**PROYECTO FINAL AED – Administración de Red mediante Protocolos**

**de Enrutamiento**

**Entrega No 2- FASE 6 DEL MÉTODO DE LA INGENIERÍA.**

**Presentado por:** Marisol Giraldo Cobo – **Código:** A00246380

**Presentado al Profesor:** Andrés Aristizábal

**PASO 6. PREPARACIÓN DE INFORMES Y ESPECIFICACIONES**

|  |  |
| --- | --- |
|  | Laboratorio I  Algoritmos y Estructuras de Datos  Diagrama de clases |
| Nombre del Proyecto: | **– Administración de Red mediante Protocolos**  **de Enrutamiento** |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andrés Aristizábal |
| Fecha: | Mayo 2 del 2020 |

**A close up of text on a white background

Description automatically generated**

|  |  |
| --- | --- |
|  | Laboratorio I  Algoritmos y Estructuras de Datos  Diseño de Pruebas |
| Nombre del Proyecto: | **– Administración de Red mediante Protocolos**  **de Enrutamiento** |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andrés Aristizábal |
| Fecha: | Mayo 2 del 2020 |

**GRAFO DE PRUEBA No 1**

|  |  |
| --- | --- |
| Clase | Principal |
| Método | BFSAdjacencyMatrix |
| Caso de Prueba | |
| 1 | |
| Objetivo | |
| Probar que el método retorna una búsqueda de los nodos por medio del recorrido de anchura | |
| Entradas | |
| NodoDeLaRed y grafo de prueba No 1 | |
| Salidas | |
| Una cadena con el siguiente recorrido: 2, 4, 1, 3, 0 | |
| Procedimiento | |
| La estrategia que usaremos para garantizar este recorrido es utilizar una cola que nos permita almacenar temporalmente todos los nodos de un nivel, para ser procesados antes de pasar al siguiente nivel hasta que la cola esté vacía. | |

|  |  |
| --- | --- |
| Clase | Principal |
| Método | DFSAdjacencyMatrix |
| Caso de Prueba | |
| 2 | |
| Objetivo | |
| Probar que el método retorna una búsqueda de los nodos por medio del recorrido por profundidad | |
| Entradas | |
| NodoDeLaRed, un arreglo de nodos visitados que esta vacío y grafo de prueba No 1 | |
| Salidas | |
| Una cadena con el siguiente recorrido: 2, 4, 3, 0 | |
| Procedimiento | |
| La estrategia que usaremos para garantizar este recorrido es recorrer desde la raíz hasta los nodos extremos u hojas por cada una de las ramas. En este caso los niveles de cada nodo no son importantes. | |

**GRAFO DE PRUEBA No 2**

1

3

55

75

7

25

15

**Nodo Origen**

|  |  |
| --- | --- |
| Clase | Principal |
| Método | DijkstraAdjacencyMatrix |
| Caso de Prueba | |
| 3 | |
| Objetivo | |
| Probar que el método retorna la ruta más corta desde un nodo origen hacia los demás nodos. | |
| Entradas | |
| NodoDeLaRed y grafo de prueba No 2 | |
| Salidas | |
| Una cadena con el siguiente recorrido: C->C =0, C -> A = 1, C -> D= 2, C ->B = 4, C->E =5 | |
| Procedimiento | |
| Durante la ejecución del algoritmo, iremos marcando cada nodo con su **distancia mínima** al nodo C (nuestro nodo elegido). Para el nodo C, esta distancia es 0. Para el resto de los nodos, como todavía no conocemos esa distancia mínima, empieza siendo infinita (∞): También tendremos un **nodo actual**. Inicialmente, el nodo actual será C (nuestro nodo elegido).   1. Se marca el nodo inicial que elegiste con una distancia actual de 0 y el resto con infinito. 2. Se establece el nodo no visitado con la menor distancia actual como el nodo actual A. 3. Para cada vecino V de tu nodo actual A: suma la distancia actual de A con el peso de la arista que conecta a A con V. Si el resultado es menor que la distancia actual de V, establécelo como la nueva distancia actual de V. 4. Marca el nodo actual A como visitado. 5. Si hay nodos no visitados, ve al paso 2. | |

