IIC2283 — Diseño y Análisis de Algoritmos
Ayudantía Interrogación 1

Juan Navarro

Problema 1

Se considera el problema de decidir si una lista de n bits contiene tres unos consecutivos. Demuestre que todo algoritmo correcto para este problema debe consultar al menos $4\lfloor \frac{n}{5} \rfloor$ posiciones.

Solución Se demuestra en base a un adversario, utilizando un juego con dos etapas: en la primera, en cada paso el algoritmo pide el valor en una posición, y el adversario se lo entrega; y luego, en la segunda etapa el algoritmo declara si existen o no tres unos consecutivos, y el adversario genera una instancia completa, que debe ser consistente con las respuestas entregadas en la primera fase. El algoritmo gana si su respuesta es correcta dada la instancia generada por el adversario, y pierde en otro caso.

En este juego, el adversario utiliza la siguiente estrategia:

- Divide la lista en bloques de largo 5.
- En la posición 5 de cada bloque, el adversario entrega el valor 0.
- Si el último bloque tiene largo menor a 5, entonces en esas posiciones también retorna 0.
- En los primeros cuatro elementos de cada bloque, el adversario responde 0 en la primera consulta en un borde, y en la última consulta de las cuatro. Responde 1 en los otros dos casos.

Si el algoritmo revisa menos de $4\lfloor \frac{n}{5} \rfloor$ posiciones, entonces en algún bloque no consulta las cuatro primeras posiciones. Luego, si el algoritmo responde SI, entonces el adversario puede completar las posiciones no consultadas con ceros, y causar que el algoritmo pierda. Por otra parte, si el algoritmo responde NO, entonces el adversario puede completar algún bloque donde no se consultaron todas las posiciones con unos, formando tres unos consecutivos, con lo cual el algoritmo también pierde.

Problema 2

Se tienen n tuercas y n pernos desordenados, donde cada tuerca coincide exactamente con un perno. En este problema se busca asignar cada perno con la tuerca que le corresponde, para lo cual la única operación disponible es probar un perno con una tuerca, y así verificar si coinciden, o si el perno es demasiado grande, o si es demasiado pequeño.

Encuentre una cota inferior al número de pruebas necesarias para encontrar la asignación correcta.

Solución Utilizamos un árbol de decisión ternario, por que la operación tiene tres resultados posibles. La cantidad de hojas del árbol puede acotarse con la cantidad de salidas posibles del algoritmo, que se pueden contar considerando que para una permutación fija de las tuercas, cada salida posible es una permutación de los pernos. De esta forma, existen al menos $n! > (n/2)^{n/2}$ hojas, por lo cual todo algoritmo correcto requiere $(n/2)\log_3(n/2) \in \Omega(n \log n)$ operaciones.

La cota anterior es ajustada, es decir existe un algoritmo determinístico para este problema, con tiempo $O(n \log n)$ en el peor caso, pero su funcionamiento no es trivial.

Problema 3

Dado el siguiente algoritmo, donde A es una lista de n valores desordenados entre 0 y 1.

function BucketSort(A)
  for $i = 0..(n-1)$ **do**
    $B[i] \leftarrow \emptyset$
  end for
  for $i = 0..(n-1)$ **do**
    Insertar $A[i]$ en la lista $B[\lfloor A[i] \cdot n \rfloor]$
  end for
  for $i = 0..(n-1)$ **do**
    Ordenar $B[i]$ con Insertion Sort
  end for
  return Concatenación de B en orden
end function

Demuestre que si los valores de A distribuyen $\mathit{Uniforme}(0,1)$, entonces en promedio el algoritmo pertenece a $\Theta(n)$.

Solución Denominamos por n_i a una variable aleatoria que indica la cantidad de elementos en el bucket i. Como insertion sort pertenece a $O(n^2)$, entonces el tiempo de bucket sort está dado por:

$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2).$$

Luego, el tiempo promedio del algoritmo es:

$$\begin{aligned}
\mathbb{E}\left[T(n)\right] &= \mathbb{E}\left[\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)\right] \\
&= \Theta(n) + \sum_{i=0}^{n-1} \mathbb{E}\left[O(n_i^2)\right] \\
&= \Theta(n) + \sum_{i=0}^{n-1} O(\mathbb{E}\left[n_i^2\right]).
\end{aligned}$$

Por lo que se necesita resolver $\mathbb{E}\left[n_i^2\right]$. Para esto, definimos el conjunto de variables aleatorias X_{ij}, donde

$$X_{ij} = \begin{cases} 1 & A[j] \text{ se asignó al bucket } B[i] \\ 0 & \text{en otro caso} \end{cases}$$

Considerando lo anterior, se tiene que

$$\begin{aligned}
\mathbb{E}\left[n_i^2\right] &= \mathbb{E}\left[(\sum_{j=0}^{n-1} X_{ij})^2\right] \\
&= \mathbb{E}\left[\sum_{j=0}^{n-1} X_{ij}^2 + \sum_{j=0}^{n-1} \sum_{\substack{k=0 \\ k \neq j}}^{n-1} X_{ij} X_{ik}\right] \\
&= \sum_{j=0}^{n-1} \mathbb{E}\left[X_{ij}^2\right] + \sum_{j=0}^{n-1} \sum_{\substack{k=0 \\ k \neq j}}^{n-1} \mathbb{E}\left[X_{ij} X_{ik}\right].
\end{aligned}$$

Luego,

$$\mathbb{E}\left[X_{ij}^2\right] = 1^2 \cdot Pr[X_{ij} = 1] + 0^2 \cdot Pr[X_{ij} = 0] = \frac{1}{n}.$$

Por otra parte, si $j \neq k$, entonces X_{ij} es independiente con X_{ik}, y por lo tanto

$$\mathbb{E}\left[X_{ij}X_{ik}\right] = \mathbb{E}\left[X_{ij}\right]\mathbb{E}\left[X_{ik}\right] = \frac{1}{n^2}.$$

Finalmente, reemplazando en la expresión anterior, se obtiene que:

$$\begin{aligned}
\mathbb{E}\left[n_i^2\right] &= \sum_{j=0}^{n-1} \mathbb{E}\left[X_{ij}^2\right] + \sum_{j=0}^{n-1}\sum_{\substack{k=0\\k\neq j}}^{n-1} \mathbb{E}\left[X_{ij}X_{ik}\right] \\
&= \sum_{j=0}^{n-1} \frac{1}{n} + \sum_{j=0}^{n-1}\sum_{\substack{k=0\\k\neq j}}^{n-1} \frac{1}{n^2} \\
&= 1 + \frac{n(n-1)}{n^2} \\
&= 2 - \frac{1}{n} \\
&\in O(1)
\end{aligned}$$

y por lo tanto, que

$$\mathbb{E}\left[T(n)\right] = \Theta(n) + nO(1) \in \Theta(n).$$