TEORÍA DE ALGORITMOS

CURSO 2010 – 2011

PRÁCTICA DE DIVIDE Y VENCERÁS

INGENIERÍA INFORMÁTICA

E.T.S.I INFORMÁTICA Y DE TELECOMUNICACIONES

UNIVERSIDAD DE GRANADA

Objetivo

El objetivo de esta práctica es que el alumno comprenda los aspectos básicos de la técnica divide y vencerás y los aplique en la resolución de problemas.

Introducción

Este guión se divide en dos partes. En la primera, utilizando el algoritmo de ordenación por mezcla ("mergesort"), se realizará el análisis para la determinación de los umbrales en forma híbrida y por tanteos. En la segunda, se deben resolver 3 problemas de complejidad creciente aunque todos con estructura similar y el mismo esquema algorítmico

1. Determinación de Umbrales para "Mergesort"

El método de ordenación por mezcla o "Mergesort" utiliza la técnica Divide y Vencerás para realizar la ordenación de un vector. Su estrategia consiste en dividir el vector en dos subvectores, ordenarlos mediante llamadas recursivas y, finalmente, en la parte más costosa del algoritmo, se combinan los dos subvectores ya ordenados. Esta idea da lugar al siguiente esquema:

```
PROCEDURE Mergesort(V,prim, ult)
V: es un vector
prim, ult: enteros que indican las posic. inicial y final de V
B, C: vectores auxiliares

 BEGIN
    IF (ult - prim) < UMBRAL
        aplicar algoritmo basico(V, prim,ult)
    ELSE
        mitad = (prim+ult)DIV 2;
        divide(V, prim, mitad, ult, B,C);
        Mezcla(B,prim,mitad);
        Mezcla(C,mitad+1,ult);
        Combinar(V,B,C)
 END
```

Un elemento clave para el buen funcionamiento del algoritmo es la determinación del valor de UMBRAL. Para ello, se realizarán las siguientes tareas a partir del código fuente proporcionado en el guión número 1 de eficiencia:

1. Calcule el umbral exacto según un enfoque experimental: ejecute el algoritmo básico y el divide y vencerás para distintos tamaños de la entrada (eficiencia empírica) y busque el valor para el cual, los tiempos coinciden. Encuentre este valor de "umbral óptimo", primero, a través

del gráfico correspondiente, y segundo, igualando las ecuaciones de ajuste y despejando el valor de N. Debe utilizar tamaños de vector pequeños y un número de repeticiones de la ordenación alto.

2. Seleccione distintos valores (inferiores y superiores al umbral óptimo) y obtenga los tiempos de ejecución reales para distintos casos del problema. Estos son los umbrales de tanteo.

3. Muestre en una gráfica comparativa la evolución del tiempo de ejecución del algoritmo para los distintos valores del umbral estudiados. Es decir, para cada selección distinta del umbral (óptimo y de tanteo) deberá dibujar una curva con los tiempos de los casos medidos empíricamente. Comente los resultados justificando cuál o cuáles de las variantes darían mejores resultados.

2. Deducir, calcular e implementar

Indique que hace el siguiente pseudocódigo. Impleméntelo, calcule su eficiencia teórica y híbrida.

```
procedure ff(A[1...n] de números) -> ( v1, v2)
begin
 if (n == 1)
   return (A[1], A[1])
 else if (n == 2)
   if( A[1] < A[2])
      return (A[1], A[2])
   else
      return (A[2], A[1])
 else
   (vl1, vl2) = ff(A[1...(n/2)])
   (vr1, vr2) = ff(A[(n/2 +1)...n])
   if (vl2 < vr2)
      v2 = vr2
   else
     v2 = vl2
   if (vl1 < vr1)
      v1 = vl1
   else
      v1 = vr1
return (v1,v2)
end
```

3. La suma de la mezcla

Una secuencia de n números naturales $C = \{c_1, c_2, \ldots, c_n\}$ es una mezcla de otras dos secuencias de números naturales $A = \{a_1, a_2, \ldots, a_n\}$ y $B = \{b_1, b_2, \ldots, b_n\}$ si y solo si para cada $1 \leq i \leq n$ se verifica que $c_i = a_i$ o $c_i = b_i$. El valor $\sum_{i=1}^{n} c_i$ se denomina la suma de la mezcla.

El problema de la suma de la mezcla se define de la siguiente manera: dadas dos secuencias de números naturales $A = \{a_1, a_2, \ldots, a_n\}$ y $B = \{b_1, b_2, \ldots, b_n\}$ y dado un número natural m, determinar si existe una mezcla C de A y B tal que $\sum_{i=1}^{n} c_i = m$. Si dicha mezcla existe, debe mostrarse. En caso contrario se debe informar que no existe.
Supongamos que n es potencia de 2. Una estrategia divide y vencerás para este problema es la siguiente.

