Geometría Computacional 2015-2
Practica 01

Prof. Adriana Ramírez Vigueras
adriana.rv@ciencias.unam.mx

Ayudante Lab. Diego Andrés Gómez Montesinos
diegoMontesinos@ciencias.unam.mx

06 Febrero 2015

1. Objetivos:

- Que implementes el ordenamiento radial dado un centro usando la orientación de vectores.
- Que hagas una implementación del concepto de gráfica (o grafo) y un algoritmo de búsqueda (BFS o DFS).
- Que aprendas a hacer visualizaciones con el *framework* Processing.

2. Marco teórico:

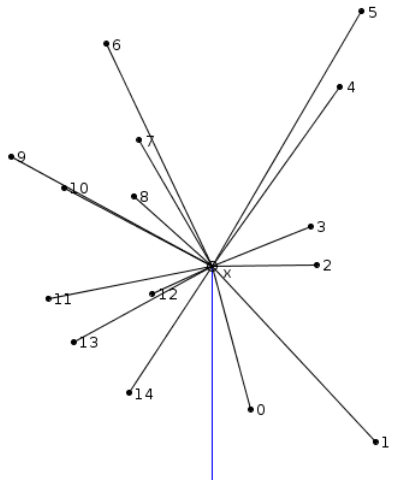
Una gráfica G, o grafo, es un par ordenado $G :=< V, E >$, donde V es un conjunto de elementos llamados vértices y E, es un conjunto de parejas de vértices de la manera (x, y), con $x, y \in V$, llamadas aristas.
Se dice que x es adyacente a y, si y solo si, existe la arista (x, y) en la gráfica G.
Una gráfica no dirigida, es una gráfica donde si $(x, y) \in E$ es un par no ordenado.

Existen dos algoritmos para recorrer una gráfica, o hacer búsquedas sobre ella, de manera ordenada: por anchura (BFS - Breadth First Search), o por profundidad (DFS - Depth First Search).

3. Descripción:

En esta practica deberás hacer la implementación de una gráfica no dirigida cuyos vértices tienen un identificador y están fijados a una posición.
Y además, para ir calentando tus habilidades de transformar un algoritmo en papel a un programa, implementarás alguna de las dos búsquedas sistemáticas en una gráfica: BFS o DFS.
Tu eliges cual implementar.

En estos algoritmos, en cada paso se necesita obtener la vecindad de un vértice x, es decir, todos los vértices que son adyacentes a x.
En tu implementación deberás obtener esta vecindad como una lista con la siguiente propiedad: imagina que tomamos a x como un centro y un rayo en una cierta dirección inicial, si vamos rotando el rayo en sentido contrario a las manecillas del reloj, los elementos de lista apareceran conforme el rayo los intersecte.
Por convención, la dirección inicial será el vector $\vec{e_2} = (0, 1)$.



(HINT) Define una operación de comparación en base al orden de vectores (vuelta derecha/izquierda) visto en clase.

4. Implementación:

La carpeta src tiene la siguiente estructura:

```
src
  |-- algorithms
  |     |-- graph
  |         |-- BFSearch.java / DFSearch.java
  |-- structures
  |     |-- graph
  |         |-- Graph.java
  |         |-- GraphVertex.java
  |-- visualization
  |     |-- graph
  |-- Main.java
```

En la subcarpeta structures/graph viene la estructura de datos que ocuparás.
La clase GraphVertex representa un vértice en la gráfica y ocupa una clase del core de Processing llamada PVector que soporta operaciones de aritmética de vectores. En tu implementación eres libre de ocupar todos los métodos de esta clase.
La clase Graph es la que tienes que acompletar para aprobar los test referentes a la estructura.

En la subcarpeta algorithms/graph viene el código base para el algoritmo de búsqueda que elgiste. La clase BFSearch y DFSearch es la que tienes que acompletar para aprobar los test referentes al agoritmo.

5. Pruebas:

Hay dos *suites* de pruebas una que prueba tu implementacion de la gráfica y otra que prueba tu implementación de la búsqueda.
Para correr las pruebas en la consola estando en el directorio de la práctica, se ejecuta el comando *ant test*. Esto genera un directorio llamado report donde se encontrarán dos archivos correspondientes a los resultados de las dos *suites*.

6. Visualizacion:

Para la parte de visualización deberán implementar un sketch de Processing con las siguientes especificaciones:

- Deberán crear una gráfica aleatoria con 10 vértices y mostrarla en pantalla. Deben mostrarse claramente los identificadores de cada vértice.
- Cada vez que el usuario de click izquierdo sobre un vértice se marca como vértice de inicio actual, y cuando de click derecho sobre un vértice se marca como vértice de final actual.
  Si se eligio el vértice de inicio, se imprime en consola el orden de recorrido desde el nuevo vértice de inicio.
- Se tendrán un evento asociado a los caracteres:
  's' - Guarda el frame como practica01.png.
  't' - Hace un overlay de la gráfica mostrando el árbol recorrido desde el vértice de inicio actual.
  'p' - Hace un overlay de la gráfica mostrando el camino del vértice de inicio actual al vértice de final actual. Se debe informar de alguna manera si no existe tal camino.
- Cada vez que el usuario de click sobre el canvas se generará otra gráfica.
  Se debe informar en consola si la gráfica generada es conexa o no.

La visualización deberá correr con la clase Main, y se ejecutará con el comando *ant run*.

7. IMPORTANTE:

Recuerda modificar la ruta de la instalación de Processing en el archivo build.xml, si tienes dudas de cómo hacerlo manda un correo al ayudante.

8. Entrega:

Esta práctica se entrega el miércoles 18 de Febrero.