PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC1001 — Algoritmos y Sistemas Computacionales — 2023-1

Solución Interrogación 1

Lunes 3-Abril-2023

Duración: 90 minutos
Responder cada pregunta en una hoja separada.

1. **[14p]** Las siguientes expresiones representan la cantidad de pasos que debe realizar un algoritmo, donde n y m son variables que dependen de los datos de entrada; por otro lado, letras como A, B, C, K son constantes. Para cada expresión indique su complejidad asintótica y finalmente ordene los algoritmos desde el menos al más eficiente asintóticamente.

   Respuesta: [2pts] cada una. Para cada respuesta:

   2pts. Si la respuesta es correcta. No es necesario justificar.

   1pto. Respuesta incorrecta, pero hay alguna justificación con poco error que indique cómo se llegó al resultado. Respuesta del estilo $O(n^2) + O(n) + O(1)$, pues se trata de una respuesta correcta pero incompleta (esa suma es $O(n^2)$).

   0pto. Respuesta incorrecta, o con justificación incorrecta, o vacía. Respuestas del estilo $O(3n^2)$ con incorrectas, pues no debe haber constantes en la notación.

   1.1) $1200n + C$
   R. $O(n)$

   1.2) $0{,}0002n^3 + 5000n^2 + 2n + 50$
   R. $O(n^3)$

   1.3) $\log n + 10 + 2n$
   R. $O(n)$

   1.4) $40n^3 + n^2 + 0{,}1n + 2^n$
   R. $O(2^n)$

   1.5) $2^{10}n^2 + K$
   R. $O(n^2)$

   1.6) $10! + B \times C$
   R. $O(1)$

   Orden de eficiencia asíntótico **[2pts]**: $O(2^n), O(n^3), O(n^2), O(n), O(1)$. Si hay uno repetido pueden ponerlo dos veces o una. También podrían ordenarlo como 1.4, 1.2, 1.5, 1.1(y/ó 1.3), 1.6. No considerar errores de arrastre. Si hay error previo, se evalúa que sea consistente.

   2pts. Orden correcto y consistente con los resultados previos.

   1pto. Un error de posición (u ordenarlos exactamente de mayor a menor)

   0pto. Dos o más errores de posición. Respuesta vacía.

2. **[28p]** Alice y Bob son los ayudantes encargados de cuidar una prueba de Introducción a la Programación. Ellos tienen que pasar la lista en una sala de n estudiantes. La lista está compuesta por h hojas, que contienen todos los nombres de los alumnos de la sala ordenados alfabéticamente. Cada hoja tiene aproximadamente n/h nombres. El objetivo de pasar lista es que todos alumnos de la sala firmen la hoja al lado de su nombre.

   Generalmente el procedimiento que usa Alice para pasar lista es:

(1) La ayudante se acerca a cada alumno, uno por uno, y hace lo siguiente.
   (a) Le pide la TUC o identificación similar.
   (b) Busca su nombre en la lista, usando búsqueda binaria.
   (c) Le pide al alumno firmar al lado de su nombre.

Sin embargo, Bob no sabe hacer búsqueda binaria por lo que siempre usa otro procedimiento:

(1) El ayudante se acerca a cada alumno, uno por uno, y hace lo siguiente.
   (a) Le pide la TUC o identificación similar.
   (b) Busca su nombre en la lista, viendo uno por uno hasta encontrarlo.
   (c) Le pide al alumno firmar al lado de su nombre.

Responda las siguientes preguntas:

2.1) **[6p]** ¿Cuál es la complejidad del algoritmo de Alice? Use la notación asintótica y argumente por qué tiene esa complejidad.

Respuesta: El algoritmo considera 1 paso para obtener la TUC, $\log n$ pasos encontrar al alumno, y 1 paso para firmar. Esto hay que repetirlo n veces. La complejidad es $n(\log n + 2) = n \log n + 2$, lo cual asintóticamente es $O(n \log n)$ ya que predomina el término $n \log n$.

2 pts. Describir como se valoró cada paso del algoritmo. 1pto si hay errores menores.

2 pts. Obtener la fórmula de complejidad. 1pto si hay errores menores. 0pto si es incorrecta. Debe incluir los pasos constantes. Si incluye el valor h en el cálculo de manera consistente se considera correcto, aunque en el peor caso se puede considerar todas las h hojas como solo una lista de tamaño total n.

2 pts. Obtener la complejidad asintótica correctamente. No debe haber constantes ni otros términos. 0pto si es incorrecta.

2.2) **[6p]** ¿Cuál es la complejidad del algoritmo de Bob? Use la notación asintótica y argumente por qué tiene esa complejidad.

