UNIVERSIDAD
DE GRANADA

Análisis de eficiencia de algoritmos

Algorítmica. Práctica 1

Jose Alberto Hoces Castro
Javier Gómez López
Moya Martín Castaño

Contenidos

Introducción

Análisis de eficiencia de algoritmos

- **Análisis de la eficiencia teórica:** estudio de la complejidad teórica de algoritmos.
- **Análisis de la eficiencia empírica:** ejecución y medición de tiempos de ejecución de los algoritmos estudiados.
- **Análisis de la eficiencia híbrida:** obtención de las constantes ocultas.

Cálculo de la eficiencia teórica

Consiste en analizar sobre el papel el peor tiempo de ejecución posible en un algoritmo para decidir en qué clase de funciones en notación $\mathcal{O}$ se encuentra

Cálculo de la eficiencia empírica

Ejecución de los algortimos en distintos agentes tecnológicos, calculando su tiempo de ejecución con la librería `<chrono>`.

Cálculo de la eficiencia híbrida

Obtención de las constantes ocultas a través de gnuplot.

Análisis de los algoritmos propuestos

Algoritmos trabajados

Se ha realizado un análisis de los siguientes algoritmos:

1. Algoritmo de Inserción
2. Algoritmo de Selección
3. Algoritmo de Quicksort
4. Algoritmo de Heapsort
5. Algoritmo de Floyd
6. Algoritmos de las torres de Hanoi

Floyd

El código del algoritmo de Floyd es el siguiente:

```
void Floyd(int **M, int dim)
{
    for (int k = 0; k < dim; k++) //O(n)
      for (int i = 0; i < dim;i++) //O(n)
        for (int j = 0; j < dim;j++) //O(n)
          {
        int sum = M[i][k] + M[k][j];
            M[i][j] = (M[i][j] > sum) ? sum : M[i][j]; //O(1)
          }
}           //Total O(n^3)
```

Floyd. Eficiencia teórica

En los comentarios del código observamos el análisis de la función.

Son tres bucles `for` anidados, cada uno $\mathcal{O}(n)$ y por tanto,

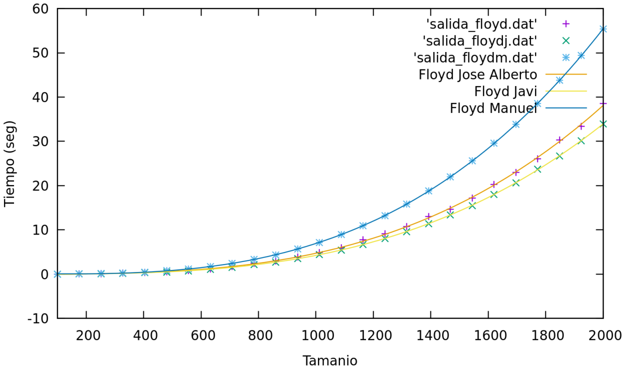
$$T(n) \in \mathcal{O}(n^3)$$

Floyd. Eficiencia empírica

Intel Core i7-6700 3.40 GHz		i5-1095G1 1.00 GHz		Ordenador Moya	
Elementos (n)	**Tiempo (s)**	**Elementos (n)**	**Tiempo (s)**	**Elementos (n)**	**Tiempo (s)**
176	0.0244106	176	0.0274773	176	0.038495
252	0.0721776	252	0.0995705	252	0.111472
328	0.155828	328	0.20657	328	0.244523
404	0.288165	404	0.307902	404	0.45528
480	0.465947	480	0.51806	480	0.761621
556	0.724968	556	0.799187	556	1.17395
632	1.09236	632	1.16729	632	1.73408
708	1.54374	708	1.65895	708	2.4355
784	2.13392	784	2.42549	784	3.29426
860	2.67022	860	3.00331	860	4.35444
936	3.52897	936	3.84788	936	5.64407
1012	4.4074	1012	4.84029	1012	7.16827
1088	5.42559	1088	5.97643	1088	8.91362
1164	6.6698	1164	7.78043	1164	10.9311
1240	8.06967	1240	9.08228	1240	13.2386
1316	9.55022	1316	10.7251	1316	15.8513
1392	11.4197	1392	12.9933	1392	18.7744
1468	13.3942	1468	14.6689	1468	21.9844
1544	15.5	1544	17.2185	1544	25.5768
1620	18.0399	1620	20.2626	1620	29.5543
1696	20.5893	1696	22.9733	1696	33.8275
1772	23.6714	1772	26.0557	1772	38.5849
1848	26.7337	1848	30.2843	1848	43.8038
1924	30.1601	1924	33.4252	1924	49.4368
2000	33.9673	2000	38.5217	2000	55.3965

