Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

TC2038 - Análisis y diseño de algoritmos avanzados

Grupo: 601

# E2. Actividad Integradora 2 Reflexión Individual

Profesores:
Luis Humberto González Guerra

Humberto Genaro Cisneros Salinas - A01723264

28 de noviembre del 2025

# Reflexión

A lo largo de toda la actividad se utilizaron dos mapas globales para poder llevar una conversión directa en tiempo constante O(1) del nombre de la colonia a su índice en el grafo y vice versa.

Primero, se leen los datos de entrada del programa. Se crean las colonias bajo la estructura de la colonia con los atributos de  nombre, coordenadas y si es central o no. Se agrega la referencia de su nombre e índice a los mapas globales. También se leen las conexiones entre las colonias y se agregan al grafo. Después se leen las nuevas conexiones y se agregan a la propiedad de built Edges del grafo. Por último, se leen las nuevas colonias y se agrega su referencia a los mapas globales.

Habíamos tenido un acercamiento incorrecto al problema y decidimos cambiar nuestra estrategia para utilizar la matriz de costos generada por el algoritmo de Floyd Warshall para el problema 2 y 3. Para esto, se hizo un pre procesamiento antes de empezar con los 4 problemas para obtener la matriz de costos y la matriz auxiliar “distAux” para poder usarlas en estos dos problemas. Para la matriz “distAux” usamos una matriz del siguiente nodo en el trayecto (se explica más adelante) y con ella se pudo reconstruir el camino otorgado por el algoritmo de TSP con programación dinámica.

Para el primer problema se identificó la necesidad de obtener el Minimum Spanning Tree del grafo, tomando en cuenta que ya existen conexiones construidas con el nuevo cableado. Para esto se utilizó el algoritmo de Kruskal, en donde se agregan las conexiones existentes antes de comenzar la ejecución. A lo largo de la ejecución se guarda el costo del MST y las conexiones identificadas como parte del mismo para poder desplegarlo como parte de la solución. La complejidad total de esta implementación es O(E*logE + E*logV) ya que E*logE es lo que toma ordenar las aristas y para cada arista se debe realizar la operación de unión con complejidad O(logV). Asumiendo que la cantidad de conexiones supera a la cantidad de colonias, la complejidad simplificada sería de O(E*logE).

Para el segundo problema se toma el algoritmo de programación dinámica con bit masking para el problema del viajero (TSP). Se recibe la matriz de distancias y la matriz auxiliar de Floyd Warshall y se usa la matriz de distancias como la matriz de costos del problema. También se hace una máscara con solo las colonias no centrales “encendidas” para poder tener una máscara de referencia para saber que ya se visitaron todas las colonias requeridas. Se pudo dejar esta máscara como tipo int porque no pueden haber más de 30 nodos e int puede almacenar hasta 32. A ojos del algoritmo del TSP, es como si las colonias centrales no existieran, cuando en su ciclo se topa con una, se las salta. Su aparición vendrá de que sean parte de un camino óptimo almacenado en distAux. Durante la ejecución del algoritmo, se hace la suma recursiva con una función de safeSum, debido a que en pruebas del problema había problemas con overflow al momento de usar INT_MAX en uno de los dos valores. Al final de la función de rutaOptima, se reconstruye la ruta óptima en un vector de la matriz de mejores candidatos o nextBest, está reconstruye la ruta al revés. Después, se utiliza esa ruta para

construir la verdadera ruta con la matriz de distAux y la función addToPath. Finalmente, se despliega la ruta óptima junto con su costo. La implementación del problema del viajero tiene una complejidad temporal de O(n^2*2^n) al tener dos parámetros que cambian con cada llamada recursiva.

Para el tercer problema se busca generar la ruta óptima entre cada nodo que es central. Se utiliza el algoritmo de Floyd Warshall para encontrar el camino más corto entre todos los nodos del grafo, guardando el siguiente nodo en un camino si este demuestra ser más eficiente. Esta matriz te dice el siguiente nodo en un trayecto. Este algoritmo tiene una complejidad cúbica de O(n^3) por su triple ciclo for anidado. Una vez establecido la matriz de distancias más cortas y su auxiliar para los caminos, se despliegan aquellos que representan recorridos de una central a otra. Este fue el algoritmo que utilizamos debido a que puede encontrar las distancias más cortas entre todos los nodos y además sirve para resolver el problema anterior.

Es importante mencionar que addToPath y printPath siguen prácticamente la misma lógica para recorrer distAux y se dejó el caso en donde no existe una ruta entre colonias, pero es algo que no se espera que pase debido al planteamiento del problema.

Finalmente, para el cuarto problema se busca el punto con la distancia más cercana a las colonias nuevas que se buscan agregar al grafo existente. Para esto se busca entre todas las colonias existentes la colonia con la distancia euclidiana más cercana a la colonia que se desea agregar y se despliega. Este algoritmo tiene una complejidad temporal de O(n*m) ya que para cada colonia nueva a agregar (m) se debe buscar entre todos las existentes en el plano (n).