Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Taller 3

Aplicaciones de Unión-Búsqueda y Colas de Prioridad

Clusterización por vecinos cercanos

Los métodos de clusterización de datos buscan agrupar conjuntos de puntos "similares". Para efectos del ejercicio, se toma un conjunto de puntos 2D que el programa lee de un archivo plano separado por comas, donde cada línea representa las coordenadas de un punto.

El agrupamiento por vecinos cercanos se implementa de la siguiente forma:

1) Cada punto del dataset es inicialmente un cluster de un elemento.
2) Por cada pareja de puntos p_i, p_j en el conjunto de datos, se calcula la distancia y se agregan a una cola de mínima prioridad (`MinPQ`).
3) Se procesan las parejas de puntos en orden creciente según su distancia (se toman de la cola de mínima prioridad) y se unen mientras que su distancia sea menor a un umbral máximo D_{max} configurable.
4) Cuando ya no queden puntos con distancia menor a D_{max}, término el proceso de clusterización. Se reportan el número de clústeres obtenidos y el tiempo que tardó la clusterización.

Ejercicios a desarrollar

1. Implementar la clase `Clustering`, que implementa como uno de sus métodos la operación `clusterizar` la cual retorna el número de clusters.
   `int clusterizar(Puntos2D[] puntos)`
2. Implementar una función de biblioteca para leer los puntos del archivo .csv:
   `Puntos2D[] leerPuntos(String filename)`
3. Un método `clasificar(Punto2D p, k)` permite clasificar nuevos puntos, buscando los k vecinos más cercanos y asignando el número de la componente conexa que más se repite entre los k vecinos.
4. Un método randomTest() genera 10 puntos aleatorios, los clasifica y reporta a que cluster pertenece cada punto.
5. Un método `graficarClusteres()` permite visualizar los puntos de cada cluster asignado un color distinto a los puntos cada cluster. Sugerencia: Utilizar StdDraw.

6. El método `main` toma como parámetro del nombre del archivo, lee los puntos y realiza la clusterización y reporta el número de clústeres obtenidos. A continuación visualiza la gráfica de los clusters y ejecuta la prueba de puntos aleatorios.
7. Estimar analíticamente el desempeño del algoritmo `clusterizar`.
8. Estimar analíticamente el desempeño del algoritmo `clasificar`.

Entregables

Remitir el código fuente de la solución implementada y un documento con el análisis de tiempo de los algoritmos `clusterizar` y `clasificar` (se aceptan Word, LibreOffice, PDF). Nombrar el archivo comprimido `TallerOpcional3-<Nombre1>-<Nombre2>…(.zip .rar .7z o .tgz)`. Para estandarizar la forma de invocar el programa, ubicar el método main y las funciones de biblioteca solicitadas en la clase `TallerOpcional3`.

En caso de utilizar estructuras de las bibliotecas del texto (`algs4.jar`) **no** anexar la biblioteca.

Grupos máximo de 3 personas.

Puntos de prueba

Archivos con puntos de prueba para el ejercicio:

- datapoints-k=2-n=200.csv (caso de prueba con 2 clusters bien separados)
- datapoints-100.csv
- datapoints-120.csv
- datapoints-150.csv
- datapoints-1000.csv
- datapoints-2500.csv
- datapoints-5000.csv

Estos archivos contiene puntos en el rango (-2,-2) a (2,2). Usar distancias máximas del orden de D_{max} en el rango 0.1 - 0.3 aproximadamente.