo, es lo que matemáticos y científicos de la computación han estado tratando de resolver por décadas. La principal motivación que nosotros tenemos para resolver este problema y desarrollar el proyecto de manera adecuada es, dado el contexto, ayudar a los mecanismos gubernamentales a encontrar la ruta más eficiente para poder visitar estas comunidades y brindarles la ayuda o asesoría correspondiente. Haciendo uso de los conocimientos adquiridos a lo largo del curso, podemos ofrecer una solución ofreciendo lo mejor de nuestras habilidades y que no requiera tanto tiempo computacional.

# Objetivos

El principal objetivo de este proyecto es resolver el Travelling Salesman Problem. Se nos pide, a partir de un dataset, obtener el listado de centros poblados ubicados en un distrito y provincia en particular. Luego, elaborar dos algoritmos para poder conectar todos los centros poblados con la menor distancia posible. Las dos soluciones tienen que funcionar para cualquier distrito y sus centros poblados que se encuentre en el dataset. Ambas soluciones deben implementarse utilizando algoritmos estudiados hasta ahora como fuerza bruta, backtracking, BFS, DFS o variantes y, por último, UCS o Dijkstra.

# Marco teórico

Primero, para idear soluciones viables, tuvimos que empezar a descartar ideas. Por ejemplo, no podemos utilizar Dijkstra porque el algoritmo encuentra la ruta más corta, no la que atraviesa todos los nodos, o en este caso los centros poblados, además que usualmente se utiliza este algoritmo con un origen y destino en mente.

***Fuerza bruta y Backtracking***

También conocido como el “Naive Approach”. Este método consiste en calcular todas las posibles rutas que recorran todos los nodos del grafo y calcular cual es la que menor distancia de recorrido tiene. Como todos los nodos se conectan entre sí, observamos que la ruta siempre será de la misma longitud que todos los nodos juntos. También, pudimos observar que la respuesta siempre sería un orden específico de todos estos nodos. En otras palabras, una permutación de todos los nodos. Para esto, se utilizó una función backtracking que permite hallar todas las permutaciones de una lista de números enteros. Estos números enteros representan a los nodos o centros poblados (eg. {0, 1, 2, 3, 4} si hay cinco ciudades). De esta manera se calculan todos los caminos que existen que recorran todos los centros poblados dentro del distrito. Aquí es donde solo tomamos en cuenta los caminos que empiezan con nuestra única variable de entrada de la función, el origen. Una vez tengamos los caminos, se calcula la distancia entre cada uno de los centros poblados, representados con valores enteros en el arreglo, en el orden de cada permutación. Es así donde al final, pasamos un filtro en donde se escoge el camino que menos distancia recorre. Esta será nuestra respuesta. La desventaja de este método es que la rapidez del algoritmo es inversamente proporcional al número de nodos. Mientras más nodos haya que evaluar, más lento será para encontrar una respuesta. Sin embargo, de esta manera, definitivamente, se obtendrá la respuesta más acertada.

***BFS del vecino más cercano***

Lo hemos llamado el método del vecino más cercano porque la lógica de este método es fundamentalmente una búsqueda constante del nodo de menor peso más próximo al nodo que está siendo evaluado. Empezamos con un punto de origen como única variable de la función. A partir de ese punto, utilizamos una búsqueda “Breadth-first search (BFS)” que tiene como función capturar los nodos vecinos y la distancia que toma en moverse hasta ellos. Cuando se están explorando, los añade a una lista en donde los ordena de menor a mayor, para que en la siguiente iteración del Loop, se escoja al primero, que es el nodo de menor distancia de separación, o también conocido como el nodo inmediato. Ahora el nodo más cercano se vuelve el nodo siendo explorado, y la lista de los nodos con sus respectivos valores de distancia de separación se reinicia. Luego, el proceso se repite nuevamente hasta haber recorrido todos los nodos del arreglo. Una de las ventajas de este método es que en la mayoría de las veces acierta con la respuesta con el método de fuerza bruta, qué es el método más seguro, pero la mayoría de las veces no termina con la respuesta más eficiente. Sin embargo, la verdadera fortaleza de este algoritmo es que es muchísimo más rápido que otros, ya que básicamente se apoya en la distancia euclidiana, que ya se calcula a la hora de generar el grafo.

***Método DFS recursivo***

Hicimos este método como parte de un experimento, en donde utilizamos el algoritmo de búsqueda Depth-First Search (DFS) para solucionar el problema de obtener todos los caminos posibles. Lo útil de este algoritmo es que puede ser utilizado con un origen y destino declarado, y también te calcula todas las rutas independientemente del tamaño que estas vayan a tener. Hemos programado este algoritmo para que solo nos de las rutas que recorren todos los centros poblados de un distrito. Tiene el mismo poder que el método de Fuerza Bruta, pero al mismo tiempo también sigue siendo igual o más lento.

# Análisis de la complejidad algorítmica

El análisis de la complejidad algorítmica para cada algoritmo utilizado en nuestra solución son los siguientes:

**Fuerza Bruta y Backtracking**: La solución de fuerza bruta tiene una complejidad de tiempo de O(n!). Ya que, se está calculando todas las permutaciones posibles de ‘n’ nodos.

**BFS del vecino más cercano y DFS**: La solución BFS y la del DFS tienen una complejidad de tiempo de O(n²). Usualmente, la complejidad de los algoritmos de búsqueda BFS o DFS es O (vértices + aristas). Como en este caso hay ‘n’ vértices y ‘n(n-1)’ aristas, ya que, todos los nodos están conectados entre sí, al operar n+n(n-1) esto da como resultado ‘n²’.

# Conclusiones

Este proyecto nos ha enseñado mucho más acerca de la complejidad de los algoritmos que hemos aprendido hasta ahora. También, que podemos encontrar muchas soluciones a este problema muy famoso, pero necesitamos mucho más tiempo de investigación. Nos hemos dado cuenta de que este problema es NP-difícil, y que, si podemos encontrar una solución en tiempo polinomial, cambiaría el campo de la ciencia de la computación para siempre.

Además, también pudimos aprender que la solución al problema del viajero vendedor tiene aplicaciones en el mundo real. Ya sea para disminuir los costos de transportación, al momento de calcular la ruta que menos distancia requiere y por consiguiente menos combustible utiliza. La minimización de costos es esencial en el mundo moderno empresarial y hay muchas variantes del TSP donde soluciones óptimas que requieren menor tiempo computacional se necesitan. Es gracias a este proceso de investigación que aprendimos acerca del VRP (Vehicle Routing Problem) y nos ha dejado muchas más ansias de aprender nuevos algoritmos.

Por último, no podemos dejar de lado la necesidad de las comunidades peruanas que corren riesgo de inundación. Por otro lado, pensamos que con una implementación de la solución a este problema del TSP, podríamos incluso encontrar la ruta más eficiente para conectar los pueblos que no tienen acceso al agua y que darle este beneficio a menor costo del presupuesto gubernamental.