Segundo examen opcional
FUNDAMENTOS DE ANÁLISIS Y DISEÑO DE ALGORITMOS
Grupo 80 - Duración: 1 hora 30 minutos

Carlos Andres Delgado S, Ing *

25 de Junio de 2014

Nombre:_______________________________
Código:________________

Nota: Recuerde colocar los procedimientos debido a que tienen un gran valor en la calificación.

1. Algoritmos de ordenamiento: Análisis y complejidad [**100 puntos**]

1.1. Selección mejor método de ordenamiento. [**30 puntos**]

Se desea listar en orden ascendente los i elementos mayores de un arreglo A.

1. Para cada uno de los métodos a continuación, calcule su complejidad en el peor caso, en función de n e i.. Para cada caso sea muy claro en la forma como presenta sus cálculos. Justifique su respuesta

   a) [**3 puntos**] Ordene los números y liste los i mas grandes.

   b) [**7 puntos**] Construya una cola de prioridad a partir de los números y llame la función que extrae el máximo i veces.

   c) [**8 puntos**] Use un algoritmo para calcular el $n-i+1$-ésimo elemento, úselo como pivote para partición, y ordene el subarreglo que contiene los i mas grandes números.

* carlos.andres.delgado@correounivalle.edu.co

Nombre:______________________________
Código:_______________

2. **[15 puntos]** ¿Cual de los tres es el mejor método?. Justifique con base en las complejidades indicadas previamente

1.2. Algoritmo SelectionSort. Análisis y complejidad [20 puntos]

Considere el siguiente algoritmo de ordenamiento:

```
SelectionSort(A)
for i = 1 to n − 1
    do min_ind ← i
       for j = i + 1 to n
           do if A[j] < A[min_ind]
                 then min_ind ← j
       A[min_ind] ↔ A[i]
```

1. **[12 puntos]** Sea $T(n)$ el numero de comparaciones entre elementos efectuadas por *SelectionSort* para entradas de tamaño n en el peor caso. Cual es el orden de $T(n)$?. Justifique

2. **[8 puntos]** Si Ud. debe escoger entre el *insertionSort* visto en clase y el *SelectionSort* descrito en este punto, ¿cual escogería? Justifique. No utilice mas de cinco (5) lineas.

1.3. Algoritmo BinaryInsertionSort. Análisis y complejidad [35 puntos]

Considere el siguiente algoritmo de ordenamiento:

```
BinaryInsertionSort(A)
    for j ← 2 to length[A]
        do BinaryInsertion(A, 1, j)
```

Donde $BinaryInsertion(A, i, j)$ recibe un arreglo A ordenado entre i y $j-1$, e inserta binariamente $A[j]$ en $A[i..j-1]$ de manera que al terminar $A[i..j]$ esta ordenado:

```
BinaryInsertion(A, i, j)
    if i < j
      then ll ← A[j]
           m ← i + ⌊(j−i)/2⌋
           if A[m] < A[j]
             then BinaryInsertion(A, m + 1, j)
             else for k = j − 1 downto m
                      do A[k + 1] ← A[k]
                  A[m] ← ll
                  BinaryInsertion(A, i, m)
    return
```

1. **[8 puntos]** Describa en la siguiente tabla, el funcionamiento de *BinaryInsertion*, describiendo el valor de cada uno de los valores solicitados en la tabla que encontrara a continuación, para el caso en que se haga un llamado con $A = [1\ 2\ 4\ 10\ 14\ 8\ 7], i = 1, j = 5$.

Nombre:______________________________
Código:_______________

No	A							i	j	ll	m
1	1	2	4	10	14	8	7	1	5		
2											
3											
4											
5											

2. **[15 puntos]** Sea $T(n)$ el numero de comparaciones entre elementos del arreglo efectuadas, en el peor caso, por *BinaryInsertion* para entradas de tamaño n. Describa la ecuación de recurrencia para $T(n)$ y resuelvala. ¿Cual es, entonces, la complejidad de *BinaryInsertionSort* ?

3. **[12 puntos]** Si Ud. debe escoger entre el *QuickSort* visto en clase y el *BinaryInsertionSort* descrito en este punto para ordenar un arreglo de n elementos, cual escogería? Justifique su respuesta. **N**o utilice mas de cinco (5) lineas para ello.

1.4. Ordenamiento en tiempo lineal. [15 puntos]

Describa un algoritmo para ordenar n enteros en el rango de 1 a n^2 en tiempo $\mathcal{O}(n)$.