Seis grados de Kevin Bacon

# Introducción

De acuerdo al juego de [Seis Grados de Kevin Bacon](https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon), cualquiera que esté en la industria del Cine de Hollywood puede conectarse a Kevin Bacon en un máximo de seis pasos, cada paso consiste en encontrar una película en la cual dos actores o actrices hayan actuado.

En este problema nos interesa encontrar el camino más corto entre cualquier par de actores o actrices eligiendo una secuencia de películas que los conecte. Por ejemplo, el camino más corto entre Jennifer Lawrence y Tom Hanks es 2: Jennifer Lawrence está conectada con Kevin Bacon porque ambos estuvieron en la película “X-Men: First Class,” y Kevin Bacon está conectado a Tom Hanks porque ambos estuvieron en la película “Apollo 13.”

Este problema puede ser modelado mediante un grafo no dirigido, en el cual los nodos son las personas y los arcos conectarían a dos personas que hayan actuado en una misma película. La idea es encontrar entonces un camino entre dos personas a quienes queremos conectar.

# Práctica

## Distribución del proyecto

Se te provee un paquete de proyecto con una estructura básica, algunos códigos ya resueltos y otros que debes resolver.

El proyecto está estructurado de la siguiente manera:

**lib/include/IOUtils.hpp y lib/src/IOUtils.cpp**: Contienen la implementación de operadores para imprimir diversas estructuras de datos que manejaremos en el proyecto, también encontrarás una macro LOG que sirve para imprimir en pantalla cualquier mensaje con el que se quiera hacer seguimiento del comportamiento. Estos mensajes solo se verán si al compilar se hace con la bandera -DDEBUG. La forma general de usarla es similar a printf, es decir, el primer parámetro es una cadena formateada y luego vendrían las variables requeridas. Estudia estos archivos, puedes agregar cualquier cosa que requieras pero no modifiques lo existente.

**lib/include/TestUtils.hpp y lib/src/TestUtils.cpp**: Contienen algunas operaciones básicas para hacer pruebas unitarias de nuestras soluciones. Estudialos, puedes agregar cualquier cosa que requieras pero no modifiques lo existente.

**lib/include/CSVStream.hpp y lib/src/CSVStream.cpp**: Contienen la implementación de la clase CSVIStream, una utilidad que permite abrir un archivo en formato [csv](https://en.wikipedia.org/wiki/Comma-separated_values) y leer fila por fila almacenando los campos en un arreglo de cadenas.

**lib/include/Degrees.hpp y lib/src/Degrees.cpp:** Contienen la implementación de la clase Degrees, esta clase es la que debes resolver.

**main/src/CSVStreamDemo.cpp:** Es un programa que ejemplifica cómo utilizar la clase CSVIStream para obtener y manejar los datos desde un archivo en formato csv. Estúdialo y entiéndelo bien, te puede servir para resolver el constructor de Degrees.

**main/src/Degrees.cpp:** Es un programa que recibe el nombre de una base de datos, pide los nombres de dos actores o actrices y ejecuta la búsqueda del camino mínimo entre estos e imprime un reporte. Ejemplo para ejecutar **./main/bin/Degress small**.

**test/src/Degrees.cpp:** Es un programa que ejecuta un caso de prueba de la clase Degrees utilizando el directorio **resources/small**.

**Makefile**: Contiene reglas de compilación para todo el proyecto: make main compilara todos los programas en main/src y generar los ejecutables en main/bin. make test hace algo similar con todo lo que está en test/src y make all compila todos los main y todos los test. Lee los comentarios sobre el uso de la biblioteca DeSiGNAR si aplica el caso.

## Modelo de Datos

### Base de datos

Se te proveen dos bases de datos para que hagas tus pruebas, las puedes encontrar en el directorio **resources**. La que está dentro del directorio **small** contiene pocos datos, así que con esto podrás hacer algunas pruebas rápidas.

Cada base de datos contiene tres archivos:

* **people.csv**: Contiene los registros de actores y actrices, cada línea tiene el id de la persona, el nombre y el año de nacimiento de la persona.
* **movies.csv**: Contiene los registros de películas, cada lína contiene el id de la película, el nombre y el año de lanzamiento.
* **stars.csv**: Contiene los registros de actuaciones. Cada línea contiene un id de persona y un id de película, esto refiere a que la persona con el id una línea actuó en la película con el id dado en la misma línea.

### Grafo

El modelo de base de datos debe ser convertido en un grafo no dirigido en el cual los nodos almacenarán los nombres de las personas y los arcos deberán conectar a dos personas que hayan actuado en una misma película, además, el arco deberá contener el nombre de la película.

El cómo implementes este el grafo queda de ti. Puedes elegir matrices o listas de adyacencia. Depende de cómo crees que sea más fácil resolver este problema.

## Tu trabajo

Tu trabajo consiste en resolver la implementación de la clase Degrees. Como podrás notar, se te provee esta clase sin atributos, tiene un constructor, algunos métodos vacíos y uno de clase ya resuelto.

**Degrees**: Este constructor debes resolverlo. Recibe como parámetros un stream de personas (actores o actrices), un stream de películas y un stream de actuaciones. Aquí deberías recorrer los streams para construir un grafo. Te sugiero que tengas el grafo como atributo de la clase.

**get\_nodes\_contents**: Este método debes resolverlo. Retorna una lista de los nombres de las personas que están contenidas en los nodos del grafo.

**make\_arc\_contents**: Esto es un método de clase que se te provee resuelto, recibe como parámetros los nombres de dos personas que deberían estar conectadas mediante un arco y el nombre de la película en la que las dos personas actúan y retorna un par que contiene un par de nombres de personas conectadas y la película en la que actúan. Un ejemplo de retorno es el siguiente: **((“Daniel Radcliffe”, “Emma Watson”), “Harry Potter and the Philosopher's Stone”)**.

**get\_arcs\_contents**: Este método debes resolverlo. Retorna una lista de pares, donde cada par contiene el par de personas que conecta y la película en la que actúan. Te sugiero que utilices el método anterior para resolver este.

**get\_degree\_path**: Este método debes resolverlo. Recibe como parámetros los nombres de dos personas denominados **source\_name** y **target\_name**. Debe retornar una lista de pares de cadenas. Cada par consiste en el nombre de una película y el nombre de una persona. El primer elemento de la lista no contiene source\_name sino que contiene el nombre de una persona conectada a source\_name y la película que los conecta y así sucesivamente para todos los demás elementos. Por ejemplo: si el llamado fuese **get\_degree\_path(“Tom Hanks”, “Chris Sarandon”)**, la lista resultante sería algo como esto: **[(“Forrest Gump”, “Robin Wright”), (“The Princess Bride”, “Chris Sarandon”)]**. Si **source\_name** y **target\_name** no están entre los nodos del grafo o son iguales, entonces se deberá retornar una lista vacía.

# Evaluación

Esto lo puedes resolver en pareja (elige bien a quien te acompaña).

Tienes todo el semestre para entregar este proyecto y podrás hacer un envío una vez por semana. Si envías y fallas, entonces podrás enviar de nuevo la siguiente semana y así sucesivamente hasta terminar el semestre.

Para poder evaluarte, debes enviar los archivos Degrees.hpp y Degrees.cpp, al correo [alejandro.j.mujic4@gmail.com](mailto:alejandro.j.mujic4@gmail.com) con el subject: [AYDA]-04-Degrees. El correo deberá tener los nombres y números de identificación de la pareja participante.