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 2016

Matemática Computacional – Laboratorio 4

1. Búsquedas exhaustivas

Para varios problemas relacionados con grafos, no existe otra solución que considerar todos los casos posibles de manera exhaustiva. A pesar de llamarse exhaustiva, es decir que "verifica todos los casos posibles", en general se pueden eliminar varios casos rápidamente, y con un poco de esfuerzo y de organización se puede controlar el proceso, su tiempo de ejecución y la cantidad de memoria que se utiliza para hacerlo. Por ejemplo, si estamos buscando el ciclo más corto en un grafo y que ya encontramos un ciclo de longitud 7, entonces si la rama actual de la búsqueda solo puede generar ciclos de longitud > 7, se puede descartarla sin exhaustar sus casos posibles uno por uno.

Para grafos, muchas de las técnicas de búsqueda se pueden dividir entre *búsquedas en profundidad* (Depth-First Search, DFS) y *búsquedas en anchura* (Breadth-First Search, BFS). En este laboratorio utilizaremos ambas técnicas para buscar ciclos de un tipo especifico.

2. Ciclo mínimo

Para encontrar un ciclo mínimo pasando por un vértice v_0, podemos hacer una búsqueda en anchura. Para eso, empezamos desde el v_0 y construimos anillos concentricos, con v_0 como 0-esimo anillo. Los anillos se construyen como sigue:

- Para construir un nuevo anillo, se utiliza todas las aristas saliendo de los vértices del anillo anterior (a parte de las aristas utilizadas para llegar a estas aristas).
- Si en la construcción del k-esimo anillo se re-encuentra un vértice ya visitado (sea del anillo en construcción o del anillo anterior), entonces obtenemos un ciclo de longitud $2k+1$ (el vértice repetido está en el anillo anterior) o $2k+2$ (si es un vértice del anillo en construcción.
- Si no se encontró ningún ciclo durante la construcción de los primeros k anillos, entonces no existe de longitud $\leq 2k+2$ que pasan por v_0.

En efecto, lo que estamos construyendo es un árbol con raíz en v_0 y tal que la altura de cada vértice v_i corresponde a la distancia mínima entre v_0 y v_i (los vértices no conectados a v_0 no aparecen en el árbol). La búsqueda se dice "por anchura" porque intentamos construir cada anillo lo más ancho posible.

El proceso sigue hasta que se repite un vértice ya encontrado o que no quedan ninguna arista no utilizada que sale de uno de los vértices conectados. Hay que destacar un caso particular: si el vértice repetido está en el anillo en construcción, entonces el ciclo podría no ser mínimo si existe alguna arista entre dos vértices del anillo anterior, por lo que sabremos si este ciclo es mínimal o no hasta haber terminado el chequeo de las aristas saliendo de vértices del anillo anterior.

Para encontrar un ciclo mínimo en el grafo G, se repite este proceso para cada vértice de G o hasta encontrar un ciclo de largo 3 (que automáticamente es mínimo).

3. Grafo conectado

En algunos casos, antes de buscar un ciclo conviene asegurarse que el grafo es conectado (como era el caso para el ciclo de Euler en el laboratorio anterior). Para eso, podemos empezar con cualquier vértice v_0 y hacer una búsqueda en anchura obteniendo anillos (concentricos) de todos los vértices que se puede alcanzar desde v_0 pasando por k aristas (o menos).

Esta búsqueda se parece mucho a la búsqueda de ciclo mínimo pasando por v_0, pero sin recordar los ciclos encontrados, y terminando solamente cuando hemos conectado todos los vértices o cuando no quedan aristas no utilizados conectados al último anillo de vértices.

4. Ciclo de Hamilton

Un ciclo de Hamilton es un ciclo cerrado que recorre todos los vértices del grafo exactamente una vez, sin repetir ninguno. Para tener un ciclo de Hamilton, un grafo debe ser conectado y todos los vértices deben tener grado a lo menos 2. A pesar de ser condiciones mínimas, estos requerimientos no son suficientes para determinar la existencia de un ciclo de Hamilton, y en general es un problema difícil determinar si un grafo admite on no un ciclo de Hamilton.

El método que presentamos aquí es la *búsqueda en profundidad*, ya que siempre intenta ir lo más lejos posible del vértice de origen. Este algoritmo utiliza un sistema de retroceso ("Backtracking") para verificar todos los casos posibles. La implementación de tales algoritmos requiere un cuidado especial para asegurar que el algoritmo termina y no entrará en un ciclo infinito. Aquí construiremos cadenas de vértices consecutivos y no repetido conectados por aristas del grafo.

1. Verificar que el grafo es conectado y que todos los vértices tienen grado a lo menos 2. Si no, el grafo no puede tener ciclos de Hamilton.