Tabla 1: Experiencia empírica de algoritmo de Floyd sin optimizar

Floyd. Eficiencia híbrida



Floyd. Eficiencia híbrida

- i7-6700 3.4GHz $\longrightarrow T_1(n) = 4.38237 \cdot 10^{-9}x^3 - 4.33753 \cdot 10^{-7}x^2 + 0.000337001x - 0.0504332$
- i5-1095G1 1.00 GHz $\longrightarrow T_2(n) = 5.12922 \cdot 10^{-9}x^3 - 1.11315 \cdot 10^{-6}x^2 + 0.00083571x - 0.134397$
- Ordenador Moya $\longrightarrow T_3(n) = 6.77297 \cdot 10^{-9}x^3 + 5.13099 \cdot 10^{-7}x^2 - 0.000427834x + 0.0714028$

Coeficiente de regresión:

- $T_1(n) \longrightarrow R^2 = 0.00204522$
- $T_2(n) \longrightarrow R^2 = 0.044778$
- $T_3(n) \longrightarrow R^2 = 0.000855184$

Hanoi

El código del algoritmo de las torres de Hanoi es el siguiente:

```
void hanoi (int M, int i, int j)
{
  if (M > 0)
    {
      hanoi(M-1, i, 6-i-j);
      hanoi (M-1, 6-i-j, j);
  }
}
```

Hanoi. Eficiencia teórica

Estamos ante un algoritmo recursivo, cuya ecuación de recurrencia es:

$$T(n) = 2T(n-1) + 1$$

$$(x-2)(x-1) = 0$$

$$T(n) = c_1 \cdot 2^n + c_2$$

Por tanto:

$$T(n) \in \mathcal{O}(2^n)$$

Hanoi. Eficiencia empírica

Intel Core i7-6700 3.40 GHz	
Elementos (n)	**Tiempo (s)**
8	0.00000136207
9	0.00000267907
10	0.00000528653
11	0.0000112702
12	0.0000234959
13	0.0000457819
14	0.0000904406
15	0.000198225
16	0.000439214
17	0.00088158
18	0.00145113
19	0.00253865
20	0.00499491
21	0.0100156
22	0.0209075
23	0.0402523
24	0.0878626
25	0.171153
26	0.339115
27	0.633015
28	1.28649
29	2.60592
30	5.05092
31	10.1126
32	20.301

i5-1095G1 1.00 GHz	
Elementos (n)	**Tiempo (s)**
8	0.0000037376
9	0.00000737613
10	0.0000145867
11	0.0000283526
12	0.0000460821
13	0.0000722887
14	0.000106264
15	0.000213395
16	0.000353459
17	0.000717674
18	0.00142487
19	0.00278949
20	0.00534407
21	0.0101673
22	0.0238254
23	0.0555082
24	0.112827
25	0.207041
26	0.344851
27	0.761311
28	1.41561
29	2.68719
30	5.41493
31	9.82069
32	20.2358

Ordenador Moya	
Elementos (n)	**Tiempo (s)**
8	0.0000017978
9	0.00000348253
10	0.00000737093
11	0.0000137999
12	0.0000274451
13	0.0000548052
14	0.000110116
15	0.000198426
16	0.000427075
17	0.000796963
18	0.00159355
19	0.00321857
20	0.00633508
21	0.012697
22	0.0253476
23	0.0506946
24	0.101314
25	0.202542
26	0.405264
27	0.809707
28	1.6195
29	3.23942
30	6.47798
31	12.9623

Tabla 2: Experiencia empírica de algoritmo de Hanoi sin optimizar

Hanoi. Eficiencia híbrida

INSERTAR GRÁFICA HANOI

Hanoi. Eficiencia híbrida

- i7-6700 3.40GHz $\longrightarrow T_1(n) =$
- i5-1095G1 1.00 GHz $\longrightarrow T_2(n) =$
- Ordenador Moya $\longrightarrow T_3(n) =$

Coeficiente de regresión:

- $T_1(n) \longrightarrow R^2 =$
- $T_2(n) \longrightarrow R^2 =$
- $T_3(n) \longrightarrow R^2 =$

Conclusiones

Conclusiones

El análisis híbrido nos confirma nuestro análisis teórico observando el coeficiente de regresión.

Lo que más influye en el tiempo es el orden de eficiencia del algoritmo.

Diversidad de agentes tecnológicos: diferentes computadores y arquitecturas da lugar a resultados distintos.