A. PEREDA B.*
R. CONTRERAS A.+

RESUMEN

Este trabajo trata de un algoritmo de ordenación, el cual se ha demostrado como bastante eficiente en aplicaciones en microcomputadores. Se desarrolla un análisis de su complejidad.

ABSTRACT

This work deals with a sorting algorithm, which has shown to be very efficient in microcomputers applications. A complexity analysis of the algorithm is developed.

* D.Sc. Computación (PUC/RJ, 1980); teoría de computación, estructuras de datos, ingeniería de software, Facultad de Ingeniería, Universidad de Concepción, Ciudad Universitaria, Concepción, Chile. Teléfono 24985 Anexo 2305.

+ M.Sc. Informática (PUC/RJ, 1983); ingeniería de software, estructuras de datos, lenguajes de programación, Facultad de Ingeniería, Universidad de Concepción, Ciudad Universitaria, Concepción, Chile. Teléfono 24985 Anexo 2305.

INTRODUCCION

El problema del costo del software, viene siendo atacado con diferentes enfoques, y variado éxito, por un sinnúmero de especialistas [2,8]. Creemos, sin embargo, que una de las tantas causas que inciden en este problema, es la utilización de técnicas (a veces) y algoritmos (puntualmente) inadecuados. Pese a existir una concepción filosófica marcadamente diferente pone aplicaciones enfocadas a microcomputadores y a computadores de gran porte, gran parte del software utilizado en los primeros es una simple adaptación de software desarrollado para máquinas grandes. Así, n o siempre el resultado de su utilización incide en un comportamiento eficiente.
En muchas aplicaciones, el tamaño de tablas a ordenar es pequeño (típicamente menos de 5.000 elementos), y es en este rango donde el algoritmo presentado ofrece ventajas respecto de los métodos tradicionales. Las pruebas iniciales demostraron la eficiencia de este algoritmo, lo que llevó a la realización del análisis que se mostrará a continuación.

EL ALGORITMO

Considerando como importantísimo el factor portabilidad, se eligió como lenguaje de implementación a PASCAL para la realización de las pruebas.
El algoritmo es el siguiente:

```
    procedure main
    procedure  asort (j,n);
1   begin
2      repeat  l ← J;
3      for  i ← j + l  to  n  do
4         if  T [ j ] > T [ i ]  then begin
5                                      l ← l + 1
6                                      T [ l ] ↔T[ i ]
7                                           end;
8      if  l ≠ j
9          then begin
10                    T [ l ] ↔ T[ j ];
11                    if l + 1 < n then  asort (l + 1, n);
12                    n ← l - 1
13                    end
14         else  j ← j + 1
15     until  j ≥  n
16    end
17  begin
18     asort (1 , n)       {n : tamaño de la tabla a ordenar}
19  end,   {main}
```

ANALISIS

Para llevar a cabo el análisis de complejidad del algoritmo asort, se determinará el número de veces que es ejecutado cada comando, al someter a ordenación una tabla de enteros de tamaño n.

Esto se consigue introduciendo contadores en la versión recursiva original, probando esta versión así modificada con las n! combinaciones posibles de los n primeros enteros positivos (TABLA 1).

n	n !	comando 3	comando 9	comando 14	Nº de llamadas recursivas
2	2	2	1	1	0
3	6	16	6	4	0
4	24	116	36	20	6
5	120	888	240	120	60
6	720	7416	1800	840	540
7	5040	67968	25120	6720	5040

TABLA 1

Por simple observación, se puede detectar que el número de veces que es ejecutado el comando 9 (rama then de if $l \neq j$) para el total de las n! tablas de tamaño n es $\frac{n!(n-1)}{2}$. De la misma manera se obtiene que la ejecución del comando 14 (else $j \leftarrow j+1$) se realiza $\frac{(n+1)!}{3!}$

Luego, el total de veces que se ejecuta el comando 8, al ordenar las n! tablas de tamaño n, es $\frac{n!\,(2n-1)}{3}$; esto da una media de ejecución de este comando de $\frac{(2n-1)}{3}$ veces por tabla.

En forma similar, se llega a que el número total de llamadas recursivas a asort es de $\frac{n!(n-3)}{4}$, para las n! tablas de tamaño n, correspondiendo una media de $\frac{(n-3)}{4}$ llamadas por tabla.

