Algoritmos y Estructuras de Datos 3 (Período II, 2019)

Instructores: Marcos Villagra — Tarea #1
Auxiliar de cátedra: Julio Mello — Agosto 2, 2019

---

(1) El procedimiento llamado InsertionSort toma como parámetro un arreglo $A[1 \dots n]$ que contiene una secuencia de n enteros que deben ordenarse. El algoritmo ordena los números de entrada en el mismo arreglo.

```
procedure InsertionSort(A)
    for j ← 2 to n do
        key ← A[j]
        // Insertar A[j] en la secuencia ordenada A[1 ... j − 1]
        i ← j − 1
        while i > 0 and A[i] > key do
            A[i + 1] ← A[i]
            i ← i − 1
        A[i + 1] ← key
```

(a) Hacer el análisis de complejidad temporal de InsertionSort.

(b) ¿Es más rápido el ordenamiento de la burbuja o InsertionSort? Argumente su respuesta.

(2) Considere el problema de abajo el cual que se conoce como el problema de búsqueda lineal.

Problema de búsqueda lineal.

ENTRADA	:	una secuencia de n números $A = \langle a_1, a_2, \dots, a_n \rangle$ y un valor ν.
SALIDA	:	un índice i tal que $\nu = A[i]$ o el valor especial `NIL` si ν no aparece en A.

(a) Escriba un pseudocódigo para la búsqueda lineal, que explora la secuencia buscando ν.

(b) Hacer el análisis de complejidad temporal de su algoritmo. ¿Cuánto es su costo asintótico?

(c) ¿Se puede hacer un algoritmo con costo $O(\sqrt{n})$ para el problema de búsqueda lineal? Argumente su respuesta.

(3) Considere ordenar n números almacenados en un arreglo A de la siguiente forma. Primero se encuentra el elemento más pequeño de A y se intercambia con el elemento en $A[1]$. Luego se encuentra el segundo elemento más pequeño de A y se intercambia con $A[2]$. Continue de esta manera para los primeros $n-1$ elementos de A. Este algoritmo se conoce como SelectionSort,

(a) Escriba un pseudocódigo para SelectionSort.

(b) ¿Por qué necesita ejecutarse solo para los primeros $n-1$ elementos, en lugar de los n elementos?

(c) Proporcione en notación asintótica el mejor y el peor de los tiempos de ejecución de SelectionSort. ¿Cuál es la entrada que hace que el algoritmo ejecute el número mínimo de operaciones básicas? ¿Cuál es la entrada que hace que el algoritmo ejecute el número máximo de operaciones básicas?

(4) Sea A un arreglo de n números naturales distintos. Si $i < j$ y $A[i] > A[j]$, entonces el par (i, j) se denomina una *inversión* de A.

(a) Listar las cinco inversiones del arreglo $\langle 2, 3, 8, 6, 1\rangle$.

(b) ¿Cuál es el arreglo con elementos del conjunto $\{1, 2, \ldots, n\}$ que tiene más inversiones? ¿Cuántos tiene? Argumente su respuesta.

(c) ¿Cuál es la relación entre el tiempo de ejecución de InsertionSort y el número de inversiones en el arreglo de entrada? Justifica tu respuesta.

(5) Justifica tu respuesta.

(a) ¿Es $2^{n+1} = O(2^n)$?

(b) ¿Es $2^{2n} = O(2^n)$?

(6) Demuestre por inducción que el i-ésimo número de Fibonacci satisface la igualdad

$$F_i = \frac{\phi^i - \widehat{\phi}^i}{\sqrt{5}},$$

donde $\phi = \frac{1+\sqrt{5}}{2}$ es el número de oro y $\widehat{\phi}$ es su conjugado.

(7) Crecimientos asintóticos relativos. Buscar en las referencias del curso las definiciones de las notaciones asintóticas Ω (omega-grande), o (o-pequeña), ω (omega pequeña) y Θ (theta-grande) e indicar para cada par de expresiones (A, B) en la tabla de abajo si A es O, o, Ω, ω o Θ de B. Suponga que $k \geq 1$, $\epsilon > 0$ y $c > 1$ son constantes. Su respuesta debe ser en forma de tabla con un "`SI`" o un "`NO`" escrito en cada casilla. Presente una breve argumentación de cada una de sus respuestas.

	A	**B**	**O**	**o**	Ω	ω	Θ
a.	$lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$lg(n)$	$lg(n^n)$					