Análisis de Algoritmos

Rubén Pérez Palacios - 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

   No entendi la pregunta pero según yo esto es lo que pasa. El adivinar la cota de tu recurrencia yo creo es lo menos trivial, ya que no puedes tener muchas cosas anidadas que parecen que no tienen expresión cerrada pero si lo tienen, o puede ser que no tengas un arbol de recursión completo y todavía se vuelve peor. Dependiendo de cual sea tu recursión puede ser que necesites muchos casos base dependiendo de lq cantidad de terminos en tu recurrencia y a cuales te envie por lo que no es tan trivial. La hipótesis inductiva pues es justamente la cota que propusiste entonces esta se sigue de la cota propuesta, pero demostrarlo puede ser no trivial.

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.

   El tiempo de ejecución no es $O(n)$ o $O(\log n)$ puesto que no son cotas superiores del tiempo del algoritmo descrito anteriormente. Luego como $O(n \log n)$ y $O(n^2)$ son ambas cotas superiores de $T(n)$ y $O(n \log n) \leq O(n^2)$ entonces la respuesta es

   $$O(n \log n).$$

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}

Puesto que $n \in \mathbb{N}$ entonces

$$\log(n) \leq n,$$

por lo tanto

$$n \log(n) \leq n^2,$$

también por ser $n \in \mathbb{N}$ se cumple que

$$n^2 \leq n^2 0$$

Para $n \geq 2$ tenemos

$$1 \leq \log(n),$$

y por lo tanto

$$n \leq n \log(n).$$

Ahora probaremos algo mas general para comparar n^{20} y 2^n. Demostraremos que a^n crece mas rapido que n^b para todo $b \geq 0$ y $a > 1$, demostrando que

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0.$$

Primero demostraremos por inducción para $b \in \mathbb{N}$. Para $b = 0$ se cumple que

$$\lim_{n \to \infty} \frac{1}{a^n} = 0,$$

para el paso inductivo por L'Hopital tenemos que

$$\lim_{n \to \infty} \frac{n^b}{a^n} = \lim_{n \to \infty} \frac{b}{\log(a)} \frac{n^{b-1}}{a^n},$$

por hipotesis de inducción concluimos

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0.$$

Al ser a menor a el techo de b para todo b real entonces podemos acotar el

$$\lim_{n \to \infty} \frac{n^b}{a^n},$$

intercambiando b por su techo y entonces al ser este igual a cero concluimos que

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0$$

para todo $b \geq 0$ y $a > 1$.

Por definición de O concluimos

$$O(n^b) \leq O(a^n).$$

Por lo tanto 2^n es la función que crece mas rapido.

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.

   Como $O(\log n) \leq O(n)$ entonces $O(n + \log n) = O(n)$, luego por lo demostrado anteriormente tenemos que

$$O(n) \subset O(n \log n),$$

   por lo tanto la respuesta es

$$n \log n.$$

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
   $w(g(n))$ E
   $O(g(n))$ 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	$A > C > D > B$

Los matrimonios generados por el algoritmo son

- Adrián y Susana
- Davidi y Virgina
- Carlos y Ursula
- Bruno y Tania

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.

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

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

Cuyo resultado son los matrimonios

- Adrián y Ursula
- Bruno y Virgina
- Carlos y Susana
- David y Tania

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).

Sean $H = \{H_1, \cdots, H_n\}$ los hombres y $M = \{M_1, \cdots, M_n\}$ las mujeres, tales que hay al menos una pareja que un hombre esta casado con la mujer que menos prefiere. Sea d el primer día que un hombre se caso con su úlima opción digamos $H_e, e \in \{1, \cdots, n\}$ es el hombre esta casado con la mujer que menos prefiere digamos M_p. Entonces en los $d-1$ días anteriores el H_e fue rechazado por las otras $n-1$ mujeres luego esto implica que las otras $n-1$ mujeres y también los otros $n-1$ hombres en el día $d-1$ ya estaban casados, también la mujer M_p no se le han propuesto por que de ser así entonces en alguno de los $d-1$ días anteriores ya todos estarían casados y el algoritmo hubiese terminado antes de d lo cual es imposible. Luego como todos los hombres en $d-1$ estan casados y d es el primer día que un hombre se casa con la mujer que menos prefiere entonces no puede existir otro hombre que se case con la mujer que menos prefiere.

Recurrencias (30pts)

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

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

.

Entonces demostraremos por Inducción Matemática, que esto es cierto con $c = 2$.

