**INFORME TRABAJO PARCIAL**

**Integrantes:**

**Daniel Núñez Robinson,**

**Diego Bedriñana Alberca,**

**Marek Joel Rivas Zavala**

**INTRODUCCION**

En el presente informe se detallará la investigación, experimentación y resultados obtenidos al realizar el Problema del Vendedor Viajero (TSP). Este problema nos plantea lo siguiente: dada una lista de ciudades y las distancias entre cada par de ellas. ¿Cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad de origen?

La motivación de este trabajo se encuentra en la misma complejidad del problema. Dado que con una menor cantidad de datos la solución resulta sencilla, al momento de incrementarse alrededor de 150,000 ciudades genera un interés respecto al porqué la solución sencilla no funciona.

**OBJETIVOS**

* Investigar las diferencias entre métodos de solución planteados para el TSP.
* Implementar un algoritmo por cada integrante con la intención de analizar y buscar posibles mejoras dentro de estos, o corroborar si no son viables para la solución del problema.
* Plantear posibles mejoras o nuevas ideas algorítmicas para la solución del TSP

**MARCO TEORICO**

1. **Fuerza Bruta:**

Es la manera más básica y puede ser la más extensa de entre todos los algoritmos de búsqueda.

Es un algoritmo en el cual se prueba todos los caminos o elecciones para llegar a la o las soluciones posibles.

Ejemplo:

Si para encontrar el Mínimo Común Múltiplo de “x” e “y” lo resolviéramos con este algoritmo el Código sería de la siguiente forma.

x = 35

y = 54

num = x

while 1:

if num% x == 0 and num % y == 0:

break

num+=1

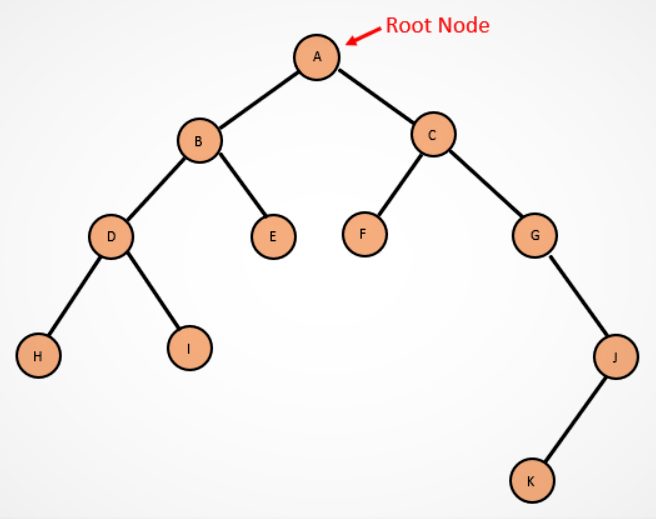
Aquí prueba cada una de las posibles soluciones hasta llegar a la solución.

1. **Breadth first search:**

O búsqueda en amplitud, es un algoritmo que ve a que posiciones inmediatas siguientes puede acceder y así en cada una de sus iteraciones.