- Divida la secuencia $A = \{a_1, a_2, \ldots, a_n\}$ en dos secuencias de tamaño $n/2$: $A_1 = \{a_1, a_2, \ldots, a_{n/2}\}$, $A_2 = \{a_{(n/2)+1}, a_{(n/2)+2}, \ldots, a_n\}$. Haga lo mismo con B, dando lugar a otras dos secuencias: $B_1 = \{b_1, b_2, \ldots, b_{n/2}\}$, $B_2 = \{b_{(n/2)+1}, b_{(n/2)+2}, \ldots, b_n\}$.
- Ahora, para cada entero m' con $0 \leq m' \leq m$ se resuelve de manera recursiva
  1. una instancia más pequeña del problema utilizando las secuencias A_1, B_1 y suma objetivo m'
  2. una instancia más pequeña del problema utilizando las secuencias A_2, B_2 y suma objetivo $m - m'$
- Si para algún m' particular existen soluciones C_1, C_2 para los dos problemas anteriores, entonces se puede obtener la solución del problema original concatenando C_1 con C_2 para dar lugar al C buscado. Si no existe tal m' entonces no hay solución para el problema de partida.

3.1. Tareas

- A partir de la descripción anterior, implemente el algoritmo.
- Muestre ejemplos de uso. El programa debe recibir como parámetros el tamaño de las secuencias n, el valor objetivo m y un valor max. Las secuencias A, B se generarán de forma aleatoria con $A[i], B[i] \in [1, \ldots, max]$.
- Calcule la eficiencia teórica de su implementación.

4. Comparación de preferencias de usuarios

En este apartado se deben diseñar dos algoritmos para comparar las preferencias de dos usuarios sobre un asunto particular. Por ejemplo, sobre una lista de peliculas, de discos, libros, etc. Estas preferencias establecen un orden sobre el conjunto de elementos al que denominaremos "ranking".

Asumiremos que el primero de los rankings es una lista ordenada de enteros de 1 a n. El segundo ranking es una permutación $a_1, a_2, \ldots, a_n$ de los enteros entre 1 y n (otro orden para los discos).

La idea básica detrás de la comparación es asumir que dos rankings serán parecidos si hay pocos intercambios.

El segundo ranking tiene un intercambio si existen indices i, j tales que $i < j$ pero $a_i > a_j$. El número de intercambios s será una medida de la diferencia entre los rankings. En otras palabras, si s es pequeño, entonces los correspondientes usuarios tienen gustos similares.

Por ejemplo, supongamos dos usuarios que establecen sus preferencias sobre un conjunto de discos

	Usuario 1	Usuario 2
Dark side of the moon (Pink Floyd)	1	1
Wish (The Cure)	2	3
Rattle and Hum (U2)	3	4
The Joshua Tree (U2)	4	2
Nevermind(Nirvana)	5	5

En este caso $s = 2$, puesto que existen dos intercambios en el usuario 2: $\{3, 2\}, \{4, 2\}$

4.1. Algoritmo básico

Diseñe un algoritmo de orden $O(n^2)$ para contar el número de intercambios (deben probarse todas las combinaciones i, j, con $i < j$).

4.2. Algoritmo Divide y Vencerás

Una estrategia divide y vencerás para este problema debe seguir los siguientes pasos.

1. Dividir la lista del usuario 2 en dos mitades m_1, m_2
2. De manera recursiva, contar los intercambios en cada mitad obteniendo valores s_1, s_2
3. Contar los intercambios donde $a_i \in m_1$ y $a_j \in m_2$, obteniendo s_3.
4. Retornar $s_1 + s_2 + s_3$

El caso base para la recursión puede ser cuando la lista tenga tamaño 1 o 2. En tal caso, contar los intercambios es trivial.

En este problema, la fase más compleja en este proceso es la combinación. Sin embargo, la tarea se puede simplificar si ambas mitades se encuentran ordenadas.

sin ordenar	$m_1 = \{1, 5, 4, 8, 10, 2\}$	$m_2 = \{6, 9, 12, 11, 3, 7\}$
ordenadas	$m'_1 = \{1, 2, 4, 5, 8, 10\}$	$m'_2 = \{3, 6, 7, 9, 11, 12\}$

A partir de este punto

- Imagine que dispone de dos punteros p_1, p_2 que apuntan al principio de las listas m'_1, m'_2 respectivamente. Intente deducir como debería mover esos punteros para obtener el número de intercambios en tiempo lineal. ¿ Cuántos intercambios existen si $m'_2[p_2] < m'_1[p_1]$?
- No es necesario aplicar un algoritmo de ordenación en cada paso. Considere como podría obtener una lista ordenada a partir de dos listas que ya están ordenadas. Aplique dicha idea cuando termina cada paso recursivo.

4.3. Tareas

- Implemente cada algoritmo
- Realice el análisis de eficiencia teórico
- Compare empiricamente los dos algoritmos.

Los programas deben recibir como parámetro el tamaño del vector n y un valor k. Este valor se utilizara para generar la permutación correspondiente al usuario 2 de la siguiente manera.

Sea $V[i] = i$, con $i = 1 \ldots n$. Ahora se generan k intercambios de la siguiente manera:

```
for l = 1 to k
    seleccionar aleatoriamente un indice i
    seleccionar aleatoriamente un indice j ≠ i
    intercambiar V[i] con V[j]
```

5. Documentación

Escriba una memoria con la descripción detallada de las tareas realizadas.

Además, para las tareas 4 y 3, comente e incluya en la memoria las partes relevantes del código fuente.

Sea cuidadoso en la redacción del informe y no olvide incluir en la portada, el nombre, apellidos y dni de todos los integrantes del grupo.

RECUERDE: las instrucciones de este guión se complementan con los comentarios y sugerencias realizadas en las horas de práctica.