- **Casos base:**
  - n = 2
    Primero

    $$T(2) = 4T(1) + 2^2 = 8,$$

    entonces

    $$T(2) \leq 2 \cdot 2^2 \log(2).$$

  - n = 3
    Primero

    $$T(3) = 4T(1) + 3^2 = 13,$$

    entonces

    $$T(3) \leq 2 \cdot 3^2 \log(3).$$

- **Hipotesis de inducción:**

$$T(k) \leq cn^2 \log n, \forall k < n$$

- **Paso Inductivo:**
  Por hipotesis de inducción tenemos

  $$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \leq 4c\left\lfloor \frac{n}{2} \right\rfloor^2 \log\left(\left\lfloor \frac{n}{2} \right\rfloor\right),$$

  por la recurrencia obtenemos

  $$T(n) \leq 4c\left\lfloor \frac{n}{2} \right\rfloor^2 \log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n^2,$$

  luego como $\lfloor xy \rfloor \leq xy$ y por monotonía de $\log$ tenemos

  $$T(n) \leq cn^2 \log\left(\frac{n}{2}\right) + n^2,$$

  por propiedades del logaritmo obtenemos

  $$T(n) \leq cn^2 \log(n) - cn^2 \log(2) + n^2 \leq O(n^2 \log(n)).$$

Por Inducción Matemática concluimos que

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

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)

    Okey el problema esta mal puesto que al tener el termino $O(n)$ nuestra recurrencia entonces esto no impide acotar por abajo esta función. Por lo que pueden pasar dos cosas, que nos pregunten la función de crecimiento asintótico en O o que nos den un termino que pueda ser acotado por encima y por abajo.

    Insertar arbol(no supe como ponerlo bonito).

    Primero veremos que pasa si nuestro termino $O(n) = n$. La suma de cada nivel da n. Si tomamos siempre los subproblemas $T(N/3)$ podemos ver que el tamaño del camino mas corto de la raiz a una hoja es $\log_3(n)$. Por lo que

    $$T(n) \geq n \log_3(n),$$

    como

    $$\log_3(n) = \frac{\log(n)}{\log(3)},$$

    por definición de Ω tenemos que

    $$\Omega(n \log_3(n)) = \Omega(n \log(n)),$$

    por lo tanto

    $$T(n) = \Omega(n \log n).$$

    Ahora regresando al enunciado original demostraremos que $T(n) = O(n \log n)$, sea d la constante escondida en el termino de la recursión $O(n)$ entonces tenemos

    $$T(n) \leq T(n/3) + T(2n/3) + dn.$$

    Entonces demostraremos por Inducción Matemática, que esto es cierto con $c = 1000d$, para facilitar los calculos $T(1) = T(2) = 1$.

    - **Casos base:**
      Podemos ver que para $n \in 3, 4, 5, 6, 7, 8$ se cumple que

      $$T(n) \leq cn \log n$$

    - **Hipotesis de inducción:**

      $$T(k) \leq cn \log n, \forall k < n$$

    - **Paso Inductivo:**
      Por hipotesis de inducción tenemos

      $$T\left(\left\lfloor \frac{n}{3} \right\rfloor\right) \leq c\left\lfloor \frac{n}{3} \right\rfloor \log\left(\left\lfloor \frac{n}{3} \right\rfloor\right),$$

      y que

$$T\left(\left\lfloor\frac{2n}{3}\right\rfloor\right) \leq c\left\lfloor\frac{2n}{3}\right\rfloor \log\left(\left\lfloor\frac{2n}{3}\right\rfloor\right),$$

sumando y por la recurrencia obtenemos

$$T(n) \leq c\left\lfloor\frac{n}{3}\right\rfloor \log\left(\left\lfloor\frac{n}{3}\right\rfloor\right) + c\left\lfloor\frac{2n}{3}\right\rfloor \log\left(\left\lfloor\frac{2n}{3}\right\rfloor\right) + dn,$$

luego como $\lfloor x \rfloor \leq x$ y por monotonía de $\log$ tenemos

$$T(n) \leq c\frac{n}{3}\log\left(\frac{n}{3}\right) + c\frac{2n}{3}\log\left(\frac{2n}{3}\right) + dn = \frac{cn}{3}\left(\log\left(\frac{n}{3}\right) + 2\log\left(\frac{2n}{3}\right)\right) + dn \leq cn\log(n) + dn,$$

