IIC2283 — Diseño y Análisis de Algoritmos
Ayudantía 4

Problema 1

En este problema se considera el problema de emparejar pernos con tuercas. Para esto, la única operación disponible es comparar un perno con una tuerca, y verificar si calzan, si la tuerca es demasiado grande, o si la tuerca es demasiado pequeña.

Parte 1

Inicialmente se tienen N pernos y N tuercas distintas, donde cada tuerca calza exactamente con un perno. Demuestre que todo algoritmo para encontrar la asignación correcta utiliza $\Omega(N \log N)$ comparaciones en el peor caso.

Solución Cada comparación entre un perno y una tuerca tiene tres resultados posibles. Luego, cualquier algoritmo que resuelva este problema se puede representar como un árbol de decisión ternario, donde los nodos internos corresponden a las comparaciones, y las hojas a las salidas posibles del algoritmo, es decir, asignaciones entre tuercas y pernos.

La cantidad de hojas se puede acotar fijando una permutación de los pernos, y luego considerando que cada asignación posible entre pernos y tuercas corresponde a una permutación de las tuercas. De ahí, se tiene que hay al menos $N!$ hojas y por lo tanto que el tiempo de todo algoritmo, en el peor caso, es al menos

$$\log_3(N!) \geq \log_3((N/2)^{N/2}) \in \Theta(N \log N)$$

Parte 2

Si ahora se tienen N pernos distintos entre sí, y M tuercas no necesariamente distintas. ¿Cuantas comparaciones se necesitan en el peor caso?.

Idea de solución Similar al anterior. En este caso, la cantidad de hojas se puede calcular considerando que por cada tuerca hay $N+1$ opciones: N pernos distintos, más la opción de que no coincida con ninguno. Luego, el tiempo del algoritmo es al menos:

$$\log_3((N+1)^M) \in \Omega(M \log N)$$

Parte 3

Suponga que los pernos pueden compararse, y por lo tanto también ser ordenados. Proponga un algoritmo para resolver el problema de la parte 2, y analice su tiempo de ejecución. ¿Como se compara con la cota obtenida?. ¿Como se podría resolver sin ordenar los pernos?.

Idea de solución En ese caso se pueden ordenar los pernos (por ejemplo, con mergesort) en tiempo $\Theta(n \log n)$, y luego asignar cada tuerca por búsqueda binaria. En total este algoritmo funciona en tiempo $\Theta((N+M)\log N)$.

Si no se pueden ordenar los pernos, entonces se puede utilizar un algoritmo similar a quicksort, basado en encontrar un par perno/tuerca, y utilizarlos para particionar los pernos y las tuercas en dos conjuntos –menores y mayores–, donde se continúa recursivamente. En el peor caso este algoritmo funciona en tiempo $\Theta(N^2)$, pero en promedio, asumiendo distribución uniforme, funciona en tiempo $\Theta(N \log N)$. De todas formas, existe un algoritmo con peor caso $\Theta(N \log N)$, pero esto es difícil de demostrar.

Problema 2

Los árboles de decisión se pueden generalizar en árboles de decisión lineales, que representan funciones de $\mathbb{R}^n$ en $\{0,1\}$. Estos son árboles ternarios, donde cada nodo interno está etiquetado con un vector $v_i \in \mathbb{R}^n$, cada arco de salida de cada nodo con un elemento distinto de $\{-1,0,1\}$, y cada hoja con un valor $\{0,1\}$.

Dado un árbol y un vector x, se calcula el valor de la función comenzando desde la raíz, y siguiendo el arco de salida que corresponda según:

$$\begin{cases} -1 & \langle x, v_i \rangle < 0 \\ 0 & \langle x, v_i \rangle = 0 \\ 1 & \langle x, v_i \rangle > 0 \end{cases}$$

hasta alcanzar una hoja.

Además, definimos que un conjunto $C \subseteq \mathbb{R}^n$ es convexo, si para todos $a, b \in C$, y $\lambda \in [0,1]$, $\lambda a + (1-\lambda)b \in C$.

Parte 1

Muestre que la intersección de conjuntos convexos es convexa.

Solución Si $\mathcal{C}$ es una colección de conjuntos convexos, y $a, b \in \cap\mathcal{C}$, entonces se considera $\lambda \in [0,1]$ arbitrario. Para todo $C \in \mathcal{C}$, se tiene que $a, b \in C$, y como C es convexo que $\lambda a + (1-\lambda)b \in C$. Luego, $\lambda a + (1-\lambda)b \in \cap\mathcal{C}$, y por lo tanto $\cap\mathcal{C}$ es convexo.

Parte 2

Dado un nodo v del árbol, denotamos por $P(v)$ al conjunto de vectores cuya evaluación pasa por el nodo v. Demuestre que para todo v, $P(v)$ es un conjunto convexo.

