UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE CIENCIAS
DEPARTAMENTO DE MATEMÁTICA Y CIENCIA DE LA COMPUTACIÓN
LICENCIATURA EN CIENCIA DE LA COMPUTACIÓN
Profesor(es): Nicolas Thériault
Primer Semestre de 2017

Complejidad de Algoritmos – Laboratorio 4

1. Objetivos

El objetivo de este laboratorio es de analizar y aplicar un algoritmo de programación dinámica.

El laboratorio se trabajará en grupos de 2 o 3 alumnos, entregando un resultado (informe y programas) por grupo.

2. Problema

Para encontrar un ciclo de longitud mínima en un grafo, se puede utilizar una búsqueda en anchura, probando con distintos puntos iniciales. El problema principal de este método es su complejidad, que puede variar entre exponencial (mejor caso) y factorial (peor caso).

De otro lado, los algoritmos de programación dinámica de Warshall y Floyd permiten obtener algoritmos en tiempo polinomial para algunas preguntas sobre la estructura de un grafo (conectividad y camino mínimo entre pares de vertices).

El objetivo de este laboratorio es de desarrollar un algoritmo de programación dinámica para encontrar un ciclo mínimo en un grafo.

3. Indicaciones

El problema a resolver en este laboratorio es abierto. Se podrán dar indicaciones y pistas, pero deberán encontrar la solución por trabajo propio.

Algunas indicaciones que les podrían ayudar:

- Identificar claramente lo que se quiere optimizar y como medirlo.
- Recordar que si se elimina una arista o un vértice (con las aristas adyacentes) de un ciclo, se obtiene un camino.
- Determinar como su algoritmo identificará el ciclo mínimo (si conocen el ciclo mínimo, ¿en qué momento de la inducción aparecerá el valor asociado a este ciclo?).
- Determinar como el algoritmo puede permitir extraer el ciclo mínimo (no solamente el largo mínimo).

4. Se solicita

1. Adaptar los algoritmos de Warshall y Floyd para encontrar un ciclo mínimo en un grafo.

2. Analizar la complejidad del algoritmo desarrollado.

3. Demostrar que el algoritmo desarrollado resuelve correctamente el problema.

4. Programar el algoritmo, ambos para grafos no-dirigidos como para grafos dirigidos (pueden ser programas separados).

5. Aplicar el programa con grafos aleatorios de $n = 2^6, 2^7, \ldots, 2^{10}$ vertices, y grado promedio (para los vértices) de $s \approx \frac{3}{5}, 2, 4$ y $\frac{n}{16}$, $\frac{n}{4}$ (no-dirigidos) o $s \approx \frac{5}{2}, 4, 8$ y $\frac{n}{8}$, $\frac{n}{2}$ (dirigidos).
Observación: si el algoritmo desarrollado no permite llegar a estos tamaños, ajustar los experimentos y justificarlo en el informe.

6. Entregar un informe (archivo pdf), detallando el análisis teórico de los algoritmos desarrollados, y los resultados de programación obtenidos.

7. Entregar los programas utilizados, bien escritos y documentados, en lenguaje C o C++.

5. Evaluación

Observación: Para este laboratorio, la eficiencia de la solución encontrada afectará la nota:

- Complejidad $O((\log n)^3)$: nota máxima 7.
- Complejidad $O((\log n)^4)$: nota máxima 6.
- Complejidad $O((\log n)^5)$: nota máxima 5.
- Complejidad polinomial de grado superior a 5: nota máxima 4.
- Complejidad exponencial: nota máxima 3.

Luego de eso, la nota del laboratorio se calculará según la ponderación siguiente:

- Análisis [**60 %**]:
  La demostración de validez del análisis de complejidad es correcto.
- Informe [**25 %**]:
  El informe detalla el análisis teórico, los algoritmos desarrollados, y los resultados de programación obtenidos.
- Implementación [**15 %**]
  El programa está escrito de forma que puede ser leído y/o re-utilizado fácilmente por otros programadores: la redacción es limpia (con espacios y divisiones claras) y bien documentada, las sub-funciones y las variables tienen nombres naturales (que indican a que sirven) o acompañadas de comentarios aclarando a que sirven. Ambos algoritmos dan el mismo resultado.