1 Problema

En esta tarea diseñarás algoritmos para encontrar el menor número de colores para colorear una grafo. Se te darán un conjunto de grafos y tu tarea es etiquetar cada nodo con el menor número de colores como te sea posible, de tal manera que todos los pares de nodos unidos por un arco no tengan el mismo color. La siguientes imágenes muestran un ejemplo:

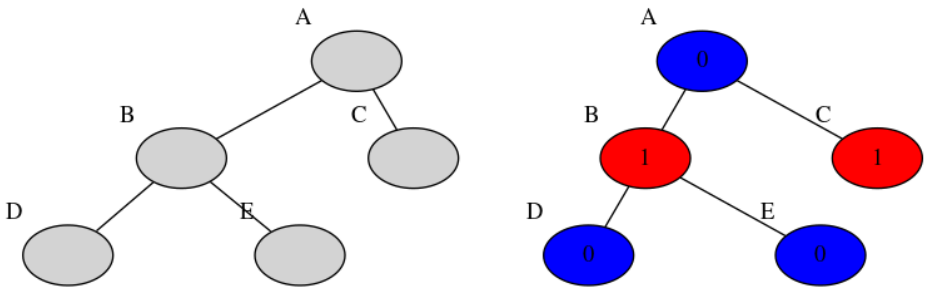

Figura 1
Ejemplo de grafo coloreado. El grafo est conformado por los nodos A,B,C,D,E. Del lado derecho, los nodo han sido etiquetados con su respectivo colo (0=Azul, 1=Rojo).

2 Formalización

Dada un grafo $G = (N, A)$ con nodos $N - 0, \ldots, n-1$ y arcos A, sea $c_i \in \mathcal{N}$ una variable que indica el color del nodo i. El problema de colorear un grafo es:

$$\begin{aligned} \textbf{minimize} \quad & \max_{i \in 0,\ldots,n-1} c_i \\ \textbf{s.t.} \quad & c_i \neq c_j \quad (i,j) \in A \end{aligned} \tag{1}$$

3 Formatos de datos

3.1 Datos de entrada

Cada problema está especificado en un archivo. Los archivos contienen $|A| + 1$ líneas. La primera línea contiene dos números: $|N|$, el número de nodos, y $|A|$ el número de arcos.
El resto de las líneas representan un arco (u_i, v_j) donde $u_i, v_j \in 0, \ldots, |N| - 1$ objetos.

3.2 Datos de salida

Tu programa deberá de generar un archivo de salida con la solución. El archivo tendrá dos líneas. La primera línea tendrá el valor solución del problema (i.e. el número de colores usados en el coloreado del grafo, i.e. el valor de la función objetivo). La segunda línea tendrá una lista de $n \in N$ números con valor c_i para cada una de las nodos.

3.3 Ejemplo

Datos de entrada (archivo `data/gc_4_1`)

```
4 3
0 1
1 2
```

```
1 3
```

Datos de salida

```
3
1 2 1 1
```

4 Instrucciones

Acepta la invitación de **Github Classroom**, el repositorio tiene un nombre con el prefijo: `or-2020-graph-coloring-problems`.
Copia los archivos dentro de la carpeta `tareas/Tarea_2` a tu repositorio de la tarea.
Utiliza `solver.py` y modifícalo agregando tus algoritmos de optimización.
Debes de poder ejecutarlo con

```
python ./solver.py ./data/<archivo>  <algorithm>
```

donde `<algorithm>` es el nombre de la función que implementa tu algoritmo.
Crea un archivo de texto llamado `soluciones.txt` que contenga por cada archivo de datos una línea como la mostrada:

```
./data/gc_4_1  <algorithm-1>
./data/gc_20_1  <algorithm-2>
... # etc
```

Si sólo tienes un algoritmo para solucionar, todas las líneas tendrán el mismo último argumento.
NOTA: Este archivo será el utilizado para evaluar tu tarea
NOTA: Tu algoritmo debe de terminar en menos de 5 horas.
NOTA: Si decides usar otro lenguaje de programación agrega un archivo `bash` con la línea de ejecución, deberás de proveer instrucciones sobre el ambiente necesario para ejecutar tu solución y deberás respetar el formato de entrada y salida.

4.1 Recursos

El archivo `solver.py` está en la carpeta `Tarea_2`, junto con el directorio `data` y este archivo.
La especificación de los problemas del *knapsack* se encuentran en el directorio `data`.

5 Calificación

- Soluciones *infeasible* recibirán 0 puntos.
- Soluciones que no concluyan en el tiempo límite, recibirán 0 puntos
- Soluciones *feasible* recibirán por lo menos 3 puntos.
- Soluciones *feasible* que alcancen un mínimo de calidad recibirán 7 puntos
- Soluciones *feasible* que superen la barrera de alta calidad recibirán 10 puntos

6 Políticas de colaboración

Creo que los problemas de la vida real se resuelven colaborando, intercambiando ideas y cotejando soluciones, por lo tanto, los incito a realizar estas actividades. Por otro lado, la tarea es *individual*, entonces, **NO** hagan:

1. Usar código que no es suyo y que no entienden
2. Pasar el código a otros compañeros o *postearlo* en algún foro.
3. Pasar *pseudocódigo* (es lo mismo que lo anterior)

7 Advertencias

1. No modifiquen los archivos de la carpeta `data`.
2. No cambien el nombre del archivo `solver.py`
3. No uses variables globales
4. No cambies los archivos de lugar, en tu repositorio `solver.py` y la carpeta `data` deben de estar en la raíz del repositorio.

8 Créditos

Los problemas de esta tarea fueron tomados de la clase *Discrete Optimization* de los profesores **Pascal Van Hentenryck** y **Carleton Coffrin**.