Alejandro Barrera Lozano
Algoritmos y Estructuras de datos

Primera entrega proyecto final

Enunciado – Ejercicio UVa (558 – Wormholes)

En el año 2163, los agujeros de gusano fueron descubiertos. Un agujero de gusano es un túnel subespacial que conecta 2 sistemas estelares a través del espacio y el tiempo. Los agujeros de gusano tienen unas cuantas propiedades:

- Los agujeros de gusano solo tienen una dirección
- El tiempo que lleva viajar a través de un agujero de gusano es insignificante
- Un agujero de gusano tiene dos puntos finales cada uno situado en un sistema estelar
- Un sistema solar puede tener más de un agujero de gusano
- Por alguna razón desconocida, empezando desde nuestro sistema solar, siempre es posible terminar en cualquier otro sistema estelar siguiendo una secuencia de agujeros
- Entre cada par de sistemas estelares, hay al menos un agujero de gusano en cualquier dirección
- No hay agujeros de gusano que terminen en el mismo sistema estelar en el que empezaron

Todos los agujeros de gusano tienen una diferencia de tiempo constante entre sus dos puntos finales. Por ejemplo, un agujero de gusano específico puede hacer que una persona viaje 15 años en el futuro. Otro agujero de gusano puede hacer que la persona viaje 42 años en el pasado.

Una brillante física, viviendo en la tierra, quiere usar los agujeros de gusano para estudiar el Big Bang. Desde que el "Warp drive" no ha sido inventado aún, no es posible para ella viajar de un sistema estelar a otro directamente. Esto puede hacerse viajando a través de agujeros de gusano, por supuesto.

La científica quiere alcanzar un ciclo de agujeros de gusano en alguna parte del universo que la puedan hacer terminar en el pasado. A partir de viajar en este círculo repetidas veces, la científica podrá viajar hacia el pasado cuanto tiempo considere necesario para ver el inicio del universo y ver el Big Bang con sus propios ojos.

Requerimientos funcionales

1. Agregar sistemas solares además de la tierra que tengan como identificador único un número entero diferente de 0.
2. Agregar agujeros de gusano entre sistemas estelares diferentes, con un punto inicial, un punto final y una constante de tiempo de viaje.
3. Calcular si es posible viajar en el tiempo de manera indefinida dados:
    - La cantidad de sistemas estelares que hay

- La cantidad de agujeros de gusano con los siguientes datos:
  - Punto inicial
  - Punto final
  - Constante de tiempo

TAD Grafo simple		
		
{inv:		
Operaciones:		
crearGrafo		-> Grafo
agregarVertice	Grafo, Vertice	->Grafo
agregarArista	Grafo, Vertice, Vertice,boolean	->Grafo
agregarArista	Grafo, Vertice,Vertice,boolean, double	->Grafo
eliminarVertice	Grafo, Vertice	->Grafo
BFS	Grafo, Vertice	->Grafo
DFS	Grafo,Vertice	->Grafo
eliminarArista	Grafo,Vertice,Vertice	->Grafo
numeroVertices	Grafo	->int
numeroAristas	Grafo	->int
darPesoArista	Grafo,Vertice,Vertice	->double
modificarPesoArista	Grafo,Vertice,Vertice,double	->double
darVertices	Grafo	-> List<Vertice>
darVertice	Grafo	->

Alejandro Barrera Lozano
Algoritmos y Estructuras de datos

Grafo()
Crea un grafo vacío, sin vértices ni aristas
pre:
post: Grafo={V{},E{}}

agregarVertice(Grafo g, Vertice v)
Agrega un vértice a el grafo
pre: $v \notin g.V \land g \rightarrow$ estaInicializado
post: $v \in g.V$

agregarArista(Grafo g, Vértice v1, Vértice v2, boolean esDirigido)
Agrega una arista entre 2 vértices del grafo, si es dirigido entonces la relación solo queda de v1 a v2, caso contrario la relación es mutua v1 a v2 y v2 a v1. Se asume peso = 1.
pre: $v1 \in g.V \land v2 \in g.V$
post: $e1=\{v1,v2,1\} \land e2=\{v2,v1,1\} \in g.E$ si esDirigido = false $\lor$ $e = \{v1,v2,1\} \in g.E$ si esDirigido = true.

agregarArista(Grafo g, Vértice v1, Vértice v2,boolean esDirigido, double peso)
Agrega una arista entre 2 vértices del grafo, si es dirigido entonces la relación solo queda de v1 a v2, caso contrario la relación es mutua v1 a v2 y v2 a v1. Peso de la arista es igual a peso
pre: $v1 \in g.V \land v2 \in g.V$
post: $e1=\{v1,v2,peso\} \land e2=\{v2,v1,peso\} \in g.E$ si esDirigido = false $\lor$ $e = \{v1,v2,peso\} \in g.E$ si esDirigido = true.