por propiedades del logaritmo obtenemos

$$T(n) \leq cn\log(n) - cn\log(3) + \frac{2cn}{3} + dn = O(n\log(n)).$$

Por Inducción Matemática concluimos

$$T(n) = O(n\log(n)).$$

Si $T(n) = T(n/3) + T(2n/3) + n$ entonces

$$T(n) = \theta(n\log(n))$$

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

    (a) $T(n) = 2T(n/3) + n\log n$
    Sea $\epsilon = 0{,}3$, luego para $n \geq 2$

    $$n^{\log_3(2)+\epsilon} \leq n \leq n\log(n),$$

    por lo que

    $$n\log(n) = \Omega(n^{\log_b(a)+\epsilon}).$$

    Ahora como

    $$\left(\frac{2}{3}\right) n(\log(n) - \log(3)) < \left(\frac{2}{3}\right) n\log(n),$$

    entonces

    $$2\left(\frac{n}{3}\right)\log\left(\frac{n}{3}\right) < [\left(\frac{2}{3}\right) n\log(n).$$

    Por el Masterm Theorem concluimos que

    $$T(n) = \Theta(n\log(n))$$

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

Sea $\epsilon = 0{,}1$, luego para $n \geq 2$

$$n^{\log_3(2)+\epsilon} \leq n^3,$$

por lo que

$$n^3 = \Omega(n^{\log_b(a)+\epsilon}).$$

Ahora con $c = 0{,}9$ tenemos que

$$7\left(\frac{n}{2}\right)^2 \leq cn^3.$$

Por el Masterm Theorem concluimos que

$$T(n) = \Theta(n^3).$$

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*.

$$T(n) = 2T(n/2) + n$$

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

$$T(n) = \Theta(n \log(n))$$

Esto es cierto por el Master Theorem, ya que $n = n^{\log_2(2)}$.

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 m veces?

Para la primera pregunta sabemos que siempre vamos a contratar a la primera persona y siempre vamos a contratar a la persona mas calificada, por lo que si la primera persona no es la mejor calificada entonces contratarias al menos 2, luego si son la misma persona entonce ya no contratarias otra persona por lo que el orden en que vengan el resto no importa, por lo que la probabilidad es

$$\frac{(n-1)!}{n!}.$$

Para la segunda pregunta como venía escrito original es que contratases a todas las personas por lo que solo hay una manera y es que entrevistes del menos calificado al mas calificado, entonces su probabilidad es

$$\frac{1}{n!}.$$

Si la n de contratar es diferente a la n de aspirantes entonces veamos lo siguiente. Queremos contratar m veces de los n aspirantes, entonces la cantidad de permutaciones de entrevistas que cumplen esto es lo mismo que fijarnos en el siguiente problema

¿Cuantas formas hay de ordernar n pilas de altura entre 1 y n en una fila (con misma ancho), tal que si te paras enfrente de todas solo puedas ver m?

Ahora sea $a_{n,m}$ el número de formas descrito anteriormente, fijemonos en la pila mas grande entonces sabemos que este no importa donde este siempre la veremos entonces digammos que en una permutación esta se encuentra en la pisición i entonces no importa el orden de las $n-i$ pilas del final estas nunca se veran, luego ahora en las primeras $i-1$ posiciones queremos que solo se vean $m-1$, por lo que hay $a_{i-1,m-1}$ formas de hacerlo (ya que podemos hacer una biyección entre las pilas de la primeras $i-1$ posiciones y pilas de altura entre 1 y $i-1$). Luego entonces obtenemos la siguiente recursión con $a_{n,0} = a_{0,0} = 1$.

$$a_{n+1,m} = \sum_{i=1}^{n+1} \frac{n! a_{i-1,m-1}}{(i-1)!}.$$

Por lo tanto la probabilidad de contratar m veces es

$$\frac{a_{n,m}}{n!}$$

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

Sean $E_k, k \in \{1,2,3,4,5,6\}$ los eventos donde el dado sale k. Sea X la variable aleatoria que nos devuelve el valor de lanzar un dado, es decir

$$X = \sum_{i=1}^{6} k \mathbb{1}_{[E_k]}.$$

Queremos calcular

$$\mathbb{E}\left[\sum_{i=1}^{n} X\right],$$

por linealidad de la esperanza esto es

$$n\mathbb{E}\left[X\right],$$

por definición de esperanza esto es

$$n\sum_{i=1}^{6} iP[X=i],$$

en caso de que el dado sea incesgado entonces concluimos

$$\mathbb{E}\left[\sum_{i=1}^{n} X\right] = n\sum_{i=1}^{6}\frac{i}{6} = n(3{,}5)$$