Análisis y diseño de software

dit
UPM

2019.02.16

Ejercicio 2

Juan Antonio de la Puente <jpuente@dit.upm.es>

ETSIT UPM

Objetivos

- Algoritmos sobre grafos
  - Algoritmo de Dijkstra

- Tareas:
  1. Medir tiempos de ejecución
  2. Validar la complejidad del algoritmo

Preliminares

- El algoritmo usa profusamente los métodos
  - Node getNode(String name)
  - List<Link> getLinks(Node node)

  por lo que estos métodos deben ser razonablemente rápidos para que no contaminen la evaluación del algoritmo
- Sugerencia: use diccionarios internamente
  - private Map<String, Node> nodeMap
  - private Map<Node, List<Link>> linkMap

  de forma que los métodos citados sean O(1)
- Estos diccionarios se van rellenando con addNode() y addLink()

Medidas

- Se espera complejidad cuadrática
- Sugerencia:

```
public class DijkstraMeter {
  ...
  public static void main(String[] args) {
    for (int n = 1000; n<= 40000; n += 2000) {
      Graph graph = new Graph();
      load(graph, n);          // generar grafo
      long t = doit(graph);  // medir tiempos
      System.out.printf("%s %d%n", n, t);
    }
  }
  ...
}
```

load()

- Para generar un grafo del tamaño deseado
  1. Genere N nodos
     - ✓ llámelos por su número de forma que siempre se puedan seleccionar nodos como
       - Node nodo = graph.getNode(“27”);
  2. Para cada nodo, genere un número fijo, X, de enlaces a nodos elegidos aleatoriamente
     - ✓ por ejemplo, X = 5
- Esto nos deja un grafo donde probablemente todos los nodos estén enlazados

doit()

- Medir tiempos de ejecución para un grafo de n nodos
- Sugerencia

```
  public static long doit(Graph graph) {
    Node n0 = graph.getNode("0");
     long t0 = System.currentTimeMillis();
     new Dijkstra(graph, n0);
     long t2 = System.currentTimeMillis();
     return t2 – t0;
  }
```

Opcional

- Si vamos a medir desde un nodo A, calcule cuantos nodos son alcanzables desde A

  crear un conjunto vacío S

  crear una cola de nodos, iniciada con A

  mientras la cola no está vacía

    se extrae un nodo B de la cola

    si B ya está en S, continuamos

    se añade B a S

    para cada enlace desde B

      se añade a la cola el destino del enlace

- S contiene todos los nodos alcanzables, que usaremos como tamaño efectivo del grafo para las medidas

Realización del ejercicio

- Se pueden formar grupos de 1, 2 o 3 alumnos
  - obligatorio poner los nombres en @author en cada fichero java
- Si el grupo es de 2 o más alumnos, conviene repartir tareas
  - al menos deben ser diferentes personas las que preparen los casos de prueba y las que preparen la solución

Cómo proceder

- Descargue el archivo ADSW_ejercicio2.zip de moodle e importe el proyecto que contiene en eclipse
- Copie las clases del ejercicio 1 en la carpeta src
  - ahora el paquete es es.upm.dit.adsw.ej2
- Implemente la clase DijkstraMeter en el mismo paquete
  - ejecute la clase para medir los tiempos de ejecución del algoritmo Dijkstra para distintos valores de N
  - realice un análisis de regresión con el programa Corrector
- Prepare un documento pdf con
  - medidas obtenidas
  - capturas de correlator
  - razonamiento de complejidad
- Guarde el documento en la carpeta *docs* de eclipse

Entrega

- Ejecute `Entrega.launch` desde eclipse
  - obtendrá mensajes orientativos
  - la nota definitiva se asigna por los profesores
  - puede comprobar que el fichero entrega.zip se ha subido a moodle