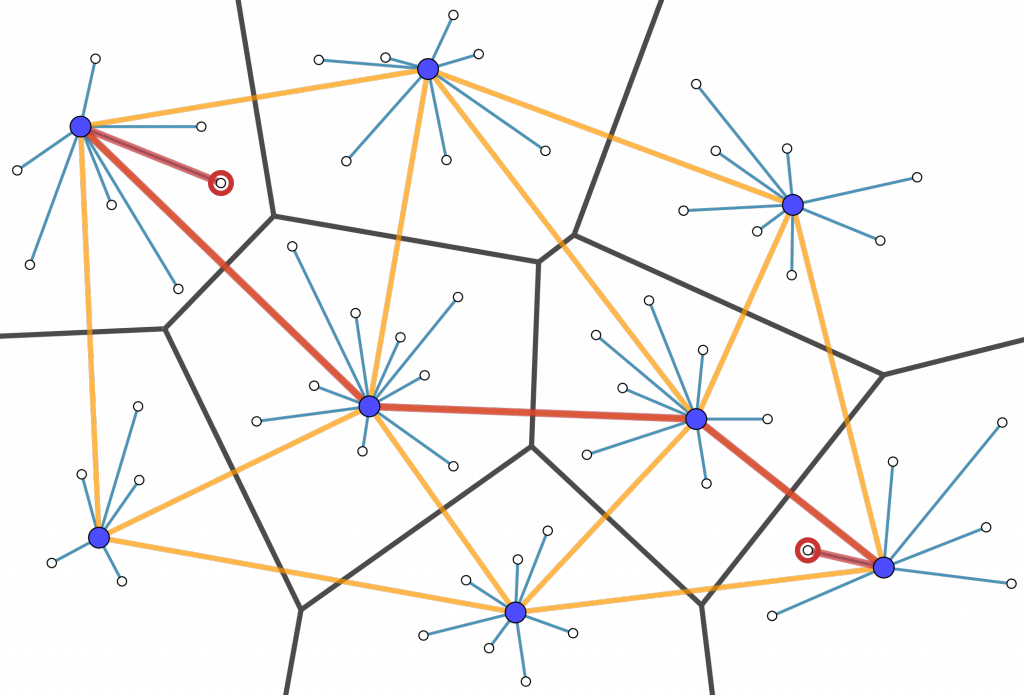
Trabajo practico Especial

Programación III



Integrantes: Menchón Agustin

Repositorio GitHub: <https://github.com/agusmenchon/TPE-Prog3>

Fecha de entrega: 2/7/2023

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 de esta segunda parte del trabajo será 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 utilizando Algoritmo Greedy

Se llevo a cabo el algoritmo de PRIM (una variante de algoritmos voraces) por su fácil ejecución, sumado a que siempre encuentra la solución óptima todos los casos. En cada iteración, el algoritmo selecciona el arco con menor peso de una cola de prioridad que se ordena por etiqueta y agrega el vértice conectado a la lista de recubrimiento mínima. El paso siguiente es actualizar la cola agregando las aristas del vértice nuevo. Una vez que la cola este vacía o la lista tenga el mismo tamaño que la cantidad de vértices del grafo termina la iteración dando como resultado el árbol de recubrimiento mínimo.

La calidad de la solución de la estrategia greedy seleccionada siempre da la óptima siempre.

*Costo computacional:* Debido al funcionamiento del algoritmo y que utiliza una PriorityQueue - las operaciones en la cola de prioridad tienen un costo logarítmico-, en el proceso de selección se agregan todos los arcos del vértice visitado a la cola y se utiliza la arista de menor valor, por lo tanto, se recorran todos los arcos y todos los vértices del grafo. La complejidad computacional es O((N + A) log N), donde V es el número de vértices y A es el número de arcos en el grafo.

Solución utilizando Algoritmo Backtracking

En la estrategia utilizada, se recorren todos los vértices del grafo y se agregan a una lista de noVisitados, en la función backtracking verifica si la lista de noVisitados esta vacía, en ese caso se actualiza la lista de recorrido mínimo con la distancia correspondiente. En el caso de que queden vértices por visitar, se itera sobre los vértices no visitados. Para cada vértice no visitado, se obtiene el arco que conecta el vértice actual con el vértice no visitado. Si el arco existe y la suma de la distancia actual y la etiqueta del arco es menor que la distancia del recorrido menor, se agrega el arco al recorrido parcial, se realiza la llamada recursiva para explorar el siguiente vértice y luego se elimina el arco del recorrido parcial.

Finalmente, se verifica si existe un vértice anterior y se realiza una llamada recursiva con el vértice anterior como nuevo vértice actual. el algoritmo utiliza la recursión para explorar todas las posibles soluciones en un grafo. A medida que avanza, va construyendo un recorrido parcial y actualiza el recorrido mínimo si encuentra un recorrido más corto

*Costo computacional:* En el peor caso, donde el grafo es completo y no hay poda, el algoritmo tendría una complejidad de tiempo exponencial, O(2^n), donde n es el número de vértices del grafo. Esto se debe a que el algoritmo realiza una exploración de todas las posibles combinaciones de vértices, lo cual implica que, para cada vértice, se generan recursivamente nuevas ramas para cada vértice no visitado, hasta que se llega a un caso donde no quedan vértices por visitar.

Tabla comparativa para las distintas entradas posibles.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algoritmo** | **Dataset1** | **Dataset2** | **Dataset3** |
| *Greedy (PRIM)* | Distancia: 55km  Ciclos: **6** | Distancia:135km  Ciclos:**15** | Distancia:440km  Ciclos:**78** |
| *Backtracking* | Distancia:55km  Ciclos: **29** | Distancia:135km  Ciclos: **263** | Distancia: 440km  Ciclos: **107996289** |

Conclusión:

* El algoritmo de backtracking aseguran la mejor solución, pero el costo computacional es tan alto que se debe evaluar si es útil para tal caso.
* El tiempo de ejecución del algoritmo backtracking asciende exponencialmente a medida que el grafo se incrementa.
* La estrategia greedy por lo general no suele encontrar una solución óptima, pero si es una posibilidad para encontrar una solucion aproximada con muy poco costo computacional.
* Es importante analizar si con variantes de algoritmos voraces se puede llegar a una solucion optima, en este caso con PRIM nos aseguramos la correcta