Trabajo práctico 3

Fecha de entrega: viernes 20 de noviembre, hasta las 20:00 horas.

Vamos a darle una mano a Aida. Al principio de cada cuatrimestre tiene que organizar la asignación de aulas según los horarios de cada materia y las características del aula que cada docente solicite como plazas disponibles, por ejemplo cañón, aire acondicionado, luz natural o cercanía con los baños, etc.

Aida ya tiene las aulas disponibles enumeradas con un color cada una. En cada inicio de cuatrimestre se para frente a su pizarra y dibuja un círculo para cada materia y las conecta con la materias que se superponen en horario. Luego cerca de cada materia pinta la lista de colores correspondiéndose con las aulas a las que podría asignarse esa materia.

Este problema se conoce formalmente como List Coloring. El problema consiste en asignarle a cada vértice de un grafo $G = (V, E)$ un color de una lista de colores disponibles (para dicho vértice), y pintar los vértices de manera que no haya dos vértices adyacentes del mismo color.

En el presente trabajo práctico se pide:

1. Diseñar e implementar un **algoritmo exacto** para el caso en el que cada materia tiene un máximo de 2 aulas posibles donde se podría dictar. Este problema, donde cada vértice tiene un máximo de 2 colores disponibles se conoce como 2-List Coloring y tiene solución en tiempo polinomial, por lo tanto, el algoritmo que ustedes implementen debe tener solución en tiempo polinomial.
2. Diseñar e implementar un **algoritmo exacto** para List Coloring y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado. Elaborar podas y estrategias que permitan mejorar los tiempos de ejecución. En los casos en los que el backtracking reduzca el problema a 2-List Coloring utilizar el algoritmo del item anterior como caso base.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo.
   c) Realizar una experimentación que permita observar los tiempos de ejecución del algoritmo en función del tamaño de entrada y de las podas y/o estrategias implementadas.
3. Diseñar e implementar una **heurística constructiva golosa** para List Coloring y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo.
   c) Describir instancias de List Coloring para las cuales la heurística no proporciona una solución óptima. Indicar qué tan mala puede ser la solución obtenida respecto de la solución factible.
   d) Realizar una experimentación que permita observar la performance del algoritmo en términos de tiempo de ejecución en función del tamaño de entrada.
4. Diseñar e implementar una **heurística de búsqueda local** para List Coloring y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado. Plantear al menos dos vecindades distintas para la búsqueda.
   b) Calcular el orden de complejidad temporal de peor caso de una iteración del algoritmo de búsqueda local (para las vecindades planteadas). Si es posible, dar una cota superior para la cantidad de iteraciones de la heurística.
   c) Realizar una experimentación que permita observar la performance del algoritmo comparando los tiempos de ejecución y la calidad de las soluciones obtenidas, en función de las vecindades utilizadas y elegir, si es posible, la configuración que mejores resultados provea para el grupo de instancias utilizado.

5. Una vez elegidos los mejores valores de configuración para cada heurística implementada (si fue posible), realizar una **experimentación sobre un conjunto nuevo de instancias** para observar la performance de los métodos comparando nuevamente la calidad de las soluciones obtenidas y los tiempos de ejecución en función del tamaño de entrada. Para los casos que sea posible, comparar también los resultados del algoritmo exacto implementado. Presentar todos los resultados obtenidos mediante gráficos adecuados y discutir al respecto de los mismos.

Condiciones de entrega y términos de aprobación

Este trabajo práctico consta de varias partes y para aprobar el trabajo se requiere aprobar todas las partes del mismo. La nota final del trabajo será un promedio ponderado de las notas finales de las partes y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por el trabajo original y un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega.

Respecto de las implementaciones, Java es el lenguaje sugerido por la Cátedra pero se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia (previa consulta con el docente). Debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación.

La entrada y salida debe hacerse por medio de archivos. No se considerará correcta una implementación que no pase los tests que se mencionaron en los puntos anteriores. 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. Deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `emilioOca@gmail.com` indicando *TP 3* y luego los apellidos de los integrantes del grupo.

Formato de entrada: La entrada comienza con una línea con tres valores enteros, n, m y c, separados por espacios (con $n > 0$, $m \geq 0$ y $c \geq 1$); los valores n y m indican la cantidad de vértices y aristas de G, respectivamente y el valor c representa la máxima cantidad de colores disponibles. Luego, siguen n líneas, representando los colores disponibles para cada vértice (en orden desde el vértice 0 hasta el vértice $n-1$). Cada una de estas líneas tiene el formato:

```
t c1 c2 ... ct
```

Donde `t` representa la cantidad de colores disponibles para el vértice ($1 \leq t \leq c$) y `c1`, ..., `ct` representan los colores disponibles para el vértice descripto en dicha línea. Todos los colores permitidos para un vértice serán enteros distintos entre 0 y $c-1$. A continuación, siguen m líneas, representando las aristas del grafo. Cada una de estas líneas tiene el formato:

```
u v
```

donde `u` y `v` son los extremos de la arista representada (numerados de 0 a $n-1$). Se puede suponer que los grafos son simples (i.e., sin bucles ni aristas repetidas).

Formato de salida:
Para los casos en que existe solución:
La salida debe contener una línea con el siguiente formato:

```
c1 c2 ... cn
```

donde `ci` es el color el cual se pinta en la solución al i-ésimo vértice.

Para los casos en que **no** existe solución:

```
X
```