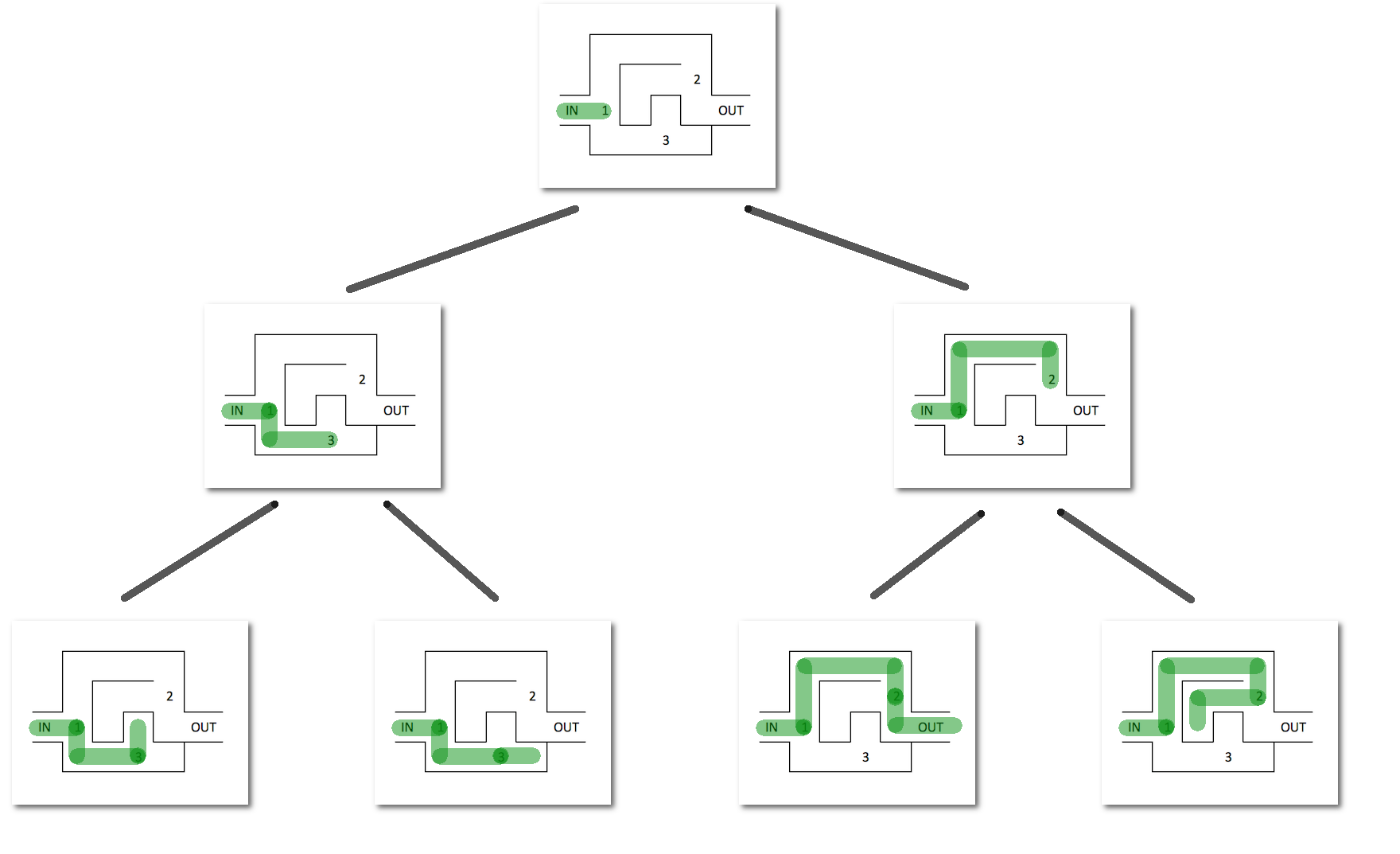
1. **Depth for first search:**

Atraviesa un árbol desde el vértice hasta un nodo hoja. Esto quiere decir que escoge un solo camino y lo recorre totalmente hasta el último estado en el que puede estar.



1. **Backtracking:**

Es un algoritmo sencillo y parecido a Fuerza Bruta hasta cierto punto. Consiste en que cuando se llega a un punto en el que el siguiente paso ya no es una posible solución y por ende los siguientes estados tampoco (nodos hijos) frena allí, ya no avanza por ese camino y retrocese.



**ANALISIS**

1. Fuerza Bruta

Procedimiento de resolución con fuerza bruta: La idea de esta solución consiste en vincular los códigos de cada centro poblado y conseguir 2 posibles candidatos para efectuar la vinculación de nodo origen -> nodo destino. Para lograr esto se ingresan los datos con un comando de ‘getline’ gracias a ‘comma separated values’ de los datos de Excel que trasladamos a un .txt y se comienza el algoritmo. En primer lugar, se forma un doble ‘for’ para seleccionar el dato de nodo origen y la comparación de este dato con los otros restantes. Por un lado, en cuanto a las coordenadas X se busca cuál de todos tiene una diferencia de valores menor que cualquier otra y se guarda esa posición como candidato a ser vinculado. Por otro lado, se hace lo mismo con las coordenadas X y luego se genera una suma para cada una de las 2 posiciones (diferencia de x + diferencia de y para candidato 1 y candidato 2) y se comparan. El motivo de esta comparación es que aquella que tenga una menor distancia sumada será el candidato elegido para el vínculo. Una vez que se elija esto la posición será validada con un bool para que la posición no pueda salir a dos nodos. Este tipo de comparaciones harán que, a largo plazo, se vayan procesando menos datos mientras más avance el código. Para finalizar el código, se escribe formalmente qué nodo se vincula con cuál y el i procede a sumarse nuevamente para continuar las comparaciones. La notación Big O es de O(n^2) ya que, en esencia, el código es simplemente un doble ‘for’ por más que se haya utilizado recursividad para facilitar el procesamiento de datos.