Respuesta: El algoritmo considera 1 paso para obtener la TUC, n pasos en el peor caso para encontrar al alumno, y 1 paso para firmar. Esto hay que repetirlo n veces. La complejidad es $n(n + 2) = n^2 + 2n$, lo cual asintóticamente es $O(n^2)$ ya que predomina el término n^2.

2 pts. Describir como se valoró cada paso del algoritmo. 1pto si hay errores menores.

2 pts. Obtener la fórmula de complejidad. 1pto si hay errores menores. 0pto si es incorrecta. Debe incluir los pasos constantes. Si incluye el valor h en el cálculo de manera consistente se considera correcto, aunque en el peor caso se puede considerar todas las h hojas como solo una lista de tamaño total n.

2 pts. Obtener la complejidad asintótica correctamente. No debe haber constantes ni otros términos. 0pto si es incorrecta.

2.3) **[4p]** Supongamos que no hay muchos alumnos en la sala ¿Quién de los dos pasará la lista más rápido? Argumenta por qué.

Respuesta: Para esta comparación debemos utlizar la fórmula de complejidad obtenida en los pasos anteriores, y no la notación asintótica pues esta representa el comportamiento ante un n muy grande. Algoritmo de Alice (A_A): $n(\log n + 2) = n \log n + 2$. Algoritmo de Bob ($A_B$): $n(n + 2) = n^2 + 2n$. Si se llegó a un resultado distinto, evaluar de manera consistente y no considerar error de arrastre.

Para valores pequeños, se pueden considerar algunos valores aproximados.

n	A_A	A_B
1	2	3
2	6	8
3	~ 10	15
4	16	24
5	~ 21	35
8	40	80
...	...	...

Se puede observar que para valores pequeños el algoritmo A_A utiliza menos pasos que A_B. Debido a la complejidad de cada uno, una vez que A_B supera a A_A, solo se alejarán cada vez más. Por lo tanto en cualquier caso Alice lo hará de manera más rápida.

En caso que se haya llegado a una complejidad distinta se debe indicar exactamente a partir de cuántos estudiantes un algoritmo pasa a ser más rápido que el otro.

2 pts. Dar la respuesta correcta.

2 pts. Deducción del resultado. 1 pto si usa la notación asintótica para calcular. Debe haber al menos dos casos concretos.

2.4) **[6p]** Lamentablemente, esta vez el malvado ayudante jefe les entregó una lista con todos los alumnos desordenados (antes la lista estaba ordenada alfabéticamente). Nota: Esto no está basado en hechos reales.

¿Qué implica este cambio? ¿Los ayudantes podrán realizar sus algoritmos de la misma manera? Para cada ayudante, argumenta por qué podrá realizar el mismo algoritmo para pasar lista, o de lo contrario, por qué deberá modificarlo.

Respuesta: En este caso el algoritmo de Alice no podrá usar el paso de búsqueda binaria pues este algoritmo supone que los datos están ordenados. En ese caso el algoritmo de Alice no puede funcionar. Ahora bien, también se puede argumentar que Alice tendría ahora que ordenar la lista antes de ejecutar su algoritmo. La complejidad de A_A dependería de ese algoritmo de ordenamiento previo. Si se considera algún algoritmo como Selectionsort, Bubblesort o Insertionsort, la complejidad cambiaría a $n^2 + n(\log n + 2) = n^2 + n\log n + 2n$ que asintóticamente es $O(n^2)$. Si se considera algún algoritmo como Mergesort o Quicksort, sería $n\log n + n(\log n + 2) = 2n\log n + 2n$ que asintóticamente es $O(n\log n)$.

El algoritmo de Bob puede funcionar de la misma manera que antes pues su ejecución no depende del orden de la lista, y seguiría teniendo complejidad $n(n+2) = n^2 + 2n$.

Al comparar ambas complejidades nuevas, el algoritmo A_A pasa a ser peor que A_B en el largo plazo si se usa un algoritmo de ordenamiento de complejidad $O(n^2)$, pero se mantiene mejor que A_B si se usa un algoritmo de ordenamiento de complejidad $O(n\log n)$.

2 pts. Indicar qué pasa con A_A. Si dice que no se puede ejecutar, debe indicar al menos una manera de modificarlo. Es válida decir que A_A se convierte en A_B porque habría que mirar toda la lista. La complejidad obtenidad debe indicarse acordemente. No es obligatorio hacer referencia a los algoritmos de ordenamiento, pero sí es una respuesta válida. 1pto si no indica cómo modificarlo.

2 pts. Indicar qué pasa con A_B. Debe indicar que no necesita modificación.

