Algoritmos y Estructuras de Datos II – Laboratorio 6.

En este laboratorio se deberá implementar el algoritmo de Dijkstra, que computa el costo de un camino de costo mínimo desde un vértice inicial hacia todos los demás vértices de un grafo.

La función a completar se encuentra en el archivo *dijkstra.c:*

```
cost_t *dijkstra(graph_t graph, vertex_t init)
```

El primer parámetro es un grafo, y el segundo es el vértice inicial. Devuelve un arreglo que en cada posición *v* contiene el costo del camino de costo mínimo desde *init* hasta *v.*

Para la implementación se recomienda guiarse por el pseudocódigo dado en la clases teóricas. Se provee una implementación del TAD *graph*, que representa a un grafo como una matriz de costos.

En la carpeta *input* se encuentran archivos en formato texto que especifican una matriz de costos. Por ejemplo, el archivo *input/example_graph_1.in* contiene la siguiente matriz:

```
6
0  4  1 10  #  #
4  0  3 #   1  #
# 2  0 8   4  #
2  #  2 0   #  #
# #  3 #   0  3
# #  2 1   #  0
```

Esta matriz representa el grafo dado como ejemplo en el video de la clase teórica sobre Dijkstra. El primer número es la cantidad de filas y columnas. Los vértices del grafo son {0,1,2,3,4,5}, el costo de la arista (2, 3) es 8. Cuando no existe la arista, se escribe # para representar costo infinito.

Se provee también el TAD *set*, un conjunto finito, que será necesario para implementar el algoritmo. La función *main* intenta leer un grafo desde un archivo, aplica el algoritmo de Dijkstra y finalmente imprime en pantalla el arreglo de costos mínimos devuelto por el algoritmo.

Makefile

Escribir un archivo Makefile para agilizar la compilación del proyecto.

Usar las referencias del punto estrella del lab05 y la siguiente:
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html#using.

Definir variables de compilación (como **CC**, **CFLAGS**, etc.), un target inicial para compilar todos los archivos necesarios (convencionalmente llamado **all)** y un target para eliminar los archivos generados en el proceso de compilación (convencionalmente llamado **clean).**

De esta manera al ejecutar **make,** equivalente a ejecutar **make all,** debería generar el archivo ejecutable

y al ejecutar **make clean** se deberían eliminar aquellos archivos generados en el proceso de compilación.

Punto estrella:

Resolver con este algoritmo el ejercicio 3 del práctico 3.2. Para ello se debe crear una función *main* que solicite al usuario la cantidad máxima de litros de nafta disponibles, las ciudades a visitar y los costos de cada una. Imprimir en pantalla las ciudades que pueden visitarse con la cantidad de nafta indicada.

Fecha de entrega: 8 de Junio.