Análisis de Algoritmos

Primer examen parcial

a entregar el lunes 12 de octubre del 2020, 16h00

Instrucciones:

1. Resuelve este examen de forma individual.
2. A enviar en formato *.pdf
3. Después de la entrega del examen tendremos una pequeña entrevista individual. Preveer tiempo entre 16h y 18h.

Opción múltiple (20pts)

1. ¿Cuál es el caso que es verdadero trivialmente al realizar una prueba por inducción? (2pts)

   A. la base
   B. la hipótesis inductiva
   C. la invariante al cíclo
   D. el lema
   E. ninguno de los anteriores

   $T(n) = T(\frac{n}{2}) + O(n)$

2. Un algoritmo recursivo trabaja resolviendo dos problemas de la mitad del tamaño recursivamente con un $overhead$ (tiempo que toma el procesamiento fuera de la llamada recursiva) de tiempo de ejecución lineal. El tiempo de ejecución está descrito de manera más precisa por: (2pts)

   A. $O(\log n)$ B. $O(n)$ C. $O(n \log n)$ D. $O(n^2)$ E. ninguno de los anteriores.

3. ¿Cuál de las siguientes funciones crece más rápido? (3pts)

   A. $n \log n$ B. 2^n C. $\log n$ D. n^2 E. n^{20}

4. ¿Cuál de las siguientes funciones crece más rápido? (3pts)

   A. $n + \log n$ B. $n \log n$ C. $n - \log n$ D. n E. dos o más funciones crecen a la misma velocidad.

5. Junto a cada uno de los cinco siguientes términos, escribe la letra de su definición apropiada. (Nota que hay más definiciones que términos. Lee las definiciones con cuidado. (10pts)

   $\Theta(g(n))$ B
   $\Omega(g(n))$ D
   $w(g(n))$ H
   $O(g(n))$ F
   $o(g(n))$ G

Definiciones:

A. $\{f(n)$: para cualquier par de constantes c_1 y c_2 positivas existe una constante $n_0 > 0$ tal que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0$ $\}$.

B. $\{f(n)$: existen constantes positivas c_1,c_2 y n_0 tal que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para toda $n \geq n_0\}$.

C. $\{f(n)$: existen constantes positivas c y n_0 tal que $0 \leq g(n) \leq cf(n)$ para toda $n \geq n_0\}$.

D. $\{f(n)$: existen constantes positivas c y n_0 tal que $0 \leq cg(n) \leq f(n)$ para toda $n \geq n_0\}$.

E. $\{f(n)$: para cualquier constante positiva $c > 0$ existe una constante $n_0 > 0$ tal que $0 \leq cg(n) \leq f(n)$ para toda $n \geq n_0\}$.

F. $\{f(n)$: existen constantes positivas c y n_0 tal que $0 \leq f(n) \leq cg(n)$ para toda $n \geq n_0\}$.

G. $\{f(n)$: para cualquier constante positiva $c > 0$ existe una constante $n_0 > 0$ tal que $0 \leq f(n) < cg(n)$ para toda $n \geq n_0\}$.

H. Ninguno de los anteriores

Stable Marriage Problem (25pts)

6. Ejecuta el algoritmo de Gale-Shapley en el siguiente conjunto de hombres y mujeres. Considera que los hombres proponen y las mujeres aceptan. Los nombres están abreviados con su primera letra.

Hombres	Preferencias
Adrián	$S > T > U > V$
Bruno	$S > V > U > T$
Carlos	$S > U > T > V$
David	$U > V > T > S$

Mujeres	Preferencias
Susana	$D > A > B > C$
Tania	$A > D > C > B$
Ursula	$C > D > B > A$
Virgina	$A > C > D > B$

7. Determina una lista de cuatro hombres y cuatro mujeres (cada uno ordenando a las cuatro personas del sexo opuesto por preferencia) donde ninguno obtenga su primera opcion, sin importar si son los hombres o las mujeres quienes proponen.

8. Prueba que si los hombres proponen, entonces a lo más, uno de ellos tiene su última opción (suponiendo que todos ordenaron en orden de preferencia al sexo opuesto).

Recurrencias (30pts)

9. Prueba por el método de sustitución que la cota inferior de la siguiente recurrencia es $T(n) = n^2 \lg(n)$:

$$T(n) = 4T(\frac{n}{2}) + n^2$$

.

10. Usa un árbol de recursión para determinar la función de crecimiento asintótico (apretada) para la recurrencia $T(n) = T(n/3) + T(2n/3) + O(n)$. Utiliza el método de sustitución para verificar tu respuesta. (15pts)

11. Resuelve las siguientes recurrencias utilizando cualquier método para encontrar su función (apretada) de crecimiento asintótico con notación Θ. Justifica tus respuestas. (20pts)

    (a) $T(n) = 2T(n/3) + n \log n$

    (b) $T(n) = 7T(n/2) + n^3$

Mergesort (10pts)

El algoritmo de Mergesort visto en clase funciona como sigue para una lista no ordenada U de n números:

- Si $n = 1$, regresa U (ya está ordenada).
- Divide U en dos listas U_1 y U_2 (aproximadamente del mismo tamaño.)
- Ordena recursivamente: $S_1 = Mergesort(U_1)$ y $S_2 = Mergesort(U_2)$
- Regresa la salida: $S = Merge(S_1, S_2)$.

12. Da la relación de recurrencia que describe el tiempo de ejecución de *Mergesort*.

13. Da la cota al tiempo de ejecución del algoritmo *Mergesort* utilizando la notación O. Prueba el resultado de esta cota.

Análisis Probabilístico (15pts)

14. En el problema de contratación de asistente HIRE-ASSISTANT visto en clase; suponiendo que los candidatos se presentan en orden aleatorio, ¿Cuál es la probabilidad de que contrates exactamente una vez? ¿Cuál es la probabilidad de que contrates exactamente n veces?

15. Utiliza variables indicadoras para calcular el valor esperado de la suma de n dados.