**UNIVERSIDAD ICESI**

*Facultad de Ingeniería*

*Periodo II de 2018*

***Algoritmos y Estructuras de Datos***

C. Gironza1, N. Salazar1, C. Olano2.

*1Ingeniería telemática, Facultad de ingeniería, Universidad Icesi, Lina Marcela Quintero Villareal,Cali Colombia.*

2*Ingeniería industrial, Facultad de ingeniería, Universidad Icesi,* [*Norha Milena Villegas Machado*](https://www.icesi.edu.co/profesores/cv/norha-villegas)*, Cali Colombia*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
*Proyecto Final de Algoritmos y Estructuras de Datos*

**1.Enunciado del Problema:**

El laberinto y el ratón

Un laberinto es un lugar formado por calles y encrucijadas, intencionadamente complejo para confundir a quien se adentre en el mismo, un laberinto tiene al menos un camino a la salida, existen muchos juegos y/o experimentos relacionados con laberintos, como por ejemplo el Experimento Kerplunk, un experimento de estimulo respuesta, el cual consiste en colocar un ratón (individuo de estudio) dentro de un laberinto el animal debe de buscar la salida o recompensa, el objetivo de este experimento es exponer como la repetición de una tarea fomenta el aprendizaje mejorando la confianza del individuo.

Se pretende crear una simulación del experimento de Kerplunk, con un laberinto que tenga al menos un camino a la salida, y un ratón que deberá salir del mismo, el usuario deberá ayudar al ratón salir del laberinto a través del camino mas corto el cual el deberá seleccionar a partir de sus cálculos, el programa debe indicarle al usuario los caminos posibles desde el inicio del laberinto hasta el final del mismo, el ratón debe poder moverse por el camino seleccionado (debe tener la animación de movimiento), al final del recorrido del ratón el programa debe mostrarle al usuario cual era el camino mas corto para salir del laberinto, también debe tener la posibilidad de generar laberintos de forma aleatoria , y contar con una opción para que el usuario genere sus propios laberintos.

**2.Diseño del TAD:**

|  |
| --- |
| TAD GRAFO matriz de adyacencia |
| matriz de adyacencia n\*n |
| Inv: El tamaño de la matriz siempre es de n\*n espacios, siendo n el número de objetos (nodos). La matriz se inicializa en 0 para todas sus posiciones, un número diferente de 0 significa que existe una arista, es decir hay una o muchas conexiones entre dos nodos. |
| Operaciones primitivas:   * Crear grafo: número de nodos  Grafo vacío. * Añadir arista: Grafo, nodo 1, nodo 2  Grafo con una conexión más * Añadir nodo: Grafo, nodo Grafo con un nodo más. * Borrar nodo: Grafo, nodo  Grafo con un nodo menos, siempre y cuando el nodo exista. * Borrar arista: Grafo, Nodo1, Nodo2  Grafo con una conexión menos, siempre y cuando está conexión exista. * Esta vacío: Grafo  booleano. |

**3.Requerimientos**

|  |  |
| --- | --- |
| Nombre | R1. Buscar camino mas corto |
| Resumen | encuentra el camino más corto de todos los posibles caminos en el grafo |
| Entradas | |
| Nodo inicial  Nodo final | |
| Resultados | |
| Camino mas corto encontrado | |

|  |  |
| --- | --- |
| Nombre | R2. Buscar camino mas largo |
| Resumen | Encuentra la ruta mas larga y menos conveniente en el grafo |
| Entradas | |
| Nodo inicial  Nodo final | |
| Resultados | |
| Camino mas largo encontrado | |

|  |  |
| --- | --- |
| Nombre | R3. asignar nodo inicial |
| Resumen | Establece el nodo inicial |
| Entradas | |
| Nodo seleccionado | |
| Resultados | |
| Nodo inicial asignado | |

|  |  |
| --- | --- |
| Nombre | R4. asignar nodo final |
| Resumen | Establece el nodo final |
| Entradas | |
| Nodo seleccionado | |
| Resultados | |
| Nodo final asignado | |

|  |  |
| --- | --- |
| Nombre | R5. crear grafo |
| Resumen | Crea un grafo con componentes especificados |
| Entradas | |
| Nodos generados | |
| Resultados | |
| Grafo creado | |
| Nombre | R6. agregar nodo |
| Resumen | Agrega un nuevo nodo al grafo |
| Entradas | |
| nodo | |
| Resultados | |
| Nodo agregado | |

