1. Introducción

En el área del Análisis de Flujos de Datos (*Data Stream Analysis*) el objetivo es desarrollar algoritmos que extraigan información útil de una colección de datos de gran volumen en el siguiente "escenario":

1. Los datos forman una secuencia (*flujo*) de longitud muy grande; frecuentemente, dichos datos llegan en línea, uno tras otro en rápida sucesión.

2. No se dispone de mucha memoria auxiliar; ni la secuencia ni tan sólo una fracción de la misma puede ser almacenada. Si la longitud de la secuencia es N se considera aceptable que el espacio auxiliar utilizado sea $O(\log N)$; obviamente lo mejor es que sea $\Theta(1)$.

3. La secuencia sólo puede recorrerse un número pequeño de veces e idealmente sólo una vez; de hecho, cuando la secuencia es en línea, sólo se puede efectuar una pasada sobre la secuencia. Cada elemento debe procesarse lo más rápido posible, el algoritmo debe ser sencillo y muy eficiente. El hecho de que la memoria auxiliar sea muy pequeña ayuda en ese sentido; la información puede residir en la caché y/o registros y el algoritmo evita costosos accesos a memoria principal.

4. No debe realizarse ninguna hipótesis estadística sobre la secuencia.

Las aplicaciones de este tipo de algoritmos son numerosas, incluyendo entre otras la monitorización del tráfico en redes, la optimización de *queries* en bases de datos, la minería de datos, las finanzas (*high-frequency trading*), el comercio electrónico, los sistemas de recomendación, etc.

Algunos de los problemas más conocidos e importantes incluyen:

1. Hallar los k elementos más frecuentes, para un k dado—esta tarea es muy importante pero en absoluto trivial, ya que no tenemos suficiente memoria auxiliar para guardar los elementos distintos de la secuencia y su frecuencia.

2. Hallar o contar k-elefantes, k-ratones y c-icebergs: un k-elefante es un elemento cuya frecuencia de aparición es mayor que k; un k-ratón es un elemento cuya frecuencia es menor que k y un c-iceberg es un elemento cuya frecuencia relativa (porcentaje sobre el número total de elementos) es superior a c $(0 < c < 100)$.

3. Determinar el número de elementos distintos, la *cardinalidad* de la secuencia; la cardinalidad n será típicamente $\ll N$, siendo N el número total de elementos; pero n es suficientemente grande para imposibilitar el almacenamiento de los n elementos distintos de la secuencia en la memoria auxiliar.

En esta práctica nos centraremos en el último de estos problemas: la estimación de la cardinalidad. Muchos algoritmos de *streaming* no pueden llevar

a cabo de manera exacta el cálculo deseado, por las limitaciones en cuanto a espacio de memoria y tiempo. Por esa razón, lo que se busca son algoritmos que nos den una buena estimación del valor real deseado, una estimación para la cual el error relativo cometido sea pequeño. En el caso de la cardinalidad, nos encontramos en la situación comentada: no podemos encontrar el valor exacto n si no disponemos de suficiente memoria auxiliar, por ello debemos contentarnos con una estimación $\hat{n}$ del valor real.

2. Pasos a seguir en la realización de la práctica

1. Buscar información en libros, artículos en revistas, Internet, etc. para documentarse sobre el tema y estudiar uno o más de los algoritmos de estimación de cardinalidad existentes

2. Elegir un algoritmo e implementarlo en C++. La mayoría de algoritmos utilizan funciones de hash, a cada elemento de la secuencia se le aplica la función de hash y el algoritmo trabaja con el *hashvalue* obtenido; la mayoría de algoritmos trabajan con una pequeña tabla de M elementos/hashvalues/contadores, siendo M un valor que puede fijar el usuario.

3. Escribir un pequeño programa que permita utilizar el algoritmo de forma sencilla; lo único que be hacer es escribir simplemente dos números: la estimación de la cardinalidad y el número total de elementos leídos. El programa debería poder usarse por ejemplo así:

   ```
   % cardest < D4.dat
   5701 27266
   ```

   Naturalmente es aún mejor si puede usarse así

   ```
   % cardest -M 256 D4.dat
   5742 27266
   ```

   donde el *flag* `-M` le permite al usuario especificar el tamaño de la memoria auxiliar que se quiere usar. Se asumirá que la secuencia de entrada es una secuencia de strings, cada uno de ellos de longitud ≤ 30.

4. Escribir otro programa que permita llevar a cabo un conjunto de experimentos extenso. En la página web de la asignatura encontraréis un conjunto de ficheros (*datasets*), cada uno de los cuales consiste en una secuencia muy grande de strings de longitud ≤ 30. El número de elementos distintos en cada dataset es un dato conocido para poder comparar la diferencia entre el valor real y los valores estimados. Para cada *dataset* deberéis:

   a) Aplicar el algoritmo de estimación un cierto número K de veces (al menos $K = 50$) y obtener con cada ejecución una estimación de la cardinalidad $\hat{n}_i$, $i = 1, \ldots, K$. Las estimaciones varían de una ejecución

a otra al usar funciones de hash universales, con una instancia diferente en cada ejecución: en la ejecución i usad la función $h_i(x) = a_i x$ mód p donde a_i es un número elegido al azar.

