FINAL STANDINGS

Sara Chica, Rodrigo Gualtero

15 de diciembre, 2012

Índice

1. Introducción **1**

2. Definición del problema **2**
2.1. Entrada 3
2.2. Salida 3

3. Modelamiento matemático **3**

4. Planteamiento de la Solución **4**

5. Conclusiones **4**

1. Introducción

Este es un problema de TIMUS, identificado con el código *1100*, en el cual se desea generar los resultados finales de acuerdo a la cantidad de problemas que resuelvan en una competencia que se obtenian anteriormente por medio de otro software que lo hacía más lento.
Como ahora hay más equipos se desea realizar un nuevo programa que permita obtenerlos con mayor rapidez.
Esto se realiza ordenando los resultados en orden descendente de acuerdo a la cantidad de problemas resueltos.
Un ejemplo de como deberían quedar organizados es el siguiente:

$$1 \Rightarrow 2$$

$$16 \Rightarrow 3$$

$$11 \Rightarrow 2$$

$$20 \Rightarrow 3$$

$$3 \Rightarrow 5$$

$$26 \Rightarrow 4$$

$$7 \Rightarrow 1$$

$$22 \Rightarrow 4$$

Ejemplo 1.1: Resultados Sin ordenar.

$$3 \Rightarrow 5$$

$$26 \Rightarrow 4$$

$$22 \Rightarrow 4$$

$$16 \Rightarrow 3$$

$$20 \Rightarrow 3$$

$$1 \Rightarrow 2$$

$$11 \Rightarrow 2$$

$$7 \Rightarrow 1$$

Ejemplo 1.2: Resultados Ordenados.

2. Definición del problema

Este problema busca ordenar en forma descendente un grupo de resultados obtenidos en una competencia en la que cada equipo debe resolver problemas y se mide en la cantidad de problemas que resuelven.

2.1. Entrada

En la primera linea entra un valor que determina el número equipos.
Seguido a esto entran el número asignado a cada uno de los equipos(ID), junto con la cantidad de problemas resueltos(M).
Pueden haber a lo mucho 150000 equipos; y cada uno debe resolver entre 1 a 100 problemas.

2.2. Salida

Se debe imprimir los resultados dados en la entrada ordenados en forma decendente, de la forma ID M.

3. Modelamiento matemático

Se sabe que los resultados están organizados si:

$$Ordenado \equiv (\forall i, j | 0 < i < j < 15000 : M_{i+1} \leq M_j)$$

Para ello existen varios algoritmos de ordenamiento, los cuales permiten organizar una lista en una sencuencia dada, de menor a mayor o viceversa.
Algunos de ellos son:
Ordenamiento por Burbuja (Bubble Sort): Consiste en organizar el vector revisando cada elemento de la lista y comparándolo con el siguiente, de esta forma si están en el orden equivocado se deben intercambiar.
Ordenamiento por Mezcla (Merge Sort): Consiste en dividir la lista e ir ordenando cada sublista recursivamente, así cuando cada sublista esté ordenada se mezclan en una sola ya ordenada.
Ordenamiento Rápido (Quick Sort): Se basa en dividir y conquistar, así pues, coge un elemento como pivote y a partir de este resitua los otros elementos. Este proceso se repite recursivamente hasta tener como resultado toda la lista ordenada.
Ordenamiento estable (Stable Sort): Consiste en ordenar un vector de acuerdo a uno de los componentes que contega un valor mayor; sin embargo este conserva el orden relativo de los elementos con valores equivalentes.

4. Planteamiento de la Solución

Para resolver este problema se utiliza Stable Sort; este ya se encuentra implementado en varios lenguajes de programación.
Stable Sort utiliza un boleano que indica la forma en que se desea ordenar (en caso de no tenerlo se usa por default en forma ascendente).
En el ejemplo 1 se organiza de acuerdo al segundo elemento.

5. Conclusiones

1. Conocer los algoritmos de ordenamiento es importante para optimizar algunos otros algoritmos como los de búsqueda y fusión; ya que estos pueden requerir listas ordenadas para tener una ejecución rápida.

2. Stable Sort es un buen método de ordenamiento cuando se tienen varios componentes en una estructura y se debe ordenar por medio de alguno de ellos.