2. Empezar con un vértice v_1 (se puede elegir cualquier vértice).

3. Cada vez que se agrega un vértice v_i, se genera un listado de las aristas saliendo desde v_i donde el vértice de llegada no está en la cadena actual. Hablaremos del *listado de aristas no verificadas para* v_i.

4. Si hay aristas no utilizadas para v_i, elegir una, agregar el nuevo vértice v_{i+1} a la cadena construida, y volver al paso 3.

5. Si $i = n$, la cadena da un ciclo de Hamilton (y el algoritmo termina) **si y solo si** la arista (v_n, v_1) está en el grafo.

6. Si el algoritmo no terminó y que no quedan aristas disponibles para v_i, $i > 1$, retroceder al vértice anterior (v_{i-1}), eliminar el arista (v_{i-1}, v_i) del *listado de aristas disponibles para* v_{i-1} y volver al paso 4.

7. Si no quedan aristas disponibles para v_1, entonces el grafo no tiene ningún ciclo de Hamilton.

Observamos que para el vértice v_1 se puede descartar una de las aristas de la lista inicial de posibilidad porque se requieren dos aristas para cada vértice y los ciclos son simétricos (se pueden leer en ambas direcciones).

5. Generación aleatoria de grafos

Para este laboratorio, generaremos grafos aleatorios siguiendo las mismas técnicas que para el Laboratorio 3, pero con algunos cambios:

- No se pide que los vértices del grafo tengan grado par (no se arreglará el grafo para eso).
- No se impone que los grafos sean conectados, por lo tanto las aristas $(a, a+1)$ también se incluirán o no según el proceso aleatorio.

El pseudo-codigo para generar estos grafos se encuentra en el Algoritmo 1.

Algorithm 1 Generación aleatoria de grafos con un ciclo de Euler

Require: Número n de vertices del grafo y propoción p de aristas por vertice.
Ensure: Grafo $G = (V, A)$ (vertices y aristas) que admite un ciclo de Euler.

```
1: V ← {1, ..., n}
2: A ← ∅
3: for a desde 1 hasta n − 1 do
4:    for b desde a + 1 hasta n do
5:       if Rand() < p then
6:          Incluir (a, b) en A
7:       end if
8:    end for
9: end for
10: return G = (V, A)
```

6. Laboratorio

- Desarrolle un programa en lenguaje C para generar grafos aleatorios (sin más cáracteristicas impuestas).

- Desarrolle programas en lenguaje C para encontrar un ciclo mínimo, determinar si el grafo es conectado y para encontrar un ciclo de Hamilton (si hay).
- Evaluar el costo computacional de cada búsqueda.
- Justificar porque el algoritmo para encontrar un ciclo de Hamilton termina y siempre encontrará un ciclo de Hamilton si existe a lo menos uno.

7. Se solicita

1. Implementar los algoritmos en lenguaje C.
2. Determinar los costos computacionales con $p \in \{1/n, 2/n, 4/n, 8/n\}$ para distintos valores de n.
3. Determinar los costos computacionales con $n = 1000$ para distintos valores de p.
4. Se debe entregar:
   - Código fuente de los algoritmos.
   - Archivo "Readme" con las instrucciones (detalladas) de compilación.
   - Informe en LaTeX que contiene:
     - Formulación de los experimentos.
     - Información del hardware (procesador) y software (librerias) utilizado.
     - Curvas de desempeño de los resultados.
     - Conclusiones.
5. Normas a cumplir:
   a) Número de Integrantes: máximo 2.
   b) Plazo de Entrega: 14 días.
   c) Forma de envío: se envía directorio comprimido zip, gz, tar, o tgz (no se aceptan otro formatos). a: **nicolas.theriault@usach.cl**
6. El nombre del archivo (y directorio) debe indicar el laboratorio ("Lab1") y a lo menos las iniciales (nombre y apellido) de cada integrante.
   - Se puede utilizar solamente el primer nombre y primer apellido de cada integrante.
   - No deber haber acentos (á → a, ñ → n, etc).
   - No deber haber espacios en blanco, utilizar "_" para separar palabras e iniciales.
   - Ejemplo: Rodrigo German Abarzúa Ortiz y Nicolas Thériault → Lab4_RA_NT.

8. Evaluación

La nota del laboratorio se calculará según la ponderación siguiente:

- Informe [**20 %**]:
  El informe está en LaTeX, en lenguaje formal, y contiene todos los detalles exigidos.

- Algoritmos y análisis [**40 %**]:
  Justifica/demuestra que el algoritmo de búsqueda de ciclo de Hamilton es correcto. Estudio de la eficiencia temporal del algoritmo implementado y estimación de como crece (basado en suficientes puntos de evaluación para justificar las conclusiones).

- Implementación [**40 %**]
  El código fuente no presenta errores o advertencias de compilación. 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.