UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

CI-1221 Estructuras de Datos y Análisis de Algoritmos
II ciclo 2012, grupos 2 y 3

I Examen Parcial

Sábado 8 de setiembre

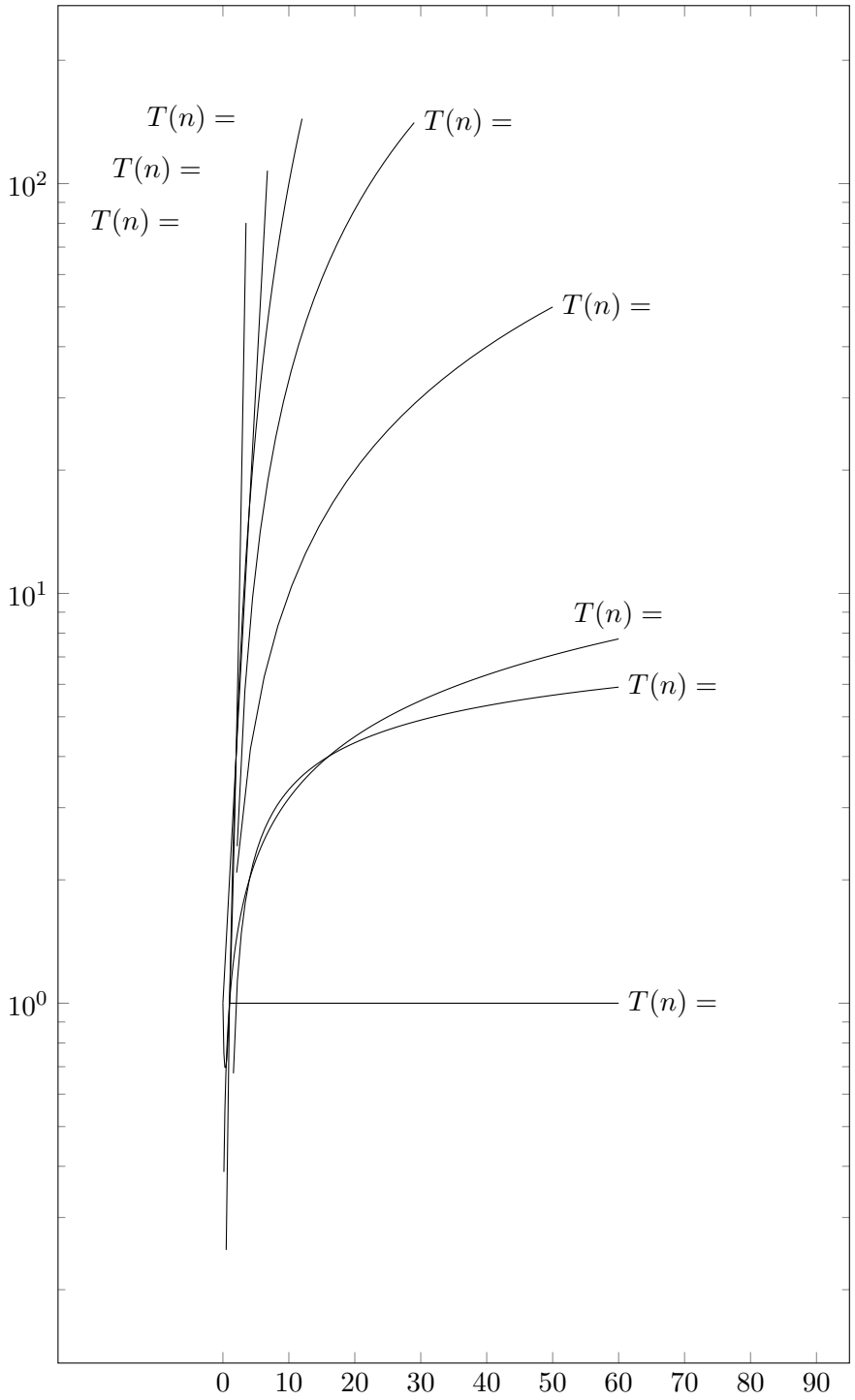
Nombre: _____ Carné: _____

El examen consta de 5 preguntas que suman 100 puntos. Cada pregunta indica el tema tratado y su valor. Si la pregunta tiene subítemes, el puntaje de cada uno de ellos es indicado dentro de los subítemes. Se recomienda echar un vistazo a los temas de las preguntas y a su puntaje antes de resolver el examen, para así distribuir su tiempo y esfuerzo de la mejor manera. Las preguntas se pueden responder en cualquier orden, pero se debe indicar en la tabla mostrada abajo los números de página del cuaderno de examen en las que está cada respuesta. Las hojas del cuaderno de examen deben estar numeradas en la esquina superior externa de cada página. El examen se puede realizar con lápiz o lapicero. *No se permite el uso de dispositivos electrónicos (calculadoras, teléfonos, audífonos, etc.).*

Pregunta	Puntos	Página	Calificación
1. *Matemática básica*	8		
2. *Algoritmos iterativos: correctitud, duración y espacio*	20		
3. *Solución de recurrencias*	15		
4. *Análisis de algoritmos recursivos*	15		
5. *Simulación de ejecución de algoritmos*	42		
Total	100		

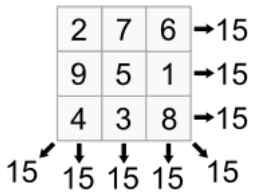
1. *Matemática básica.* [8 pts.]
   Identifique en el gráfico de la derecha cada una de las funciones de la izquierda. (Note que el eje de las ordenadas está en escala logarítmica).

