PRÁCTICA 2: EN FIN MARCIAL, MÁS CORRE EL LOGARITMO QUE EL LINEAL (Y MÁS CORRE SI EL CAMINO ES LARGO)

1. Introducción

Todo buen científico o ingeniero[1], debe ser como Santo Tomás, que hasta que no toca no cree. Es decir, una vez obtenido un resultado teórico, antes de aceptarlo como verdadero, hay que confirmarlo en la práctica mediante experimentación. Si los experimentos contradijesen lo que se predecía a nivel teórico, habría que revisar la teoría, por si contuviese algo incorrecto; más el diseño de los experimentos, por si éstos estuviesen mal realizados o las condiciones en las cuales se realizaron entraban en conflicto con lo asumido por la teoría.

2. Objetivos

El objetivo de esta práctica es aprender como analizar a nivel empírico, es decir, mediante experimentación, el ritmo de crecimiento u orden de uno o varios algoritmos, con objeto tanto de: (1) confirmar los resultados obtenidos a nivel teórico sobre su complejidad; como de (2) comparar empíricamente la eficiencia de varios algoritmos.

3. Actividades

3.1 Implementación de los algoritmos de búsqueda lineal y binario.

Implementar los algoritmos de búsqueda en un vector ordenado lineal y de búsqueda binaria. Ambos algoritmos se proporcionan en pseudocódigo en el apéndice A de esta práctica.

3.2 Análisis empírico de la eficiencia de dichos algoritmos

Una vez implementados ambos algoritmos, se deberá analizar empíricamente la eficiencia temporal, o tiempo de ejecución, de dichos algoritmos. La eficiencia se medirá en función del tamaño del vector a ordenar. Por tanto se deberá:

1. Decidir que tamaños de vector se van a analizar. Estos tamaños han de ser diferentes, variados y suficientes. Cuantos más tamaños se analicen, mejor (dentro de lo razonable).
2. Por cada tamaño, se deben analizar varios casos de prueba. Un único caso por tamaño no es estadísticamente relevante. Se recomienda un mínimo de 10 ejemplares por tamaño de vector, probando diversas condiciones de búsqueda. Por ejemplo, no es lo mismo buscar un elemento que está en el vector o uno que no está en el vector.
3. El tiempo de ejecución para cada tamaño será la media de los tiempos de ejecución de cada ejemplar para ese tamaño.
4. Cada ejemplar concreto se deberá aplicar a ambos algoritmos.
5. Una vez que obtengamos los tiempos medios de ejecución para cada algoritmo y para cada tamaño del vector, dibujar (con ayuda de un computador o a mano) la gráfica con los tiempos de ejecución de cada algoritmo en función de n.

En lugar de realizar la ejecución de cada algoritmo manualmente, se aconseja crear un programa de análisis de los algoritmos que realice todas las medidas de forma automática y conjunta. Dicho programa imprimirá los diferentes resultados por pantalla (o los guardará en

[1] Aunque el Dr. Sheldon Cooper califique a los ingenieros de "*Umpa Lumpas*" de la ciencia.

un archivo). De esta forma podremos dejar el computador trabajando, pongamos por espacio de ocho horas, y si nada no previsto ocurre, recoger los resultados todos juntos al final.

4. Criterios de Evaluación y Aclaraciones

El principal elemento a evaluar de esta práctica será la calidad del análisis realizado. Se evaluará especialmente la selección de los tamaños del vector, el número de muestras recogidas, la hetereogenidad de los casos de prueba y la realización de un programa que recopile todos estos datos de forma automática. Las gráficas a mano adecuadamente dibujadas tendrán la misma nota que su equivalente creado mediante computador.

La práctica se entregará a través de la plataforma *moodle* siguiendo las instrucciones en ella proporcionadas. Se deberá rellenar un informe proporcionado a través de la plataforma *moodle*.

A través de *moodle* se proporcionan clases Java para generar vectores ordenados y números aleatorios para buscar en dichos vectores, así como un ejemplo de cómo se miden tiempos en Java.

Apéndice A. Algoritmos de búsqueda lineal y ordenados

A.1 Algoritmo de búsqueda lineal en un vector ordenado

```
// Pre: (tamaño(v) >= fin)
FUNCION esta(v : VECTOR(ENTERO), e: ENTERO, fin: NATURAL) : BOOLEANO ES
   i : NATURAL;
   i := 0;
   MIENTRAS ((i < fin) AND (v[i]!= e)) HACER
      i := i + 1;
   FINMIENTRAS
   DEVOLVER (i < fin);
FINFUNCION
```

A.2 Algoritmo de búsqueda binaria en un vector ordenado

```
// Pre: (tamaño(v) == fin) AND (ordenado(v))
FUNCION esta(v : VECTOR(ENTERO), e: ENTERO, fin: NATURAL) : BOOLEANO ES
       estaRecursivo(v,e,0,fin-1);
FINFUNCION
```

```
// Pre: (inicio <= fin < tamaño(v)) AND (ordenado(v,inicio,fin))
FUNCION estaRecursivo(v : VECTOR(ENTERO), e: ENTERO, inicio, fin: NATURAL) :
BOOLEANO ES

       result : BOOLEANO; result := FALSO;

       SI (inicio == fin) ENTONCES
          result := (v[inicio] == e)
        SINO
          medio : NATURAL; medio := (inicio + fin) DIV 2;
          SI (v[medio] == e) ENTONCES
             result := VERDADERO;
          SINOSI (e < v[medio]) ENTONCES
             result := estaRecursivo(v,e,inicio,medio-1);
          SINO
             result := estaRecursivo(v,e,medio+1,fin);
          FINSI
        FINSI

       DEVOLVER result;
FINFUNCION
```

Pablo Sánchez Barreiro.