Universidad Andrés Bello

Algoritmos y Estructuras de Datos
Laboratorio 2
Programación en C: TDAs no Lineales – Grafos

Profesores: Elizabeth Montero e Irene Zuccar

Ayudantes: Matías Vargas, Kevin Muena y Matías Poblete

Universidad Andrés Bello, Facultad de Ingeniería
Santiago-Viña del Mar, Chile.

FECHA DE ENTREGA: Domingo 12 de mayo, 23:55 hrs

1. Objetivo

El objetivo de este Laboratorio es:

- Diseñar e implementar un Tipo de Dato Abstracto no lineal que permita representar los principales componentes del problema enunciado a resolver.
- Diseñar e implementar las operaciones necesarias para la implementacón del algoritmo que resuelve el problema.
- Comprender y solucionar un problema clásico de las Ciencias de la Computación.

2. Enunciado

Encontrar el camino más corto desde un punto de inicio hasta un punto de destino dado un conjunto de puntos y las longitudes de ruta que los conectan es un problema ya conocido, e incluso es parte de nuestra vida cotidiana, ya que los programas de ruta más corta están ampliamente disponibles en la actualidad.

A la mayoría de las personas les gustan mucho estas aplicaciones porque les hacen la vida más fácil. Bien, tal vez no sea mucho más fácil ahora que casi todos pueden tener acceso a dispositivos de navegación GPS capaces de calcular el camino más corto, la mayoría de las rutas que forman la ruta más corta se vuelven más lentas debido al tráfico pesado.

Como la mayoría de las personas intentan seguir el mismo camino, no vale la pena seguir estas instrucciones. Con esto en mente, su jefe le pide que desarrolle una nueva aplicación

a la que solo él tendrá acceso, lo que le ahorrará tiempo cada vez que tenga una reunión o cualquier evento urgente. Tu jefe te pide que tu programa entregue como respuesta el camino casi más corto. Él define el camino casi más corto como el camino casi más corto que va desde un punto de partida hasta un destino tal que ninguna ruta entre dos puntos consecutivos pertenece a una ruta más corta desde el punto de partida hasta el destino. Por ejemplo, supongamos que la figura 1 representa el mapa dado, con círculos que muestran puntos de ubicación y líneas que representan rutas directas de un solo sentido con longitudes indicadas. El punto de inicio se marca como I y el punto de destino se marca como D. Las líneas en negrita pertenecen a un camino más corto (en este caso hay dos caminos más cortos, cada una con una longitud total de 4). Así, el camino casi más corto sería el indicado por líneas discontinuas (longitud total 5), ya que no incluye rutas directas pertenecientes a cualquiera de los dos caminos más cortos. Tenga en cuenta que podría existir más de una respuesta posible, por ejemplo, si la ruta con longitud 3 tuviese longitud 1. Es posible también que no exista solución.

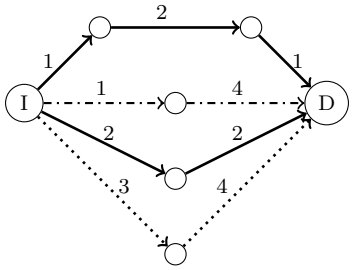

***Figura* 1:** Ejemplo

2.1. Entrada

La primera línea de un caso de prueba contiene dos enteros N($2 \leq N \leq 500$) y M ($1 \leq M \leq 10^4$), separados por un solo espacio, indicando respectivamente el número de puntos en el mapa y el número de rutas unidireccionales existentes que conectan dos puntos directamente. Cada punto se identifica mediante un número entero entre 0 y $N-1$. La segunda línea contiene dos enteros S y D, separados por un solo espacio, indicando respectivamente los puntos de inicio y destino ($S \neq D$; $0 \leq S, D \leq N$). Cada una de las siguientes líneas M contiene tres enteros U, V y P ($U \neq V$; $0 \leq U, V \leq N$; $1 \leq P \leq 10^3$), separados por espacios individuales, que indican la existencia de una ruta de ida desde U hasta V con la distancia P. Hay como máximo una ruta desde un punto dado U hasta un punto V, pero observe que la existencia de una ruta desde U a V no implica que haya una ruta de V a U y, si existe tal ruta, puede tener una diferente longitud. El final de la entrada se indica mediante una línea que contiene solo dos ceros separados por un solo espacio. La entrada debe leerse desde el archivo `casi.in`. La figura 3 muestra un ejemplo de entrada especificada para el problema que se muestra en la figura 2.

