Relacion 1 ALG.pdf

Relacion 1 ALG varios ejercicios Resueltos

2º Algorítmica

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
UGR - Universidad de Granada

8. Dos algoritmos tienen un tiempo de ejecución $T_1(n)= 100n^2$ minutos y $T_2(n)= 5n^3$ minutos. ¿Cuál es el orden de eficiencia de ambos? ¿En qué casos es más eficiente usar un algoritmo u otro? Calcula el tamaño de caso n_0 para el cual, a partir de ese tamaño de problema, uno de los algoritmos es mejor que el otro.

$T_1(n)= 100n^2 \rightarrow O(n^2)$ $\qquad$ $T_2(n)= 5n^3 \rightarrow O(n^3)$

En órdenes de eficiencia, será mejor el algoritmo que tenga mejor orden, en este caso T_1.

$\lim_{n\to\infty} (100n^2/5n^3) = 0$

Los dos van a tardar lo mismo cuando $T_1(n) = T_2(n) \rightarrow 100n^2 = 5n^3 \rightarrow 100n = 5n \rightarrow n=(100/5)=20$

24. El siguiente algoritmo devuelve la longitud de una cadena de caracteres. ¿Cuál/cuáles son los parámetros que definen el tamaño del problema? Analiza y calcula la eficiencia del algoritmo.

```
int longitud(char v[]){
    int i=0;                 ← O(1)
    while(v[i] != '\0')            O(n)    O(n)
        i++;                 ← O(1)
    return i;                ← O(1)
}
```

25. El siguiente algoritmo tiene como entrada un vector de enteros y devuelve en dos vectores separados los números pares y los impares contenidos en el primer vector. ¿Cuál/cuáles son los parámetros que definen el tamaño del problema? Analiza y calcula la eficiencia del algoritmo.

```
void DesgloraParesImpares(const int original[], int nOriginal, int pares[], int & nPares,
                                                        int impares[], int & nImpares){
    int i;
    //Inicialmente pares e impares no tienen componentes útiles
    nPares=0;                                   ← O(1)
    nImpares=0;                                 ← O(1)
    for(i=0; i < nOriginal; i++){
        if(original[i] % 2 == 0){               ← O(1)
            pares[nPares] = original[i];        ← O(1)
            nPares++;                           ← O(1)      O(n)        O(n)
        }else{
            impares[nImpares] = original[i];    ← O(1)
            nImpares++;                         ← O(1)
        }
    }
}
```

26. El siguiente algoritmo comprueba si una cadena de caracteres es palíndromo. ¿Cuál/cuáles son los parámetros que definen el tamaño del problema? Analiza y calcula la eficiencia del algoritmo.

```
bool esPalindromo(char v[]){
    bool pal = true; //Suponemos que lo es                  ← O(1)
    int inicio = 0, fin = strlen(v) -1; //Inicio y fin de la cadena   ← O(1), O(n)
    while((pal) && (inicio < fin) {
        if(v[inicio] != v[fin])       ← O(1)
            pal = false;              ← O(1)
        inicio++;                     ← O(1)
        fin--;                        ← O(1)
    }
    return pal;                       ← O(1)
}
```

O(n/2)

O(n)

n = tamaño de v

Máx{O(n), O(n/2)} = O(n)

27. El siguiente algoritmo busca un elemento en un vector ya ordenado. ¿Cuál/cuáles son los parámetros que definen el tamaño del problema? Analiza y calcula la eficiencia del algoritmo.

```
Int Busqueda(int *v, int n, int elem){
    int inicio, fin, centro;
    inicio = 0;                       ← O(1)
    fin = n-1;                        ← O(1)
    centro = (inicio+fin)/2;          ← O(1)
    while((inicio <= fin) && v[centro]!=elem){
        if(elem < v[centro])          ← O(1)
            fin = centro-1;           ← O(1)
        else
            inicio = cento+1;         ← O(1)
    }
    if (inicio > fin)                 ← O(1)
        return -1;                    ← O(1)
    return centro;                    ← O(1)
}
```

$O(\log_2(n))$

$O(\log_2(n))$

31. El siguiente código realiza la ordenación de un vector por el método QuickSort. Calcule la ecuación en recurrencias del algoritmo, para los casos mejor y peor.

```
void QuickSort(int* v, int ini, int fin) {

    int pospivote;
    if(ini < fin){
        pospivote= pivotar(v, ini, fin);
        QuickSort(v, ini, pospivote-1);
        QuickSort(v, pospivote+1, fin);
    }
}
```