En conclusión, este tipo de solución no es óptimo sin embargo lentamente puede lograr cumplir con solucionar una pequeña cantidad del problema que otros códigos no podrían si es que se utilizaran métodos que llenen la pila aún más como divide y vencerás o una recursividad más severa.

1. Djikstra

Se utilizó un algoritmo muy parecido a Dijkstra, pero solo de un nivel, en otras palabras, lo que hace es buscar el nodo no visitado de entre sus vecinos, se mueve a él y luego repite, así hasta llegar al nodo objetivo.

Problemas con mis códigos:

En el código “MiCodigo3.ipynb” en el que uso recursividad se llega a un límite de 3317 llamadas a la función recursiva.

En el código “MiCodigo4.ipynb” en el que uso un for acepta más de 10000 Nodos, pero no los 145255 Nodos. Aparece un error de “list index out of range”.

1. Fuerza Bruta con Pitágoras y sectores

La idea general de Fuerza Bruta se mantiene, sin embargo, se cambian dos elementos:

* Los datos ingresados ya se encuentran ordenados del mas cercano al mas lejano respecto al punto de origen
* Al momento de comparar cual es la ciudad mas cercana solo se usa una parte de los datos ingresados (los 10 más cercanos al nodo basado en su posición)

La primera modificación se logra creando una lista de datos extra y reordenándolas en base a que tan alto es su valor pitagórico (distancia respecto al punto de origen). La segunda modificación se emplea sobre la lista auxiliar.

Con estas modificaciones, se espera delimitar la búsqueda para que nos bote un camino óptimo. Sin embargo, este camino no es necesariamente el mejor, pero si uno de los mejores.

Al finalizar este algoritmo, su tiempo asintótico se mantiene igual que el de fuerza bruta, siendo este O(n^2).

El principal problema con este algoritmo es que no necesariamente los nodos mas cercanos al origen van a ser los más cercanos entre sí, pues algunos nodos pueden estar a una distancia similar del origen, pero por el otro lado del plano (posición negativa).

Esto hace que la creación del camino, a nivel gráfico, sea similar a un BFS, cuando cabe la posibilidad que una búsqueda DFS pueda ser la óptima.

En conclusión, este algoritmo se demoraría menos en ejecutarse que un algoritmo de Fuerza Bruta tradicional, sin embargo, solo garantiza obtener un buen camino, no el mejor.

**CONCLUSIONES**

En conclusión, el problema del Vendedor Viajero presenta un gran reto para la Informática, pues a pesar de las múltiples soluciones planteadas por diferentes personas, sigue siendo irresolvible en un tiempo aceptable.

Los datos recabados para el desarrollo de este informe sirvieron para entender las soluciones planteadas previamente por otros trabajos y como punto de partida para la creación de los algoritmos presentados.

Los algoritmos planteados, sirvieron para probar nuevas soluciones al problema e intentar mejorar algoritmos ya planteados, pero lamentablemente solo se pudo comprobar que no son el camino adecuado a seguir para la resolución del problema.

Dado la complejidad del problema, lo más recomendable para futuras investigaciones sería centrarse más en la parte matemático-lógica del problema.

**BIBLIOGRAFÍA:**

* What is brute force in programming? (<https://www.quora.com/What-is-brute-force-in-programming>)
* Breadth first search and depth first search (<https://www.ics.uci.edu/~eppstein/161/960215.html>)
* Depth First Search Algorithm: What it is and How it Works (<https://edgylabs.com/depth-first-search-algorithm-what-it-is-and-how-it-works>)
* Backtracking explained (<https://medium.com/@andreaiacono/backtracking-explained-7450d6ef9e1a>)