$T(n) = 1$
$T(n) = n$
$T(n) = n^2$
$T(n) = 2^n$
$T(n) = \sqrt{n}$
$T(n) = n^n$
$T(n) = n \lg n$
$T(n) = \lg n$



2. *Algoritmos iterativos: correctitud, duración y espacio.* [20 pts.]

Un cuadrado mágico de orden n es un arreglo de tamaño $n \times n$ que contiene los números 1 a n^2, tal que todas las filas, columnas y diagonales suman lo mismo: $n(n^2+1)/2$. Un ejemplo de un cuadrado mágico de orden tres es el siguiente:



El siguiente algoritmo determina si un arreglo de tamaño $n \times n$ es un cuadrado mágico de orden n o no:

```
1  int esMagico( int A[n][n], int n ) {
2      int contador[n*n];
3      for (int i=0; i<n*n; ++i) contador[i]=0;
4      for (int i=0; i<n; ++i) {
5          for (int j=0; j<n; ++j) {
6              if ( A[i][j]>0 && A[i][j]<=n*n ) {
7                  contador[A[i][j]-1]++;
8              } else {return 0;}
9          }
10     }
11     for (int i=0; i<n*n; ++i) {
12         if (contador[i] != 1) {return 0;}
13     }
14     int sumaFila[n];
15     int sumaColumna[n];
16     int sumaDiagonal[2] = {0};
17     for (int i=0; i<n; ++i) {
18         sumaFila[i] = 0;
19         sumaColumna[i] = 0;
20         for (int j=0; j<n; ++j) {
21             sumaFila[i] += A[i][j];
22             sumaColumna[i] += A[j][i];
23         }
24         sumaDiagonal[0] += A[i][i];
25         sumaDiagonal[1] += A[n-i-1][i];
26     }
27     int sumaEsperada = (1+n*n/2)*n;
28     for (int i=0; i<n; ++i) {
29         if ( sumaFila[i] != sumaEsperada ) {return 0;}
30         if ( sumaColumna[i] != sumaEsperada ) {return 0;}
31     }
32     for (int i=0; i<2; ++i) {
33         if ( sumaDiagonal[i] != sumaEsperada ) {return 0;}
34     }
35     return 1;
36 }
```

a) Demuestre (parcialmente) que el algoritmo es correcto. Para ello identifique un invariante apropiado para el ciclo que inicia en la línea 17 [4 pts.] y muestre como los pasos de

inicialización [2 pts.], mantenimiento [4 pts.] y terminación [2 pts.] permiten inferir la correctitud de lo efectuado en ese ciclo.

b) Encuentre cotas asintóticas ajustadas para el tiempo de ejecución y espacio extra requeridos por el algoritmo en el peor caso. [4 pts c/u.]

3. *Solución de recurrencias.* [15 pts.]
Resuelva las siguientes recurrencias asumiendo que $T(n) = \Theta(1)$ para $n \leq 1$ y $c > 0$.

a) $T(n) = 2\,T(n/\sqrt{2}) + c\,n^2 + c_0$. [5 pts.]

b) $T(n) = 2\,T\,(0{,}75\,n) + c\,(n \lg n)^2 + c_0$. [10 pts.]

4. *Análisis de algoritmos recursivos.* [15 pts.]
Suponga que el algoritmo de ordenamiento por mezcla [*mergesort*] se modifica para que en vez de que se parta un arreglo de tamaño n en dos arreglos de tamaño $n/2$ (para ordenarlos recursivamente), lo parta en tres arreglos de tamaño $n/3$.

a) Calcule el tiempo de ejecución del algoritmo y determine si este algoritmo es mejor que el algoritmo de ordenamiento por mezcla estándar. [7 pts.]

b) Analice cómo se afecta el número de comparaciones requerido por el algoritmo *en el peor caso*: ¿aumenta, disminuye, o permanece igual? En caso de que se afecte, en qué proporción aumenta o disminuye el número de comparaciones. [8 pts.]

5. *Simulación de ejecución de algoritmos.* [42 pts.]
Simule la ejecución de los siguientes algoritmos sobre el arreglo $A = \langle$*ene, feb, mar, abr, may, jun*$\rangle$, mostrando el (los) estado(s) del (los) (sub)arreglos después de cada paso (iteración del ciclo principal o llamado a subrutina del algoritmo). Asuma un criterio de ordenamiento *alfabético*. Después del primer paso incorrecto el resto de pasos no suman puntos. [6 pts. c/ simulación, excepto ordenamiento por montículos: 6 pts. por monticularizar y 6 pts. por ordenar].

a) Ordenamiento por selección.

b) Ordenamiento por inserción.

c) Ordenamiento por mezcla [*mergesort*]. (Si el tamaño del [sub]arreglo es impar, *redondee* el punto medio hacia abajo, es decir, que la parte izquierda del subarreglo sea más pequeña que su parte derecha, y no al revés. Por ejemplo, un arreglo de tamaño 3 debería ser partido en un arreglo de tamaño 1 [a la izquierda] y uno de tamaño 2 [a la derecha]).

d) Ordenamiento rápido [*quicksort*] (usando el último elemento como pivote).

e) Ordenamiento usando montículos [*heapsort*]

f) Ordenamiento por residuos [*radixsort*] (asumiendo que en cada pasada se toma un caracter distinto).