N = fin – ini

$$T(n) = \begin{cases} 1 \text{ si } n <= 0 \\ n + T(n/2) + 1 \text{ si } n>0 \text{ o } n + 2T(n/2) \\ \text{caso peor} \qquad \text{caso mejor} \end{cases}$$

```
int pivotar(int *v, int ini, int fin) {
    int i= ini, j= fin, aux, pivote;

    pivote= v[ini];
    do { i++; } while (v[i]<=pivote && i<=fin);
    do { j--; } while (v[j]>pivote);
    while (i<j) {
        aux=v[i]; v[i]=v[j]; v[j]=aux;
        do { i++; } while (v[i]<=pivote && i<=fin);
        do { j--; } while (v[j]>pivote);
    }
    aux=v[ini]; v[ini]=v[j]; v[j]=aux;
    return j;
}
```

35. Calcule la ecuación en recurrencias del código siguiente:

```
int minimo(int v[], int posIni, int posFin) {
	int resul1, resul2;
	if (posIni==posFin)
		return v[posIni];
	else {
		resul1= minimo(v, posIni, (posIni+posFin)/2);
		resul2= minimo(v, (posIni+posFin)/2+1, posFin);
		return (resul1 > resul2) ? resul2 :resul1 ;
	}
}
```

n = posFin – posIni;

$$T(n) = \begin{cases} 1 & \text{si } n = 0 \\ 2T(n/2) & \text{si } n > 0 \end{cases}$$

ELNH → $T(n) = 2T(n/2) + 1$

38. Calcule la eficiencia del siguiente código:

```
void ejemplo3 (int n)
{
  int i, j, x=0, y=0;             ← O(1)

  for (i = 1; i <= n; i++)        ← O(1)
      if (i % 2 == 1)             ← O(1)
      {                                                                O(n²)
         for (j = i; j <= n; j++) ← O(1)   O(n)
              x++;                ← O(1)           O(n)   n(O(n-i) + O(i))
         for (j = 0; j < i; j++)  ← O(1)   O(n)
              y++;                ← O(1)
      }
}
```

49. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n)= 4T(n-1)-4T(n-2)$

$T(n) – 4T(n-1) + 4T(n-2) = 0$ → ELH
$x^n – 4x^{n-1} + 4x^{n-2} = 0$ → $(x^2-4x+4)\ 4x^{n-2} = 0$
$(x^2-4x+4) = 0$ → $x = 2$

$r = 1$, $R_1 = 2$, $M_1 = 2$
$T(n) = c_{10}2^n + c_{11}2^n n$

$O(2^n n)$

50. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n)= 3T(n-1)-3T(n-2)-T(n-3)$

$T(n) – 3T(n-1) + 3T(n-2) + T(n-3) = 0 \rightarrow$ ELH
$x^n – 3x^{n-1} + 3x^{n-2} + x^{n-3}= 0 \rightarrow (x^3 – 3x^2 + 3x -1)\ x^{n-3} = 0$
$(x^3 – 3x^2 + 3x -1) \rightarrow x = 1$

$r =1, R_1 = 1, M_1 = 3$
$T(n) = c_{10}1^n + c_{11}1^n n + c_{12}1^n n^2$

$O(n^2)$

51. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n)= 2T(n-1) + n^2$

$T(n) - 2T(n-1) = n^2 \quad \rightarrow$ ELNH

Parte homogénea	Parte no homogénea
$x^n – 2x^{n-1} = 0$	$n^2 = b_1^n q_1(n)$
$(x-2)\ x^{n-1} = 0$	$b_1 = 1, q_1 = n, d_1 = 2$

$P_H(x) = (x-2)(x-1)^3$

$T(n) = c_{10}2^n + c_{20}1^n + c_{21}1^n n + c_{22}1^n n^2$

$O(2^n)$

52. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n)= 2T(n-1) + n + n^2$

$T(n) - 2T(n-1) = n + n^2 \rightarrow$ ELNH

Parte homogénea	Parte no homogénea
$x^n – 2x^{n-1} = 0$	$n + n^2 = b_1^n q_1(n) + b_2^n q_2(n)$
$(x-2)\ x^{n-1} = 0$	$b_1 = 1, q_1 = n, d_1 = 1$
	$b_2 = 1, q_2 = n, d_2 = 2$

$P_H(x) = (x-2)(x-1)^5$
$T(n) = c_{10}2^n + c_{20}1^n + c_{21}1^n n+ c_{22}1^n n^2 + c_{23}1^n n^3 + c_{24}1^n n^4$

