**Algoritmos y Estructuras de Datos III**
Primer Cuatrimestre 2019
**Trabajo Práctico 2**

*Departamento de Computación*
*Facultad de Ciencias Exactas y Naturales*
*Universidad de Buenos Aires*

**Modelando problemas con grafos**

---

# Enunciado

El objetivo del trabajo práctico es resolver los problemas propuestos de diferentes maneras, realizando posteriormente una comparación entre los diferentes algoritmos utilizados.

Se debe:

1. Describir los problemas a resolver dando ejemplos de los mismos y sus soluciones.

Luego, por cada método de resolución:

2. Explicar de forma clara, sencilla, estructurada y concisa, las ideas desarrolladas para la resolución del problema. Para esto se pide utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas (**¡sin usar código fuente!**). Se debe también justificar por qué el procedimiento desarrollado resuelve efectivamente el problema. **En caso de que el algoritmo resuelva de manera exacta el problema.**
3. Deducir una cota de complejidad temporal del algoritmo propuesto (en función de los parámetros que se consideren correctos) y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada.
4. Dar un código fuente claro que implemente la solución propuesta.

   El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (comentarios pertinentes, nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.).

Por último:

5. Realizar una experimentación computacional para medir la performance de los programas implementados, comparando el desempeño entre ellos. Para ello se debe preparar un conjunto de casos de test que permitan observar los tiempos de ejecución en función de los parámetros de entrada, analizando la idoneidad de cada uno de los métodos programados para diferentes tipos de instancias.

Para cada problema se listan los algoritmos que se deben considerar.

**Solo se permite utilizar $c++$ como lenguaje para resolver los problemas.** Se pueden utilizar otros lenguajes para presentar resultados.

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados.

# Problemas

### Segmentation is my fault

En Procesamiento de Imágenes uno de los problemas clásicos es el de *segmentación*. Segmentar una imagen significa dividir los píxeles en conjuntos disjuntos que guarden cierta coherencia entre sí. Por ejemplo, la segmentación de una imagen debería poder identificar cada objeto particular que se encuentra en la misma y diferenciarlo del resto, o debería poder separar algo que se encuentra al frente de la imagen contra algo que se encuentra en el fondo.


Figura 1: Imagen segmentada

En la figura 1 vemos un ejemplo de una imagen segmentada. A la izquierda se puede ver la imagen original y a la derecha el resultado de la segmentación. En este caso se presenta una segmentación de una calidad muy alta ya que se es capaz de segmentar objetos pequeños con una muy buena precisión.

El problema de segmentación de imágenes no es nada simple. A lo largo de los años se han desarrollado métodos con diversas técnicas para atacar este problema. En nuestro caso, buscaremos abordar este problema de una perspectiva de grafos, utilizando la noción de Árbol Generador Mínimo para generar los conjuntos de píxeles que represetan cada sector de la imagen.

### Parámetros y formato de entrada/salida

La entrada consistirá de una primera línea con dos enteros $w$ y $h$, que representan el ancho y el alto de la imagen a procesar. Luego les sucederán $h$ líneas, cada una con $w$ números enteros entre 0 y 255 que representan la intensidad de los píxeles de la imagen, considerándola como una imagen en escala de grises.

La salida consistirá de $h$ líneas. Cada linea deberá contener $w$ enteros. El entero en la posición $ij$ de la salida se corresponderá con el pixel de la imagen en dicha posición. Dicho entera representará el conjunto al que pertenece el pixel.

### Metodología

Para resolver el problema de segmentación se deberá utilizar las nociones presentadas en [1].

En la sección 3 de dicho trabajo se presentan los conceptos necesarios utilizados posteriormente en el algoritmo propuesto. En particular se definen las métricas que se utilizarán para identificar la similaridad o disimilaridad entre regiones de píxeles. Luego, en la sección 4 se presenta el algoritmo propiamente dicho junto con el enunciamiento de varias propiedades que este cumple, y sus demostraciones.