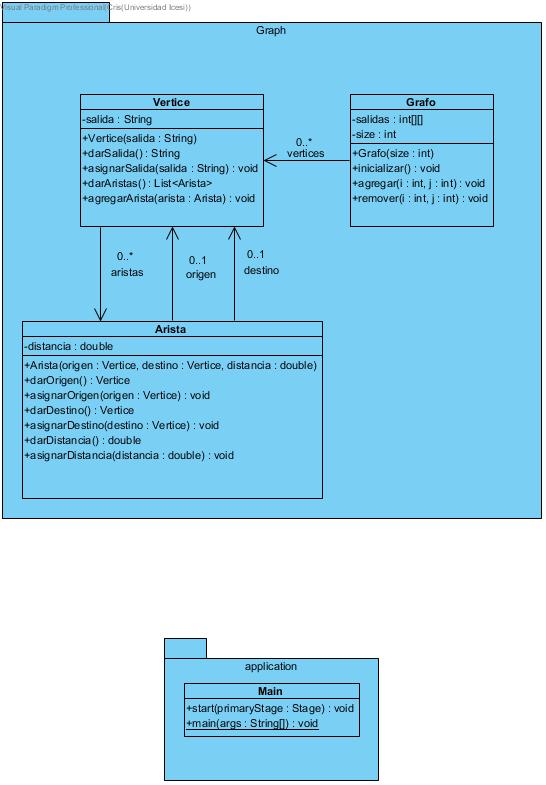
**3.Casos de prueba**

|  |  |  |  |
| --- | --- | --- | --- |
| clase : Grafo | | método : inicializar | |
|
| caso # | Descripción | estado inicial | resultado |
| 1 | Se crea un grafo auxiliar con los mismos valores que tiene el grafo y se compara si hubo cambios durante el uso del método | Se tiene un grafo vacío de tamaño 5 | El árbol no tiene ningún cambio pues al comparar un grafo vacío con uno inicializado, ambos de mismo tamaño, estos, son iguales |
| 2 | Se crea un grafo auxiliar con los mismos valores que tiene el grafo y se compara si hubo cambios durante el uso del método | Se tiene un grafo de tamaño 5  Con las siguientes conexiones  (1, 2)  (3, 2)  (2, 0) | El árbol tiene cambios, pues un árbol inicializado es diferente a uno con conexiones |

|  |  |  |  |
| --- | --- | --- | --- |
| clase : Grafo | | método :agregar | |
|
| caso # | Descripción | estado inicial | resultado |
| 1 | Se agregan al grafo las conexiones  (0, 0)  (1, 2)  Y a compararlo con un grafo inicializado para saber si hubo cambios | Se tiene un grafo vacío de tamaño 5 | El árbol tiene cambios, pues un árbol inicializado es diferente a uno con conexiones |
| 2 | Se agregan al grafo la conexión  (1, 2)  Se ve si el objeto entre el nodo 1 y 2 tiene dos conexiones, pues inicialmente este ya tenía una conexión entre esos dos nodos | Se tiene un grafo de tamaño 5  Con las siguientes conexiones  (1, 2)  (3, 2)  (2, 0) | Entre el nodo 1 y 2 ahora hay dos conexiones |

|  |  |  |  |
| --- | --- | --- | --- |
| clase : Grafo | | método :remover | |
|
| caso # | Descripción | estado inicial | resultado |
| 1 | Se remueve al grafo una posible conexión  (1, 2) | Se tiene un grafo vacío de tamaño 5 | El grafo al estar vacío, no fue posible remover una conexión así que sigue estando vacío |
| 2 | Se remueve al grafo la conexión  (1, 2) | Se tiene un grafo de tamaño 5  Con las siguientes conexiones  (1, 2)  (3, 2)  (2, 0) | El grafo ya sólo tiene las siguientes conexiones  (3, 2)  (2, 0) |

**4.Diagrama de TAD**

****