2 pts. Indicar cuál es mejor bajo las nuevas condiciones. Debe ser consistente con lo que mencionó en los dos puntos anteriores. No es obligatorio hacer referencia a los algoritmos de ordenamiento. 0 pts si indica cuál es mejor pero no especifica como varía cada algoritmo.

2.5) **[6p]** Ante el cambio de la pregunta anterior (lista desordenada), Bob propone un nuevo algoritmo para pasar lista:

(1) El ayudante dice en voz alta cada nombre de la lista, y para cada uno:
   (a) Espera hasta que el alumno levante la mano.
   (b) Le pide al alumno que se acerque al frente de la sala **con su TUC**.
   (c) Le pide al alumno que firme al lado de su nombre.

Sin embargo, existe un problema más. Los ayudantes están cuidando la tenebrosa sala A7. La particularidad de esta sala es que está inclinada, lo que significa que es muy incómoda para sentarse y además si se te cae un lápiz este rodará hasta el final de la sala (esto sí está basado en hechos reales). Bob se dio cuenta que, al usar su nuevo algoritmo, a muchos alumnos se les caía un lápiz, causando que perdieran alrededor de 5 minutos buscándolo. Esto significa que, por ejemplo, si a 12 alumnos se les cae un lápiz, pasar la lista tomará 1 hora más.

Considerando que hay muchos alumnos en la sala y existe un retraso de 5 min por cada alumno que se le cae el lápiz. ¿Cuál algoritmo es mejor? ¿Alguno de los anteriores o el nuevo algoritmo de Bob? Menciona la complejidad en notación asintótica del nuevo algoritmo.

Respuesta: El tiempo de moverse para cada estudiante se puede considerar constante: C. El nuevo algoritmo de Bob es 1 paso para esperar que el alumno levante la mano, C para que se mueva adelante considerando que se puede caer el lápiz, 1 paso que firme. El tiempo está acotado por $C + 2$. Como hay que repetirlo para los n estudiantes, la complejidad es $2n + 2C$, que tiene complejidad asintótica lineal: $O(n)$. Este algoritmo es asintóticamente más eficiente que los otros dos, pero para poder comparar correctamente habría que estimar en segundos lo que demora sacar la TUC y firmar. Supongamos que cada uno de esos pasos son 5 segundos. El tiempo para tomar la lista a cada estudiante en segundos sería $5 \times 60 + 10 = 130$ segundos. Esto sigue siendo un valor constante por lo tanto la complejidad asintótica sigue siendo $O(n)$. Sin embargo para valores pequeños de n el tiempo de pasar la lista puede crecer muy rápidamente.

2 pts. Obtener y justificar la complejidad del nuevo algoritmo. 1pto respuesta/justificación incompleta.

2 pts. Obtener la complejidad asintótica del nuevo algoritmo.

2 pts. Argumentar cuál es mejor. Es válido indicar que el nuevo algoritmo de Bob es mejor, pero debe hacer al menos una referencia a la implicancia de que la parte constante puede ser muy alta.

3. **[12p]** En la siguiente tabla, cada columna representa la medición de tiempo de cuatro algoritmos distintos: A_1, A_2, A_3, y A_4. La columna de la izquierda indica el tamaño de la entrada. Deduzca la complejidad asintótica de cada algoritmo medido.

   (3 pts cada algoritmo)

n	A_1	A_2	A_3	A_4
1	2	1	0	2
2	4	4	1	4
4	8	16	2	16
8	16	68	3	256
64	128	4200	6	$1{,}8 \times 10^{19}$
256	512	65700	8	$1{,}15 \times 10^{77}$
1024	2048	1138440	10	$1{,}7 \times 10^{308}$

   Respuesta:

   A_1: $O(n)$. 1pto si escriben $O(2n)$. 0pto si es otra fórmula.

   A_2: $O(n^2)$. 1pto si es una fórmula cuadrática. 0pto si es otra fórmula.

   A_3: $O(\log n)$. Debe ir sin base, o con base 2. 2pts si especifican otra base. 1pto si es alguna fórmula logarítmica. 0pto si es otra fórmula.

   A_4: $O(2^n)$. 1pto si es otra base o fórmula exponencial. 0pto si es otra fórmula.

   Como ayuda se le entrega una tabla de referencia de potencias de 2:

n	2^n	n	2^n	n	2^n
1	2	11	2048	21	2097152
2	4	12	4096	22	4194304
3	8	13	8192	30	1073741824
4	16	14	16384	40	1099511627776
5	32	15	32768	64	$1{,}8 \times 10^{19}$
6	64	16	65536	256	$1{,}15 \times 10^{77}$
7	128	17	131072	1024	$1{,}7 \times 10^{308}$
8	256	18	262144		
9	512	19	524288		
10	1024	20	1048576		