**GRAFO DE PRUEBA No 3**

1

4

5

1

2

2

|  |  |
| --- | --- |
| Clase | Principal |
| Método | FloydWarshallAdjacencyMatrix() |
| Caso de Prueba | |
| 4 | |
| Objetivo | |
| Probar que el método compara todos los posibles caminos entre cada par de nodos. Esto se consigue al ir mejorando un estimado de la distancia entre dos nodos, hasta que el estimado es óptimo. | |
| Entradas | |
| No necesita entrada. Utiliza toda la estructura de datos. Grafo de Prueba No 3 | |
| Salidas | |
| Una matriz:   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 1 | 2 | 3 | 4 | | 1 | - | 2 | 3 | 4 | | 2 | 1 | - | 1 | 4 | | 3 | 1 | 2 | - | 4 | | 4 | 1 | 2 | 3 | - | | |
| Procedimiento | |
| La estrategia que usaremos para garantizar este recorrido es: A partir de una tabla inicial compuesta de 0`s (no hay correspondencia inicial en el grafo) y 1`s (hay una correspondencia, llamase “flecha”, entre nodos), obtiene una nueva matriz denominada “Matriz de Clausura Transitiva” en la que se muestran todas las posibles uniones entre nodos, directa o indirectamente. Es decir, si de “A” a “B” no hay una “flecha”, es posible que si haya de “A” a “C” y luego de “C” a “B”. Luego, este resultado se verá volcado en la matriz final. | |

**GRAFO DE PRUEBA No 4**

8

7

9

5

5

7

15

8

9

6

11

|  |  |
| --- | --- |
| Clase | Principal |
| Método | PrimAdjacencyMatrix |
| Caso de Prueba | |
| 5 | |
| Objetivo | |
| Probar que el método retorna un árbol de expansión mínima. Es decir, es capaz de encontrar un subconjunto de las aristas que formen un árbol que incluya todos los vértices del grafo inicial, donde el peso total de las aristas del árbol es el mínimo posible. | |
| Entradas | |
| No necesita entrada. Utiliza toda la estructura de datos. Grafo de Prueba No 4 | |
| Salidas | |
| Una cadena del árbol de expansión mínima con un peso de 39. | |
| Procedimiento | |
| 1. Se marca un vértice cualquiera. Será el vértice de partida. 2. Se selecciona la arista de menorpeso incidente en el vértice seleccionado anteriormente y se selecciona el otro vértice en el que incide dicha arista. 3. Repetir el paso 2 siempre que la arista elegida enlace un vértice seleccionado y otro que no lo esté. Es decir, siempre que la arista elegida no cree ningún ciclo. 4. El árbol de expansión mínima será encontrado cuando hayan sido seleccionados todos los vértices del grafo. | |

|  |  |
| --- | --- |
| Clase | Principal |
| Método | KruskalAdjacencyMatrix |
| Caso de Prueba | |
| 6 | |
| Objetivo | |
| Probar que el método retorna un árbol de expansión mínima. Es decir, es capaz de encontrar un subconjunto de las aristas que formen un árbol que incluya todos los vértices del grafo inicial, donde el peso total de las aristas del árbol es el mínimo posible. | |
| Entradas | |
| No necesita entrada. Utiliza toda la estructura de datos. Grafo de Prueba No 4 | |
| Salidas | |
| Una cadena del árbol de expansión mínima con un peso de 39. | |
| Procedimiento | |
| 1. Se selecciona, de entre todas las aristas restantes, la de menor peso siempre que no cree ningún ciclo. 2. Se repite el paso 1 hasta que se hayan seleccionado |V| - 1 aristas. Siendo V el número de vértices. | |