Si se desglosa el número de llamadas recursivas, según el tamaño de la tabla llamada, se obtiene el resultado mostrado en TABLA 2.

n	2	3	4	5	6	Total
4	6					6
5	36	24				60
6	252	168	120			540
7	2016	1344	960	720		5040

De esta tabla se puede concluir que el número de llamadas recursivas (NR) de las n! tablas posibles de tamaño n es:

$$n! * NR_n = (n+1)! \sum_{i=2}^{n-2} \frac{1}{(i+2)(i+3)}$$

En media, el número de llamadas recursivas realizadas durante la ordenación de una tabla de tamaño n es:

$$NR_n = (n+1) \sum_{i=2}^{n-2} \frac{1}{(i+2)(i+3)} \qquad n \geq 4$$

con $NR_3 = 0$, $NR_2 = 0$;

pero

$$\sum_{i=2}^{n-2} \frac{1}{(i+2)(i+3)} = \frac{(n-3)}{4(n+1)}$$

luego, este resultado coincide con el anterior obtenido a partir de la observación de valores totales.

Respecto al número de comparaciones de clave (if T [j] > T [i]...), se tiene el desglose mostrado en TABLA 3.

TABLA 3

tamaño / iteración n	1	2	3	4	5	6	Total
3	4	6					16
4	20	12	24				116
5	120	72	48	120			888
6	840	504	336	240	720		7416
7	6720	4032	2688	1920	1440	5040	67968

Esto da un total de comparaciones (C_n) para n! permutaciones de:

$$n!\ C_n = 2 \cdot (n+1)\ ! \sum_{i=2}^{n} \frac{(n-i)}{(n-i+2)\ (n-i+3)} + n!\ (n-1)$$

En media, el número de comparaciones realizadas al ordenar una tabla de tamaño n es

$$E\left[C_n\right] = 2\ (n+1) \sum_{i=2}^{n} \frac{(n-i)}{(n-i+2)\ (n-i+3)} + n - 1$$

Para determinar el valor de $E\left[C_n\right]$, se debe calcular el valor de la sumatoria

$$\sum_{i=2}^{n} \frac{(n-i)}{(n-i+2)\ (n-i+3)}$$

que se puede descomponer en:

$$= \sum_{i=2}^{n} \frac{n}{(n-i+2)\ (n-i+3)} - \sum_{i=2}^{n} \frac{i}{(n-i+2)\ (n-i+3)}$$

para la primera sumatoria

se tiene que

$$\sum_{i=2}^{n} \frac{n}{(n-i+2)\ (n-i+3)} = \frac{n\ (n-1)}{2\ (n+1)}$$

para calcular la segunda sumatoria hacemos $j = n-i+2$, lo que implica $i = n-j+2$, obteniendo:

$$\sum_{i=2}^{n} \frac{i}{(n-i+2)\ (n-i+3)} = \sum_{j=n}^{2} \frac{(n-j+2)}{j(j+1)}$$

luego

$$\sum_{i=2}^{n} \frac{(n-i)}{(n-i+2)\,(n-i+3)} = \frac{n\,(n-1)}{2\,(n+1)} - \sum_{j=n}^{2} \frac{(n-j+2)}{j\,(j+1)}$$

$$= \frac{n\,(n-1)}{2\,(n+1)} - (n+2) \sum_{j=n}^{2} \frac{1}{j\,(j+1)} + \sum_{j=n}^{2} \frac{1}{(j+1)}$$

$$= \frac{n\,(n-1)}{2\,(n+1)} - \frac{(n+2)\,(n-1)}{2\,(n+1)} + \sum_{i=3}^{n+1} \frac{1}{i}$$

$$= \frac{-2\,(n-1)}{2\,(n+1)} + \sum_{i=3}^{n+1} \frac{1}{i} = -\frac{(n-1)}{(n+1)} + \sum_{i=3}^{n+1} \frac{1}{i}$$

pero $\sum_{i=3}^{n+1} \frac{1}{i} = \gamma - 1.5 + \ln\,(n+1) + \frac{1}{2(n+1)} -$

$$\sum_{k=2}^{\infty} \frac{A_k}{(n+1)\,(n+2)\ldots(n+k+2)}$$

con $A_k = \frac{1}{k} \int_0^1 x(1-x)\,(2-x)\ldots(k-1-x)\,dx$

siendo γ la constante de Euler.

Luego:

$$\sum_{i=2}^{n} \frac{(n-i)}{(n-i+2)\,(n-i+3)} = \frac{(n-1)}{(n+1)} + \gamma - 1.5 + \ln\,(n+1) + \frac{1}{2(n+1)}$$

$$- \sum_{k=2}^{\infty} \frac{A_k}{(n+1)\,(n+2)\ldots(n+k+2)}$$

Entonces

$$E\,[\,C_n\,] = 2(n+1)\Big[-\frac{(n-1)}{(n+1)} + \gamma - 1.5 + \ln\,(n+1) + \frac{1}{2(n+1)} -$$

$$\sum_{k=2}^{\infty} \frac{A_k}{(n+1)\ldots(n+k+2)} \Big] + n - 1$$

$$E\,[\,C_n\,] = -\,2(n-1) + 2\,(\gamma - 1.5)(n+1) + 2(n+1)\ln\,(n+1) + 1 - 2(n+1)*$$

$$\sum_{k=2}^{\infty} \frac{A_k}{(n+1)\ldots(n+k+2)} + n - 1$$

y finalmente

$$E\,[\,C_n\,] = 2(n+1)\,\ln\,(n+1) + 2\gamma - 1 + (2\gamma-4)n - 2(n+1)\sum_{k=2}^{\infty}\frac{A_k}{(n+1)\ldots(n+k+2)}$$

