UNIVERSIDAD DEL BÍO-BÍO



FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

CAMPUS CONCEPCIÓN

TAREA N°1

ANÁLISIS Y DISEÑO DE ALGORITMOS

|  |  |  |
| --- | --- | --- |
| **Integrantes** | **:** | Iván Sobarzo.  Carlos Ulloa.  Juan Pablo González.  Javier Díaz. |
| **Grupo** | **:** | 6. |
| **Docente** | **:** | Erick Elejalde Sierra. |
| **Asignatura** | **:** | Análisis y Diseño de Algoritmos. |
| **Fecha Entrega** | **:** | 29 de Agosto de 2013. |

**INTRODUCCIÓN**

El procedimiento de algoritmos requiere de la utilización de técnicas para su realización sistemática, de tal forma que se apliquen métodos que vuelvan más fácil su desarrollo y confirmen su funcionamiento; añadiendo que, es importante determinar la eficiencia de dichos algoritmos con el término de estimar su habilidad, productividad y tiempo de respuesta.

El objetivo principal del presente informe es utilizar las técnicas fundamentales de diseño y análisis de algoritmos aprendidas en clases, para comprender la naturaleza de los problemas, independientemente de aspectos de implementación (tanto hardware como software) y resolverlos de forma eficiente.

Específicamente desarrollaremos tres problemas que ilustran la complejidad de espacio y tiempo usado en los algoritmos, demostraremos que estos son correctos y a través de un análisis, los clasificaremos según su complicación en tiempo para el peor caso posible.

Finalmente en el problema tres, mediremos la dificultad del algoritmo dado, para posteriormente diseñar el mejor algoritmo para la solución del problema.

**PROBLEMAS**

1. Considere el diseño VLSI layouts. Considere que este layout está formado por un conjunto de polígonos cuyos lados están siempre en ángulo recto y que definen componentes de diseño de un circuito. Los polígonos son representados por una lista cerrada de puntos. Para efectos de fabricación se requiere que estos polígonos sean descompuestos en rectángulos. Diseñe y analice un algoritmo que fragmente estos polígonos en rectángulos.
2. Dado un procedimiento "caja negra que determina la mediana en tiempo lineal en el peor caso, diseñe un algoritmo que en tiempo lineal resuelva el problema de encontrar el k-ésimo elemento para cualquier valor de k”.
3. Suponga el siguiente algoritmo usado para la evaluación de un polinomio p(x) = an x n + an-1xn-1 +… + a0.

|  |
| --- |
| Algorithm 1: Function Polinomio |
| **Input :** polinomio p, valor x  **Output:** r - resultado de evaluar p(x)   1. r ← a0; 2. x power ← 1; 3. **for** i ← 1 **to** n **do** 4. xpower ← x \* xpower; 5. r ← r + ai \* xpower; 6. **return** r; |

1. ¿Cuántas multiplicaciones y adiciones son hechas en el peor caso?
2. ¿Cuántas son hechas en el caso promedio?
3. ¿Puede tener un algoritmo mejor?

**DESARROLLO**

1. int i=0 1

while (array [i]! = ‘\0’) n

i = i + 1 2

read (int k) 1

Imprimir (array [k]) 1

Orden 1 + n

O(n)

n + 5 ≤ C1 n con n0 = 1

C1= 2

⇒ n + 5 ∈ ⍬(n)

El algoritmo de caja negra no influye, ya que en el mejor de los casos habría que aplicar una búsqueda secuencial, hasta encontrar el valor de la posición de dicho elemento para obtener el tamaño del arreglo y así hacer una búsqueda directa.

1. Respuestas:
2. Multiplicaciones = 2n

Adiciones = n

1. En este algoritmo en particular ( evaluar un polinomio) su ejecución, el número de multiplicaciones y adiciones en cada caso (peor, mejor, promedio).

∴ Multiplicaciones = 2n

Adiciones =n

1. Si

|  |
| --- |
| 1. r ← a0 2. **for**i ← n - 1 **to** 0 **do {** 3. r = r \* x + ai ;} 4. **return** r; |

Ambos algoritmos son del mismo orden de complejidad , pero sus tiempos de ejecución son distintos, este algoritmo ejecuta N multiplicaciones y N adiciones.

En conclusión este algoritmo ejecuta la mitad de tiempo que el algoritmo anterior.

**CONCLUSION**

A partir de lo planteado anteriormente podemos deducir que antes de construir un programa se sugiere elegir un buen algoritmo, entendemos por bueno que utilice pocos recursos, siendo comúnmente lo más importante el tiempo que lleve realizarse y el espacio que utiliza en memoria que requiera. No podemos decir que todos los algoritmos son "casi iguales" y creer en nuestra habilidad como programadores para transformar un mal algoritmo en un producto eficaz. Es asimismo no poder decir que uno confía en la creciente potencia de las máquinas y el desvalorizar las mismas como recurso para todos los problemas que existan.

En el análisis de algoritmos se considera usualmente el caso peor, si bien a veces conviene analizar igualmente el caso mejor y hacer alguna estimación sobre un caso promedio. Para emanciparse de factores de implementación tales como el lenguaje de programación, la habilidad del programador, la máquina soporte, etc. se suele trabajar con un cálculo asintótico que indica cómo se comporta el algoritmo para una gran cantidad de datos. Para problemas pequeños es cierto que casi todos los algoritmos son "parecidos", siendo más importantes otros aspectos como la codificación, legibilidad, etc. Los órdenes de complejidad sólo son importantes para grandes problemas.