eliminarVertice(Grafo g, Vertice v)
Elimina un vértice del grafo
pre: $v1 \in g.V$
post: $v1 \notin g.V$

BFS(Grafo g, Vertice v)
Realiza el algoritmo Breadth First Search en el grafo g a partir del vértice v
pre: $v \in g.V$
post: añade atributos vi.pred y vi.d para todo vértice conexo en el grafo g

DFS(Grafo g, Vertice v)
Realiza el algoritmo Depth First Search
pre: $v \in g.V$
post: añade atributos vi.pred y vi.d y vi.f para todo vértice conexo en el grafo g

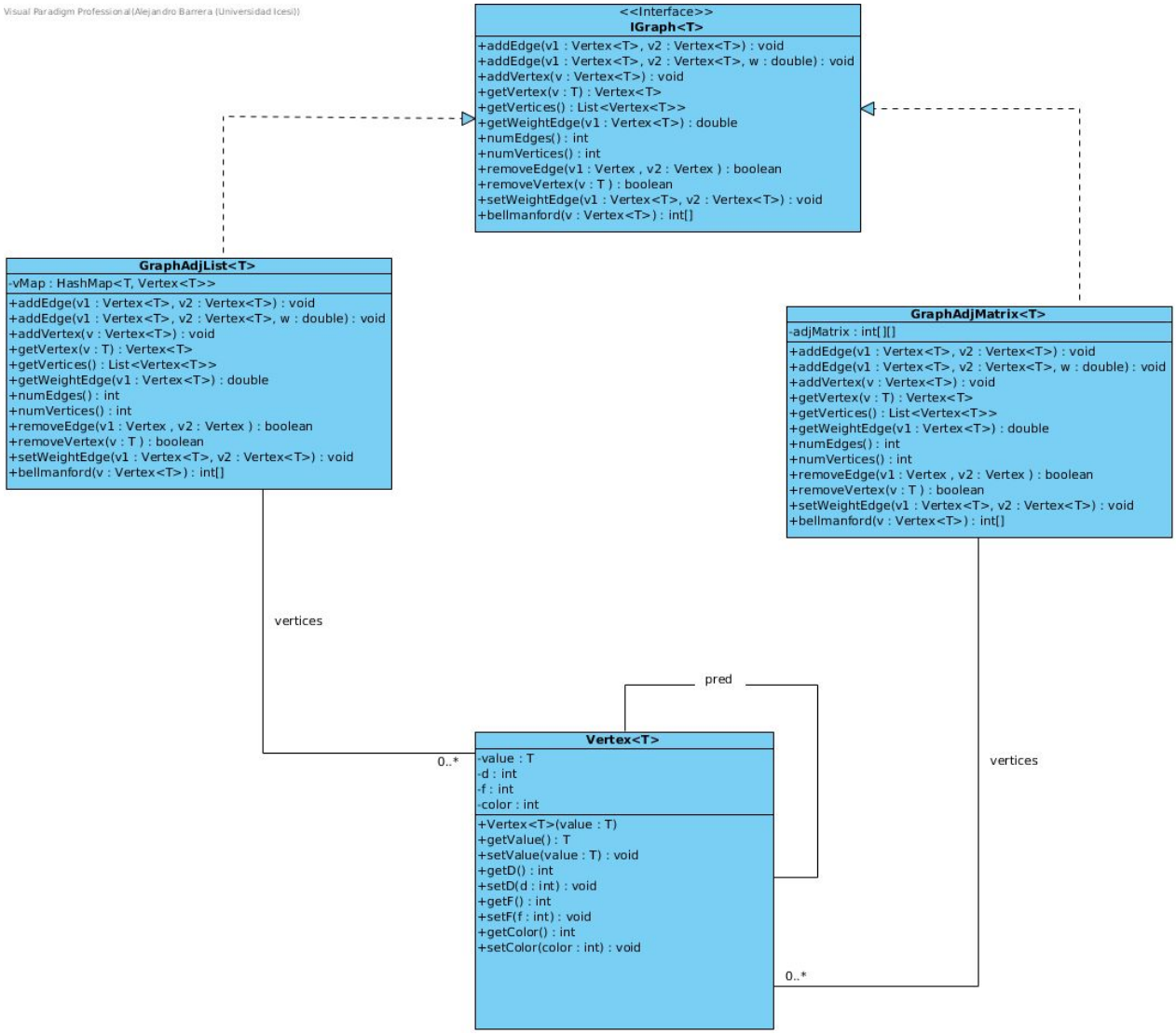
eliminarArista(Grafo g, Vertice v1, Vertice v2)
Elimina la arista e = {v1,v2}
pre: $e \in g.E$
post: $e \notin g.E$

numeroVertices(Grafo g)