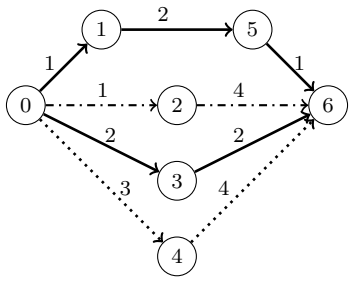

***Figura* 2:** Ejemplo con nodos numerados

```
7 9
0 6
0 1 1
0 2 1
0 3 2
0 4 3
1 5 2
2 6 4
3 6 2
4 6 4
5 6 1
0 0
```

***Figura* 3:** Ejemplo entrada

2.2. Salida

Para cada caso de prueba en la entrada, su programa debe imprimir una sola línea, que contiene -1 si no es posible coincidir con los requisitos, o un entero que represente la longitud del camino casi más corto encontrado seguido del camino. La salida debe mostrarse por salida estándar. La figura 4 muestra un ejemplo de la salida especificada.

```
elizabeth@cancun: $ ./a.out
5
0 2 6
```

***Figura* 4:** Ejemplo salida

3. Lo que se pide en este laboratorio

Se espera que usted construya un programa escrito en C, que resuelva el problema del camino casi más corto. Para esto, su programa debe leer un archivo de texto con los datos de la red y determinar el camino casi más corto o especificar que no existe.

Dentro de su programa usted debe diseñar e implementar un TDA para el manejo de los datos del algoritmo. Este TDA se debe basar en una implementación de listas dinámicas.

4. Acerca de la Entrega

4.1. Reglas

- Se realiza en equipos de dos personas.
- Se entrega a través de moodle a más tardar a las 23:59 h. del día especificado.
- Se debe entregar como un archivo comprimido que contenga ~~el manual de usuario en pdf~~ el archivo makefile y el código fuente de la implementación. Los entregables se detallan en la sección 4.2.
- El nombre del archivo debe ser "L1-{`Apellido1`}-{`Apellido2`}.{`extension`}". Donde `Apellido1` y `Apellido2` corresponden a los apellidos de cada uno de los integrantes del equipo y `extension` ∈ {~~`zip`~~, `rar`, `tar.gz`}.
- No se revisarán tareas atrasadas.

Además considere las siguientes reglas de evaluación:

- Ante la ausencia de alguno de los archivos requeridos, la nota del Laboratorio será un 1.0.
- Si el programa no se puede ejecutar, se evaluará el Laboratorio con nota máxima 4.0. Si el programa funciona, se evaluará su ejecución, y también el código fuente.
- ~~Si obtiene una nota menor a 4.0 en el programa, no se evaluará el manual de usuario. Si obtiene una nota igual o superior a 4.0, la nota de este laboratorio será 80 % de la nota en el programa y un 20 % de la nota obtenida en el manual de usuario.~~
- Si existe sospecha de copia (con otros compañeros, o desde internet) se evaluará el Laboratorio con nota mínima: 1.0.
- Las consultas las debe realizar directamente al ayudante, de lunes a viernes a los correos especificados o directamente en horario de laboratorio.

4.2. Entregables

La entrega considera el código fuente Y el archivo makefile correspondiente ~~un manual de usuario~~.

4.3. Código fuente

1. Debe implementar este trabajo en lenguaje C (no C++).
2. Debe incluir un *makefile* para compilar.
3. Debe construir una función para cada tarea que realice su código, cuidando los tipos de datos de las entradas y de las salidas.
4. Debe usar identificadores representativos para sus variables, parámetros de entrada y para sus funciones.
5. Debe comentar cada una de sus funciones, estructuras y tipos de datos que defina, indicando una descripción de la labor que lleva a cabo, sus entradas, y sus salidas.
6. Su código debe estar correctamente indentado (uso de sangrías para cada subbloque de instrucciones).
7. Puede trabajar con el IDE, entorno y compilador de C, que más le acomode. No obstante lo anterior, su código debe poder ser compilado y ejecutado sin problemas en Linux.
8. El sistema debe ser robusto, se penalizarán los errores no manejados, de cualquier tipo.

4.4. Manual de usuario

1. El manual de usuario debe incluir una detallada explicación de la forma en la que se debe ejecutar su programa.
2. Debe mostrar, al menos, un ejemplo de lo que se debe esperar como salida para una entrada determinada.
3. Debe indicar el comportamiento de su programa, ante los posibles escenarios que puedan ocurrir cada vez que el usuario digite algo, ya sea válido o inválido, con al menos un ejemplo.
4. Se considera dentro de la evaluación del manual de usuario, ortografía y redacción.

