

# Informe para el Trabajo Práctico Especial - Programación III

Cagliolo, Camilo & Fonzo Baniles, Luca

02/07/2023

## Consigna

*Las autoridades de una ciudad deciden construir una red de subterráneos para resolver los constantes problemas de tráfico. La ciudad ya cuenta con  $N$  estaciones construidas, pero todavía no tienen ningún túnel que conecte ningún par de estaciones entre sí. La red de subterráneos que se construya debe incluir a todas las estaciones (es decir, que de cualquier estación  $H$  pueda llegar a cualquier otra estación  $J$ , ya sea de manera directa o atravesando otras estaciones). Sin embargo, debido al acotado presupuesto, las autoridades desean construir la menor cantidad de metros de túnel posibles. Para esto han calculado cuantos metros de túnel serían necesarios para conectar de manera directa cada par de estaciones existentes.*

## Objetivo

El objetivo es resolver el problema planteado mediante dos técnicas algorítmicas distintas: Backtracking y Greedy. Luego se deberán comparar los resultados teniendo en cuenta distintas métricas que permitan visualizar, mínimamente, la calidad de la solución y el costo de obtener dicha solución, con ambas técnicas.

## Solución

Los objetos a ser representados -estaciones y túneles- en el sistema se adaptan perfectamente a una estructura de datos de tipo grafo. Para ello se utilizará la estructura de datos implementada en la primera entrega del Trabajo Práctico Especial de la materia. Además, en esta estructura de grafo no dirigido, el objetivo equivale a encontrar el *minimum spanning tree* o árbol expandido mínimo del grafo.

## Algoritmo greedy

**Estrategia.** La estrategia empleada para la implementación del algoritmo greedy consistió en seleccionar para cada vértice del grafo el arco más corto disponible; los arcos disponibles son aquellos arcos que no tuvieran como vértice destino un vértice con un arco ya asignado. Esta estrategia casi nunca dará la solución óptima porque no tiene la capacidad de formar *redes* (es decir, vértices



lo contrario no tiene sentido seguir construyendo: las distancias son cantidades siempre positivas. Esto permite acortar el tiempo de ejecución eliminando varias exploraciones subóptimas.

## Análisis

El cuadro 1 muestra una tabla comparativa de los resultados y sus diferencias entre los algoritmos con estrategias *greedy* y *backtracking*, analizando el tiempo de ejecución y la cantidad de ciclos, además de la complejidad computacional teórica.

Algoritmo	Dataset 1	Dataset 2	Dataset 3
Greedy $O(n^2)$	Tiempo: 0,248 <i>ms</i> Ciclos: 14 Distancia total: 80 <i>km</i>	Tiempo: 0,499 <i>ms</i> Ciclos: 27 Distancia total: 145 <i>km</i>	Tiempo: 0,466 <i>ms</i> Ciclos: 104 Distancia total: 520 <i>km</i>
Backtracking $O(n!)$	Tiempo: 1,609 <i>ms</i> Ciclos: 86 Distancia total: 55 <i>km</i>	Tiempo: 15,83 <i>ms</i> Ciclos: 1686 Distancia total: 135 <i>km</i>	Tiempo: 10,45 <i>min</i> Ciclos: 1,182,011,744 Distancia total: 450 <i>km</i>

Cuadro 1: Comparación de costos y calidad de solución entre las dos estrategias.

## Conclusiones

A partir de lo analizado y lo trabajado en clase, es posible llegar a las siguientes conclusiones:

- Los algoritmos de tipo greedy pueden otorgar soluciones que están muy cerca de ser la óptima, consumiendo muchos menos recursos que su contraparte de backtracking. Sin embargo, no hay forma de saber a priori -en general- qué tan óptima es la solución, pues eso dependerá íntegramente del problema particular y el algoritmo implementado.
- Los algoritmos de backtracking aseguran la solución más óptima, por lo que si el costo computacional es pagable, pueden ser útiles.
- Una combinación de ambas metodologías podría permitir podar gran parte del árbol de búsqueda del algoritmo de backtracking, utilizando como filtro los resultados del algoritmo greedy.
- El tiempo de ejecución para los algoritmos de backtracking escala enormemente con el tamaño de la entrada, volviéndose rápidamente una estrategia inviable.
- Existen algoritmos de tipo greedy con estrategias considerablemente mejores que la elegida. Particularmente los algoritmos de Prim o Kruskal, que además de tener complejidades generalmente menores, aseguran la solución correcta.