Retorna el numero de vertices en el grafo pre: post:
numeroAristas(Grafo g) Retorna el número de aristas en el grafo pre: post:
darPesoArista(Grafo g, Vertice v1, Vertice v2) Retorna el peso de la arista que va desde v1 hasta v2 pre: v1,v2 ∈ g.V post:
modificarPesoArista(Grafo g, Vertice v1, Vertice v2, double peso) Modifica el peso de la arista que va desde v1 hasta v2 pre: v1,v2 ∈ g.V post: e = {v1,v2,peso}, e ∈ g.E
darVertices(Grafo g) Retorna la lista de vertices que tiene el grafo g pre: g != NIL post: List<Vertice> = {v1,v2,....,vn} ∈ g.V
darVertice(Grafo g, E element) Retorna un vertice del grafo g buscandolo por su identificador pre: g != NIL post: v ∈ g.V , v.value = element

Alejandro Barrera Lozano
Algoritmos y Estructuras de datos

Diagrama de clases



Diseño de pruebas unitarias

Prueba 1: Verifica si el metodo agregarVertice de la clase grafo funciona correctamente

Clase	Método	Escenario	Entrada	Salida
Grafo	+addVertex(Vertex):void	Grafo vacio	Vértice con valor 4	El grafo tiene un solo vértice con valor 4

Grafo	+addVertex(Vertex):void	Grafo con un conjunto de vértices V = { 10, 2, 5}	Vértice con valor 6	El grafo ahora tiene un conjunto de vértices Vx ={10,2,5,6}

Prueba 2: Verifica que el método agregarArista funcione correctamente

Grafo	+addVertex(Vertex,Vertex,boolean):void	Grafo con 2 vértices v1 = 1 v2 = 2 g.V = { v1, v2}	Arista entre v1 y v2 sin peso y no dirigida esDirigido = false	Hay una arista e1 = (1,2,1) e2 =(2,1,1) peso siendo igual a 1
Grafo	+addVertex(Vertex,Vertex,double,boolean):void	Grafo con 2 vértices v1 = 1 v2 = 2 g.V = {1,2}	Arista no dirigida entre v1 y v2 con peso w = 5 esDirigido = false	Hay una arista e1 =(1,2,5) e2 =(2,1,5)
Grafo	+addVertex(Vertex,Vertex,double,boolean): void	Grafo con 2 vértices v1 = 1 v2 = 2 g.V = {1,2}	Arista dirigida entre v1 y v2 con peso w = 5 esDirigido = true	Hay una arista e =(1,2,5)

Prueba 3: Verifica que el método eliminarVertice funcione correctamente

Grafo	+removeVertex(T):void	Grafo con 1 vértice v1 = 4	Eliminar vértice con valor = 4 t = 4	Grafo sin vértices
Grafo	+removeVertex(T):void	grafo no dirigido no ponderado con 2 vértices v1 = 4 v2 = 2 e1 = (4,2,1) e2 = (2,4,1)	Eliminar vértice con valor 4 T t = 4	Grafo que contiene únicamente v2 con ninguna arista asociada

Alejandro Barrera Lozano
Algoritmos y Estructuras de datos

Prueba 4

Grafo	+removeEdge(Vertex,Vertex):void	Grafo con 2 vértices y una arista que los relaciona v1 = 1 v2 = 4 e = (1,4,1)	Eliminar arista entre v1 y v2	Grafo con 2 vértices no conexos
Grafo	+removeEdge(Vertex,Vertex):void	Grafo con 3 vértices noDirigido = true v1 = 2 v2 = 4 v3 = 6 e1 = (2,4,1) e2=(4,2,1)	Eliminar arista entre v1 y v2	Grafo con 3 vértices no conexos
Grafo	+removeEdge(Vertex,Vertex):void	Grafo con 3 vértices noDirigido = false v1 = 2 v2 = 4 v3 = 6 e1 = (2,4,1) e2=(4,2,1)	Eliminar arista entre v1 y v2	Grafo con 3 vértices y una arista e1 = (4,2,1)

Prueba 5: Verificar que el recorrido BFS se esté realizando correctamente

Grafo	+bfs(Vértice):void	Grafo con 4 vértices 1,2,3,4 y 3 aristas e1 = (1,2,1) e2 =(2,3,1) e3 = (3,4,1)	Realizar recorrido bfs desde el vértice 3	Queda un árbol con raíz 3, qué recorrido en preorden se veria asi {3,2,4,1}G
Grafo	+bfs(Vértice):void	Grafo con 4 1,2,3,4 vértices y 3 aristas e1 = (1,2,1) e2 = (1,3,1) e3= (1,4,1)	Realizar recorrido bfs desde el vértice 3	Se recorre el grafo, la distancia entre 3 y 1 es 1, entre 3 y 2 es 2 y entre 3 y 4 es 2

Bibliografía:

UVa - Wormholes

https://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=499