Anexos

Para resolver este problema es necesario tener conocimientos en teoría de grafos.

No se pedirá manual de usuario. Sin embargo, tu programa debe tener exactamente la siguiente interfaz, al momento de ser ejecutado:

Ingrese nombre de archivo: Aquí el usuario digitará el nombre del archivo de texto (con punto y extensión)
Longitud del camino casi más corto: Aquí tu programa debe mostrar el costo total del camino que encontró.
El camino casi más corto es: Aquí tu programa debe imprimir el camino casi más corto que encontró.

Algoritmo Dijkstra para cálculo de ruta más corta modificado

1. Creación de matriz de distancias:

	0	1	2	3	4	5	6
0	0	1	1	2	3	0	0
1	0	0	0	0	0	2	0
2	0	0	0	0	0	0	4
3	0	0	0	0	0	0	2
4	0	0	0	0	0	0	4
5	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0

2. Creación de vector de distancias:

0	1	2	3	4	5	6
inf	inf	inf	inf	inf	inf	inf

3. Creación de vector de nodos procesados:

0	1	2	3	4	5	6
false	false	false	false	false	false	false
inf	inf	inf	inf	inf	inf	inf

4. Creación de lista de nodos de acceso:

0	1	2	3	4	5	6
false	false	false	false	false	false	false
inf	inf	inf	inf	inf	inf	inf
-	-	-	-	-	-	-

5. Comenzando con el nodo inicio, se analizan las distancias a los nodos directamente conectados. Si la distancia es menor a la registrada, se modifica. Se marca como procesado y se anota el (los) nodo (s) desde el (los) cual (es) se accede.

   Para el nodo 0 se ha encontrado una distancia = 0 pasando por el nodo 0:

0	1	2	3	4	5	6
true	false	false	false	false	false	false
0	inf	inf	inf	inf	inf	inf
0	-	-	-	-	-	-

   Para el nodo 1 se ha encontrado una distancia = 1 pasando por el nodo 0. No hay otra forma de acceder al nodo 1.

0	1	2	3	4	5	6
true	true	false	false	false	false	false
0	1	inf	inf	inf	inf	inf
0	0	-	-	-	-	-

Para los nodos 2, 3 y 4 se realiza el mismo análisis y se obtiene el mismo tipo de resultado:

0	1	2	3	4	5	6
true	true	true	true	true	false	false
0	1	1	2	3	inf	inf
0	0	0	0	0	-	-

Para el nodo 5 se ha encontrado una distancia = 3 correspondiente a la suma del costo para llegar a 1 más el costo para llegar desde 1 a 5. Esta es la única forma de acceder al nodo 5.

0	1	2	3	4	5	6
true	true	true	true	true	true	false
0	1	1	2	3	3	0
0	0	0	0	0	1	-

Para el nodo 6 se ha encontrado una distancia = 5 correspondiente a la suma del costo para llegar a 2 más el costo para llegar desde 2 a 6.

0	1	2	3	4	5	6
true	true	true	true	true	true	false
0	1	1	2	3	3	5
0	0	0	0	0	1	2

Para el nodo 6 se ha encontrado una distancia = 4 (menor que la registrada) correspondiente a la suma del costo para llegar a 3 más el costo para llegar desde 3 a 6.

0	1	2	3	4	5	6
true	true	true	true	true	true	false
0	1	1	2	3	3	4
0	0	0	0	0	1	3

Para el nodo 6 se ha encontrado una distancia = 7 (mayor que la registrada) correspondiente a la suma del costo para llegar a 4 más el costo para llegar desde 4 a 6.

0	1	2	3	4	5	6
true	true	true	true	true	true	false
0	1	1	2	3	3	4
0	0	0	0	0	1	3

Para el nodo 6 se ha encontrado una distancia = 4 (igual que la registrada) correspondiente a la suma del costo para llegar a 5 más el costo para llegar desde 5 a 6.

0	1	2	3	4	5	6
true	true	true	true	true	true	false
0	1	1	2	3	3	4
0	0	0	0	0	1	3,5

El proceso termina cuando todos los nodos se han marcado como procesados. Para reconstruir los caminos de costo mínimo se debe considerar la información de nodos de acceso. Así, para llegar al nodo 6, el camino de costo mínimo tiene una distancia de cuatro y se obtiene pasando por los nodos 3 y 5. Al nodo 3 se accede directamente desde el nodo 0. Al nodo 5 se accede pasando por el nodo 1.

0	1	2	3	4	5	6
true	true	true	true	true	true	true
0	1	1	2	3	3	4
0	0	0	0	0	1	3,5