b) Preparar tablas y gráficas que muestren, para cada dataset por separado, las estimaciones $\hat{n}_i$ obtenidas, los errores relativos cometidos ($|n - \hat{n}_i|/n$) en cada ejecución, el valor medio de las estimaciones $\overline{n} = \frac{1}{K} \sum_{1 \leq i \leq T} \hat{n}_i$, el error estándar

$$SE = \frac{\sqrt{\sum_{i=1}^{K} \hat{n}_i^2 - n^2}}{n}$$

y en cuántas estimaciones se comete un error relativo menor al 1 %, en cuántas un error relativco entre el 1 % y el 5 %, en cuántas entre el 5 % y el 10 %, etc. También debe obtenerse el tiempo de la ejecución i-ésima T_i y el tiempo medio $\overline{T} = \sum_{1 \leq i \leq K} T_i$ y el tiempo medio por elemento $\bar{t} = \overline{T}/N$ siendo N el número total de elementos del datasat. Puede convenir hacer un programa que recibe un *dataset* y los valores K, n y N y genera un archivo con toda la información necesaria para preparar los gráficos y tablas.

Adicionalmente deberéis

c) Preparar una tabla resumen con los valores $\overline{n}$, SE y $\bar{t}$ de cada dataset.

d) Para el dataset `D1.dat` se llevará además el mismo tipo de experimentos descritos en los apartados anteriores, pero variando el parámetro M para estudiar empíricamente como al aumentar la memoria auxiliar disponible M mejora la precición de las estimaciones (disminuye el SE). Debe experimentarse al menos con 10 valores distintos de M. Preparar gráficas y tablas que muestren la relación entre M y la calidad de las estimaciones. No es necesario dar los datos de cada una de las K ejecuciones del algoritmo con un valor de M fijado, sólo hay que presentar en las tablas y gráficas como varían los parámetros $\overline{n}$, SE, $\bar{t}$ en función de M. Lo más sencillo es tener un programa similar al anterior pero en el que se indique también los valores de M con los que se ha de repetir cada experimento; o bien generalizar el programa anterior, de modo que si no se especifica los valores de M con los que hacer las ejecuciones sólo se harán K ejecuciones fijando un valor M por defecto, p.e. $M = 128$.

5. Escribir un informe de la práctica. El informe contendrá al menos las siguientes partes:

a) Resumen de la investigación bibliográfica realizada. Descripción a alto nivel (p.e. con pseudocódigo) del algoritmo escogido.

b) Descripción breve del programa para llevar a cabo los experimentos. Tablas y gráficas debidamente comentadas.

c) Conclusiones del trabajo.

d) Apéndices. Código C++ del algoritmo de estimación. Código C++ de estructuras de datos y funciones auxiliares. Código C++ del programa que realiza la simulación y la recogida de datos. Bibliografía consultada, enlaces a páginas Web consultadas.

3. Entrega

La práctica se realiza en equipos de dos personas. La entrega se efectuará no más tarde del día 12 de Enero de 2014 a las 12:00. Tenéis que enviar un archivo `tar.gz` o `zip` a la dirección `alg@lsi.upc.edu`. Es buena idea que el nombre del archivo esté formado por los apellidos de los dos intehrantes del equipo, separados por un guón, p.e., `canals-perez.zip`. En el *subject* del mensaje poned `PRACTICA TA` *`Apellido1 Apellido2`*. No olvidéis incluir el nombre completo de los dos integrantes del equipo en la portada del informe. Además debéis dejarme una copia en papel (sin encuadernar, basta una sola grapa en la esquina superior) del informe de la práctica; podéis dejar la copia en mi buzón (2ª planta del edificio Omega, despacho 241).

El contenido del archivo enviado por correo electrónico será el siguiente:

1. Un archivo PDF con el informe.

2. Los ficheros fuente `.cc`. Es recomendable dividir el programa en dos o tres módulos separados. Deberían ir en un subdirectorio `src`. Agregad un fichero `README` que explique que contiene cada archivo y cómo obtener los programas ejecutables pedidos: el que hace una estimación única sobre un archivo o secuencia dados, y el o los que llevan a cabo los experimentos y recogen los datos para cada *dataset*. Explicad en el fichero cómo se han de usar los ejecutables (los parámetros se dan en la línea de comandos como argumentos? se introducen por el canal estándar de entrda? se han de escribir en un fichero auxiliar aparte?

3. Los ficheros con los datos experimentales recogidos y a partir de los cuales se preparan las tablas y gráficas. Para cada dataset, los datos experimentales obtenidos deberían ir en uno o más ficheros específicos. Poned todos estos ficheros en un subdirectorio `data`.

4. Los ficheros de los gráficos se pueden poner en un subdirectorio `plots`. No es obligatorio dar los ficheros con los gráficos, ya que los gráficos irán incorporados en el informe, De manera parecida si queréis ilustrar alguna parte de vuestro informe mediante diagramas, figuras, etc. es algo muy aconsejable, y los ficheros correspondientes también los podéis enviar: en tal caso, poned los ficheros de este tipo en un subdirectorio `figs`.

Para la preparación de las gráficas aconsejo utilizar `gnuplot`, es bastante fácil de usar. Si sabéis LaTeX os animo a utilizarlo para preparar el informe.