Universidad Nacional Autónoma de México

Facultad de Ciencias

Análisis de Algoritmos | 7083

Tarea 2 : | Algoritmos de ordenamiento

Sosa Romo Juan Mario hola | 320051926

09/09/24

1. **Cada año, cuando los estudiantes llegan a Hogwarts, el sombrero seleccionador clasifica a los nuevos estudiantes en una de las cuatro casas Gryffindor, Hufflepuff, Ravenclaw o Slytherin. Un profesor pone el sombrero en la cabeza del estudiante y después de tiempo constante el sombrero le dice a qué casa se unirá. Este año, los gemelos Weasley llenaron casi todo Hogwarts con una pegajosa sustancia marrón, momentos antes de la selección anual. Como resultado, la selección tuvo que llevarse a cabo en los pasillos del sótano, donde había tan poco espacio para moverse que los estudiantes tenían que hacer una larga fila. Después de que todos supieran en qué casa estaban, los estudiantes trataron de agruparse por casa, pero había muy poco espacio en el pasillo para que más de un estudiante se moviera a la vez. Afortunadamente, el sombrero seleccionador tomó algoritmos hace muchos años, por lo que sabía cómo agrupar a los estudiantes en tiempo lineal a pesar del poco espacio ¿Qué algoritmo usó el sombrero seleccionador?.**

2. **Queremos ordenar una lista *S* de n enteros que contiene muchos elementos duplicados. Supongamos que los elementos de *S* sólo tienen $O(log\, n)$ valores distintos.**

   (a) **Encuentre un algoritmo que toma a lo mas $O(n\, log\, log\, n)$ tiempo para ordenar *S*.**

   (b) **¿Por qué no viola la cota inferior de $O(n\, log\, n)$ para el problema de ordenación.**

3. **Suponga que tenemos dos arreglos ordenados $A[1, \ldots n]\ y\ B[1, \ldots, n]$ y un entero k. Describe un algoritmo para encontrar el *k-ésimo* elemeneto en la unión de *A* y *B*. Por ejemplo, si $k = 1$, tu algoritmo debe regresar el elemento más pequeño de $A \cup B$; si $k = n$, tu algoritmo debe regresar la mediana de $A \cup B$. Puedes suponer que los arreglos no contienen duplicados. Tu algoritmo debe tener complejidad de tiempo $\Theta(log\, n)$. Hint: Primero resuelve el caso especial k=n.**

4. **Sea *A* un arreglo de n númreos enteros distintos. Suponga que *A* tiene la siguiente propiedad: existe un indice $1 \leq k \leq n$ tal que $A[1], \ldots, A[k]$ es una secuencia incremental y $A[k+1], \ldots, A[n]$ es una secuencia decremental.**

   (a) **Diseña y analiza un algoritmo eficiente para encontrar k.**

   (b) **Sii no conoces el valor de n, cómo resuelves el problema.**

5. **Rotar un arreglo significa hacer un corrimiento a la derecha de todos los elementos, excepto el último que pasará a la primera posición del arreglo. Por ejemplo $A = [1, 3, 4, 5, 7, 10, 14, 15, 16, 19, 20, 25]$ y después de una rotación $A = [25, 1, 3, 4, 5, 7, 10, 14, 15, 16, 19, 20]$.**

Dado un arreglo ordenado *A* de n números enteros distintos que ha sido rotado un número desconocido de veces. Diseña un algoritmo de $O(log\, n)$ tiempo que encuentre el *k-ésimo* elemento del arreglo. Por ejemplo, $A = [15, 16, 19, 20, 25, 1, 3, 4, 5, 7, 10, 14]$ y $k = 5$ la respuesta es 7.

6. **Eres un joven cientifico quien acaba de recibir un nuevo trabajo en un gran equipo de 100 personas (tu eres la persona 101). Un amigo tuyo en quien confias te dijo que tienes mas colegas honestos que metirosos; eso es todo lo que te dijo. Un mentiroso es una persona que puede decir una verdad o una mentira, mientras que una persona honesta es alguien que siempre dice la verdad. Claro, a ti te gustaría saber quienes son los mentirosos y quienes son honestos, asi que decides empezar una investigación. Haces una serie de preguntas a tus colegas, Ya que no quieres aparecer sospechoso, decides solo hacer preguntas del tipo "¿Y una persona honesta?" y claro, quieres hacer las menos preguntas posibles. ¿Puedes distinguir a todos tus colegas honestos?. ¿Cúal es la mínima cantidad de preguntas que tienes que preguntar en el peor caso? Puedes asumir que tus colegas se conocen muy bien entre ellos, lo suficiente para decir si otra persona es un mentiroso o no. (Hint: Agrupa a las personas en pares (X, Y) y pregunta a X si Y es mentiroso y después a Y preguntale si X es mentiroso o a los 2 si el otro es honest; Analiza las 4 posibles respuestas. Una vez encuentres a una persona honesta, puedes usarlo para encontrar a todos los otros. Como desafio, ¿puedes resolver el problema con menos de 280 preguntas?.)**

   Generaliza la estrategía de arriba para mostrar que dadas *n* personas tal que menos de la mitad son metirosos, puedes separar a los mentirosos de los honestos en $\Theta(n)$ preguntas.

7. **Sea $A[1, \ldots, n]$ un arreglo de números reales. Diseña un algoritmo para realizar cualquier secuencia de las siguientes operaciones:**

   - **Add(i, j)**: suma el valor y al *i-ésimo* número.
   - **PartialSum(i)**, regresa la suma de los primeros i números, es decir $PartialSum(i) = A[1] + \cdots + A[i]$

   No hay inserciones ni eliminaciones; el único cambio es a los valores de los números. Cada operación debe tomar $O(log\, n)$ pasos. puedes utilizar un arreglo adicional de tamaño n como espacio de trabajo.

8. **Permutaciones de Josephus: Supongamos que n personas están sentadas alrededor de una mesa circular con n sillas, y que tenemos un entero positivo $m \leq n$. Comenzando con la persona con etiqueta 1, (moviendonos siempre en la dirección de las manecillas del reloj) comenzamos a remover los ocupantes de las sillas como sigue: Primero eliminiamos la persona con etiqueta m. Recursivamente, eliminamos al *m-ésimo* elemento de los elementos restantes. Este proceso continua hasta que las n personas han sido eliminadas. El orden en que las personas han sido eliminadas, se le conoce como la *(n,m)-permutación de Josephus*. Por ejemplo si $n = 7$ y $m = 3$, la ,*(7,3)-permutación de Josephus* es: {3,6,2,7,5,1,4}**

   (a) **Supongamos que m es constante. De un algoritmo lineal para generarla *(n,m)-permutación de Josephus*.**

(b) **Supongamos que** m **no es constante. Describa un algoritmo de complejidad** $O(n \log n)$ **para encontrar la *(n,m)-permutación de Josephus*. Hint: Construya un árbol de búsqueda de rangos.**