$O(n^4)$

54. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n)= T(n-1) + n$

$T(n) – T(n-1) = n \quad \rightarrow$ ELNH

Parte homogénea	Parte no homogénea
$x^n – x^{n-1} = 0$	$n = b_1^n q_1(n)$
$(x-1)\ x^{n-1} = 0$	$b_1 = 1, q_1 = n, d_1 = 1$

$P_H(x) = (x-1)^3$
$T(n) = c_{10}1^n + c_{11}1^n n + c_{12}1^n n^2$

$O(n^2)$

56. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: T(n)= 2T(n-1) + 1

$T(n) – 2T(n-1) = 1$ → ELNH

Parte homogénea	Parte no homogénea
$x^n – 2x^{n-1} = 0$	$1 = b_1^n q_1(n)$
$(x-2)\ x^{n-1} = 0$	$b_1 = 1,\ q_1 = 1,\ d_1 = 0$

$P_H(x) = (x-2)(x-1)$
$T(n) = c_{10}2^n + c_{20}1^n$

$O(2^n)$

57. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: T(n)= 3T(n/2) + n

$T(n) – 3T(n/2) = n$ → ELNH $n = 2^m$

Parte homogénea	Parte no homogénea
$2x^m – 6x^{m-1} = 0$	$2^m = b_1^m q_1(m)$
$(x-3)\ 2x^{m-1} = 0$	$b_1 = 2,\ q_1 = 1,\ d_1 = 0$

$P_H(x) = (x-3)(x-2)$
$T(m) = c_{10}3^m + c_{20}2^m$
$T(n) = c_{10}3^{log2(n)} + c_{20}2^{log2(n)}$

$O(3^{log2(n)})$

59. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: T(n) = 5T(n-1) – 8T(n-2) + 4T(n-3)

$T(n) - 5T(n-1) +8T(n-2) – 4T(n-3) = 0$ → ELH

$x^n – 5x^{n-1} + 8x^{n-2} – 4x^{n-3} = 0$
$(x^3 – 5x^2 + 8x – 4)\ x^{n-3} = 0$ → $x = 2, x = -2, x = 1$

$P_H(x) = (x+2)(x-2)(x-1)$
$T(n) = c_{10}2^n + c_{20}(-2^n) + c_{30}1^n$

$O(2^n)$

60. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n) = 5T(n-1) + 6T(n-2) + 4·3^n$

$T(n) – 5T(n-1) -6T(n-2) = 4·3^n$ → ELNH

Parte homogénea	Parte no homogénea
$x^n – 5x^{n-1} – 6x^{n-2} = 0$	$4·3^n = b_1^n q_1(n)$
$(x^2 – 5x -6)\ x^{n-2} = 0$	$b_1 = 3,\ q_1 = 4,\ d_1 = 0$

$P_H(x) = (x-6)(x+1)$
$T(n) = c_{10}6^n + c_{20}(-1^n) + c_{30}3^n$

$O(6^{n)}$

61. Calcula la eficiencia del algoritmo cuyo tiempo de ejecución viene dado por la siguiente ecuación en recurrencias: $T(n) = T(n/2) \cdot T^2(n/4)$

$T(n) – T(n/2) – T^2(n/4) = 0$ → ELH $U(n) = \log_2(T(n)),\ n = 2^m$

$U(n) - U(n/2) - 2U(n/4) = 0$
$U(2^m) - U(2^{m-1}) - 2U(2^{m-2}) = 0$
$x^m – x^{m-1} – 2x^{m-2} = 0$
$(x^2 – x – 2)\ x^{m-2} = 0$ → $x = (1+\sqrt{5})/2,\ x = (1-\sqrt{5})/2$

$P_H(x) = (x - (1+\sqrt{5})/2)(x - (1 - \sqrt{5})/2)$
$T(m) = c_{10}((1+\sqrt{5})/2)^m + c_{20}((1-\sqrt{5})/2)^m$
$T(n) = c_{10}((1+\sqrt{5})/2)^{\log2(n)} + c_{20}((1-\sqrt{5})/2)^{\log2(n)}$
$T(n) = 2$ ^ $(c_{10}((1+\sqrt{5})/2)^{\log2(n)} + c_{20}((1-\sqrt{5})/2)^{\log2(n)})$

$O($ 2 ^ $(c_{10}((1+\sqrt{5})/2)^{\log2(n)} + c_{20}((1-\sqrt{5})/2)^{\log2(n)}))$