USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Introducción al Análisis de Algoritmos

Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos
Ing. Edgar René Ornélyz
Tutor Esvin González

¿Te gusta programar?

¿Qué es un algoritmo?

Algoritmo

¿Qué es un algoritmo?

Conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante la realización de pasos sucesivos que generen una salida o cumplen un objetivo.

¿Cómo determinar lo ‘bueno’ o ‘malo’ que es un algoritmo?

Desarrollo de algoritmos

¿Qué recursos intervienen en el desarrollo y la ejecución de un algoritmo?

Computacionales

- Procesamiento
- Memoria
- Almacenamiento

No computacionales

- Análisis y diseño
- Implementación

Eficiencia de algoritmos

Existen varios conceptos relacionados con la lógica y la matemática

- Principio de invarianza
- Eficiencia
- Casos de análisis
- Notación de la gran O
- Órdenes de eficiencia
- Ecuación característica
- Matemática útil

Invarianza

Dos implementaciones de un mismo algoritmo no difieren más que en una constante multiplicativa.

Eficiencia

Medida del uso de los recursos computacionales requeridos por la ejecución de un algoritmo en función del tamaño de las entradas.

T(n)

Casos de análisis

Para medir el tiempo de ejecución de un algoritmo se tienen varios tipos o casos de análisis.

- Mejor caso
- Peor caso
- Caso promedio
- Análisis probabilístico
- Análisis amortizado

Comparando algoritmos

Algoritmo 1		Algoritmo 2	
n	**T(n) microsegundos**	**n**	**T(n) microsegundos**
10	3	10	5
20	3	20	12
30	4	30	15
100	15	100	35
1000	200	1000	200

Notaciones asintóticas

Estudian el comportamiento de un algoritmo cuando el tamaño de sus entradas es lo suficientemente grande; ignorando lo que ocurre para entradas pequeñas y obviando factores comunes o constantes.

Orden de eficiencia

Un algoritmo tiene un tiempo de ejecución de ***orden f(n)*** para un ***tipo*** de función ***f***, si existe una constante positiva ***C*** y una implementación del algoritmo capaz de resolver cada caso del problema en un tiempo acotado superiormente por ***C*** * ***f(n)***, donde ***n*** es el tamaño de la entrada del algoritmo.

No interesa tanto determinar la función T(n) de forma exacta, sino más bien interesa saber el orden que tendrá dicha función.

Notación O

La notación O grande (cota superior asintótica) dice que una función $T(n)$ es $O(f(n))$ si existen constantes n_0 y C tales que $T(n) < C * f(n)$ para todo $n > n_0$.

Notación O

La notación O grande dice que la función ***T(n)***, (función que determina la eficiencia de un algoritmo) pertenece al orden ***f(n)*** obviando cosas como las **factores multiplicativos** o las **constantes aditivas**.

```
Alg1 (entrada : n) {
    //Hace algo
}
```

Para el algoritmo Alg1 se ha encontrado que su eficiencia está dada por:

$$T(n) = 5\,n^2 + 38$$

Se puede decir que:

$$O(T(n)) = O(5\,n^2 + 38)$$

$$O(T(n)) = O(5\,n^2) + O(38)$$

$$O(T(n)) = O(5\,n^2) + O(38)$$

$$O(T(n)) = O(5\,n^2) + O(38)$$

$$O(T(n)) = O(n^2)$$

Alg1 es de orden **cuadrático**

Órdenes de eficiencia más comunes

Notación	Nombre	Notación	Nombre
O(1)	Constante	O(n)	Lineal
O(log(n))	Logarítmico	O(n^c)	Polinómico de grado c
O(n log(n))	Cuasilineal	O(n!)	Factorial
O($\sqrt{n}$)	Sublineal	O(c^n)	Exponencial

Propiedades de la notación O grande

Siendo:

- k un número real
- f_1 y f_2 son funciones
- $O(g)$ y $O(h)$ son notación O de orden g y h respectivamente

Podemos afirmar que:

$$O(f_1 + f_2) = O(f_1) + O(f_2)$$

$$O(k\, f_1) = O(f_1)$$

$$O(f_1 + k) = O(f_1)$$

$$f_1 = O(g)$$
$$f_2 = O(h)$$
$$f_1 + f_2 = O(\, max(g, h)\,)$$

$$f_1 = O(g)$$
$$f_2 = O(h)$$
$$f_1 * f_2 = O(g * h)$$

Ejercicio en clase #1

Calcular el orden en notación O grande de las siguientes funciones:

$$T(n) = n(3n^2 + \log(2) + 5n)$$

$$T(n) = (n^2+n)(3n + 20)$$

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

$$T(n) = \log(n) * n^5$$

$$T(n) = (n^2)^3 + 10^7$$

Respuestas ejercicio #1

Las respuestas a los ejercicios propuestos anteriormente son las siguientes:

$O(T(n)) = O(n^3)$

$O(T(n)) = O(n^3)$

$O(T(n)) = O(n^2)$

$O(T(n)) = O(\log(n) * n^5)$

$O(T(n)) = O(n^6)$

Ejercicio en clase #2

Sabiendo que una **operación elemental** es cualquier operación aritmética, lógica o relacional o cualquier asignación simple y que su tiempo de ejecución es de una unidad, determine el orden del siguiente algoritmo.

```
max = arr[0]
for(i = 1 to n) {
    if(arr[ i ] > max) {
        max = arr[ i ]
    }
}
```

/* **arr** es un arreglo que va desde 1 hasta n posiciones */

Respuesta ejercicio #2

t(1) para **max = arr_1**
t(1) para **i = arr_2**
t(1) para la comparación entre **i** y **n**
t(1) para la comparación entre **arr_i** y **max**
t(1) para la asignación **max = arr_i**

Las instrucciones en **naranja** se ejecutan **una única vez**, mientras que las instrucciones en **azul** se ejecutan **n - 1 veces**, sabiendo esto podemos decir que:

T(n) = 2 * t(1) + 3 * (n - 1) * t(1)
T(n) = 3n - 1

```
max = arr[0]
for(i = 1 to n) {
    if(arr[ i ] > max) {
        max = arr[ i ]
    }
}
```

/* Por lo tanto **O(T(n))** es de orden lineal */

¿Dónde?

En el curso de Estructuras de Datos ... ¿dónde utilizamos esto?

- Algoritmos de ordenamiento
- Algoritmos de búsqueda
- Inserción o eliminación en estructuras de datos

Enlaces e información de interés

- MOOC de Git y Github
  - Curso masivo online
  - https://goo.gl/ccTa5P
- Slack
  - El enlace estará disponible durante 30 días
  - https://goo.gl/QdYuGk
- Lecturas para la próxima semana
  - Tipos de datos genéricos
  - Diferencia entre lenguaje, IDE y framework

Referencias

- Análisis y Diseño de Algoritmos: Notación asintótica
  - Dr. Jesús A. González Bernal
  - Ciencias Computacionales INAOE
  - Disponible en: https://goo.gl/WiPGZu
- Análisis y Diseño de Algoritmos: La eficiencia de los algoritmos
  - DECSAI, Universidad de Granada
  - Disponible en: https://goo.gl/R5DA1R
- Material adicional
  - P versus NP https://youtu.be/UR2oDYZ-Sao

Gracias por su atención