El número de comparaciones, en el caso peor (número máximo de comparaciones para ordenar tabla de tamaño n) es $\frac{1}{2}\,n\,(n-1)$, y en el caso mejor (número mínimo de comparaciones para ordenar tabla de tamaño n) $2(n-2)$.

Es decir,

$$2(n-2) \leq C_n \leq \frac{1}{2}\,n\,(n-1)$$

En lo que se refiere al número de trueques realizados, se deben considerar dos factores, lo trueques $T\,[\,1\,] \leftrightarrow T\,[\,i\,]$ (comando 6) y los trueques $T\,[\,1\,] \leftrightarrow T\,[\,j\,]$ (comando 10).

En el segundo caso ($T\,[\,1\,] \leftrightarrow T\,[\,j\,]$), se sabe que, en media, para ordenar una tabla de tamaño n, se realizan $\frac{(n-1)}{2}$ trueques (número de veces que se ejecuta la rama then del if $1 \neq j$).

Para el caso del trueque $T\,[\,1\,] \leftrightarrow T\,[\,i\,]$, elnúmero de veces que este trueque es realizado se puede calcular, observando que el comando 6 se ejecuta (en media) la mitad de las veces que se realiza la comparación de claves ($T\,[\,j\,] > T\,[\,i\,]$), luego, este número es $(n+1)\,\ln\,(n+1) + \gamma - 0.5 + (\gamma-2)n - (n+1)\sum_{k=2}^{\infty}\frac{A_k}{(n+1)\ldots(n+k+2)}$

Entonces, el número total de trueques realizados(en media), al ordenar una tabla de tamaño n es la suma de este valor anterior con $\frac{(n-1)}{2}$; es decir

$$E\,[\,T_n\,] = (n+1)\,\ln\,(n+1) + \gamma - 1 + (\gamma - 1.5)n - (n+1)\sum_{k=2}^{\infty}\frac{A_k}{(n+1)\ldots(n+k+2)}$$

CONCLUSIONES

El algoritmo presentado, ofrece algunas ventajas, en cuanto al tiempo de ejecución, si se le compara, para tablas de menos de 5000 elementos, con una versión mejorada del QUICKSORT [7]. Se realizaron pruebas de comparación de tiempos, utilizando un computador VAX 11/780, mostrándose a continuación algunos de los resultados obtenidos (*)

(*) El tiempo de CPU está medido en milisegundos y las tablas fueron generadas utilizando una rutina random construida según la estrategia propuesta por Sedgewick [8]

n sort	500	1000	1500	2000	4000	4500
	80	180	270	430	960	1090
asort	60	180	270	370	1150	1120
	90	160	280	410	970	1120
	100	230	350	490	1120	1260
Quicksort	110	200	340	470	1000	980
	100	260	320	420	910	1160

TABLA 4

La complejidad del algoritmo propuesto, O(n log n), para el comportamiento promedio, y los tiempos mostrados en la tabla anterior además de su simplicidad de implementación, lo hacen apropiado para su utilización en microcomputadores.

BIBLIOGRAFIA

[1] Gonnet, G.H. "Handbook of Algorithms and Data Structures" Addison-Wesley P.C. (1984)

[2] Horowitz, E., Sahni, S. "Fundamentals of Computer Alrorithms" Computer Science Press. (1978)

[3] Knuth, D. "The Art of Computer Programming" Vol 1. 2nd Ed. Addison-Wesley P.C. (1975)

[4] Knuth, D. "The Art of Computer Programming" Vol 3. Addison-Wesley P.C. (1973)

[5] Larsen, H.D. "Rinehart Mathematical Tables, formulas and curves" Holt, Rinehart and Winston. (1960)

[6] Ryshik, Gradstein "Tables" VEB Deutscher Verlag der Wissenschaften, DDR. (1963)

[7] Sedgewik, R. "Quicksort" Computer Science Dept. Stanford U. Technical Report STAN-CS-75-492. (1975)

[8] Sedgewik, R. "Algorithms" Addison-Wesley P.C. (1983)