Una vez presentado el algoritmo en su forma general, se deben especificar ciertos aspectos específicos según la representación que se decida utilizar para pasar de la imagen original a un grafo utilizable por el algoritmo. En nuestro trabajo debemos utilizar las nociones presentadas en la sección 5 para realizar nuestra implementación y posterior experimentación.

Para la solución del problema, se deben considerar los siguientes algoritmos en sus resoluciones intermedias:

- Estructura de Disjoint Set con arreglo de representación.
- Estructura de Disjoint Set con árbol de representación.
- Estructura de Disjoint Set con árbol de representación y *Path Compression*.

Para la experimentación, se recomienda utilizar imágenes conocidas como las que se pueden encontrar en

- `http://sipi.usc.edu/database/database.php?volume=misc`
- `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/`

Sumado a la experimentación cuantativa de los algoritmos ya mencionada, en este problema se debe considerar una experimentación cualitativa que se centre en estudiar la calidad de las soluciones entregadas al variar los diferentes aspectos de la solución.

## Llenalo con super

Luego de finalizar nuestro software de segmentación de imágenes, decidimos salir a venderlo por toda la Argentina. Nuestro negocio recién está arrancando así que no tenemos mejores opciones que hacer nosotros mismos las ventas de manera personal, moviendonos entre ciudad y ciudad donde queremos ubicar nuestro software. Como la nafta está muy cara, y eso impacta fuertemente en nuestro incipiente negocio, queremos saber la mejor forma de movernos entre dos ciudades gastando la menor cantidad de plata posible en combustible. En nuestro mapa tenemos $n$ ciudades y $m$ rutas bidireccionales que conectan pares de ciudades. Cada ruta tiene un valor $l_i$ que indica la cantidad de litros de nafta que necesita nuestro vehículo para recorrerla. A su vez, cada ciudad tiene un costo asociado $c_i$ que indica el precio de un litro de nafta en dicha ciudad. Si sabemos que nuestro auto tiene un tanque de nafta con capacidad máxima para 60 litros, queremos saber cuál es el costo mínimo para movernos entre un par de ciudades dadas.

### Parámetros y formato de entrada/salida

La entrada consistirá de una primera línea con dos enteros $n$ y $m$, correspondientes a la cantidad de ciudades y a la cantidad de rutas, respectivamente. A continuación, habrá $n$ líneas. Cada línea tendra un entero $c_i$, representando el costo del litro de combustible en la ciudad $i$. Luego le sucederán $m$ líneas con tres enteros, $a_i$, $b_i$ y $l_i$. $a_i$ y $b_i$ indicará las ciudades conectadas por la ruta, mientras que $l_i$ indicará la cantidad de litros necesarios para recorrerla. Asumiremos que todos los valores en la entrada serán enteros positivos.

La salida consistirá de $n*(n-1)$ líneas. Cada línea tendrá tres enteros $a_i$, $b_i$, $s_i$. $a_i$ y $b_i$ deben indicar las ciudades consideradas. $s_i$ indicará el costo mínimo necesario para ir de una ciudad a la otra. Las líneas en la salida deberan estar ordenadas según el ordén lexicográfico de las ciudades pertinentes.

**Algoritmos**

Para la solución del problema, se deben considerar los siguientes algoritmos en sus resoluciones intermedias:

- Algoritmo de Dijkstra.
- Algoritmo de Dijkstra con *priority queue*.
- Algoritmo de Bellman-Ford.
- Algoritmo de Floyd-Warshall.

**Fechas de entrega**

- *Formato Electrónico:* Jueves 30 de Mayo de 2019, hasta las **23:59 hs**, enviando el trabajo (informe + código) a la dirección `algo3.dc@gmail.com`. El subject del email debe comenzar con el texto `[TP2]` seguido de la lista de apellidos de los alumnos.
- *Formato físico:* Viernes 31 de Mayo de 2019, a las 18 hs.

**Importante:** El horario es estricto. No se considerarán los correos recibidos después de la hora indicada.

# Referencias

[1] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004.