Idea de solución Por inducción en la altura. Si r es la raíz, $P(r)$ es igual a $\mathbb{R}^n$, y por lo tanto es convexo. En los otros casos, si el padre de v es t, entonces $P(v)$ es la intersección entre $P(t)$ y $\{s | t \cdot s \,\square\, 0\}$, donde $\square$ es $<, =, >$ dependiendo del caso. $P(v)$ es convexo por hipótesis de inducción, y el segundo conjunto cumple la definición de convexidad por las propiedades del producto punto. Finalmente, como $P(v)$ es la intersección de conjuntos convexos, entonces es convexo.

Parte 3

Muestre que dos puntos llegan a la misma hoja, si y solo si pertenecen a la misma componente conexa.

Idea de solución Dos puntos llegan a la misma hoja si y solo si cumplen las mismas desigualdades en cada nodo del camino desde la raíz. Por construcción, este conjunto de desigualdades define una única región de $\mathbb{R}^n$, que es disjunta de todas las otras.

Parte 4

Si el árbol calcula una función F, denominamos $\#F_0$ al número de componentes donde la función tiene valor 0, y $\#F_1$ al número de componentes donde tiene valor 1. Demuestre que todo árbol que calcula F tiene altura mayor o igual a $\log_3(\#F_0 + \#F_1)$.

Idea de solución Por la parte 3, el número de componentes corresponde al número de hojas, por lo tanto, si hay $\#F_0 + \#F_1$ componentes, entonces hay igual cantidad de hojas. Luego, como es un árbol ternario con $(\#F_0 + \#F_1)$ hojas, entonces su altura es al menos $\log_3(\#F_0 + \#F_1)$.

Parte 5

Demuestre que todo árbol lineal que calcula F:

$$F(x) = \begin{cases} 1 & x \text{ tiene elementos repetidos} \\ 0 & x \text{ no tiene elementos repetidos} \end{cases}$$

tiene altura al menos $(n/2)\log_3(n/2)$. Para esto considere un vector $z = (z_1, z_2, \ldots, z_n)$ con todas sus componentes distintas, junto con $z' = (z_2, z_1, \ldots, z_n)$.

Idea de solución Por definición $F(z) = F(z') = 0$. Luego, se considera la función $H(x) = x_2 - x_1$. Ahí, se tiene que $H(z) = -H(z')$, y que tanto $H(z)$ como $H(z')$ son distintos de cero. Esto implica que existe $\lambda^* \in [0, 1]$ tal que $H(\lambda^* z + (1 - \lambda^*)z') = 0$. Si definimos $z^* = \lambda^* z + (1 - \lambda^*)z'$, entonces se cumple que $z_1^* = z_2^*$, y por lo tanto que $F(z^*) = 1$.

Lo anterior muestra que z y z' pertenecen a componentes distintas, por que las componentes son convexas, y en todos los puntos de una componente la función tiene el mismo valor. Con este argumento se puede mostrar que cada permutación de z termina en una hoja distinta, y por lo tanto que hay al menos $n!$ hojas. Con ese resultado, se obtiene la cota pedida usando que $n! \leq (n/2)^{n/2}$.

Parte 6

Se considera el problema de encontrar elementos repetidos en una lista de largo n. Demuestre que todo algoritmo basado en comparaciones para este problema requiere $\Omega(n \log n)$ operaciones.

Idea de solución Se consideran algoritmos que puedan ser representados con árboles donde en cada nodo se comparan posiciones fijas del arreglo. Dado un árbol de este tipo, se puede construir un árbol lineal de la misma altura, transformando cada nodo donde se compare x_i con x_j, en un nodo con vector $e_i - e_j$. Luego, si existiera un árbol de comparaciones de altura $o(n \log n)$, entonces existiría un árbol lineal de la misma altura, lo que es una contradicción con el resultado anterior.

Problema 3

Determine cotas inferiores para los siguientes problemas:

1. Determinar el par de puntos más cercano dentro de un conjunto de puntos en el plano.

   Idea de solución Se utiliza una reducción desde el problema de encontrar números repetidos en una lista. En este caso, se considera una reducción con dos funciones H_1, H_2, tales que $ER(w) = H_1(PMC(H_2(w)))$, donde ER corresponde a encontrar elementos repetidos, PMC encontrar el par más cercano, H_2 la función que genera puntos de la forma $(x, 0)$ por cada $x \in w$, y H_1 la función que revisa si los puntos retornados por PMC son iguales.

En este caso, se tiene que

$$\Omega(n \log n) \leq \Theta(n) + t_{PMC}(n) + \Theta(1)$$

y por lo tanto que $t_{PMC}(n) \in \Omega(n \log n)$.

2. Dados n puntos en el plano, construir un árbol de cobertura de costo mínimo, donde el costo entre dos puntos es la distancia euclidiana correspondiente.

   Idea de solución Reducción desde elementos repetidos. En este caso, $ER(w) = H_1(MST(H_2(w)))$, donde H_2 es igual a la primera parte, MST calcula el árbol de cobertura de costo mínimo, y H_1 revisa si algún arco del árbol tiene costo 0.

   La reducción es correcta por que todo árbol de cobertura de costo mínimo tiene un arco de costo mínimo, y por lo tanto hay elementos repetidos si y solo si hay un arco de costo cero en el árbol final.

   El tiempo se puede acotar como:

$$\Omega(n \log n) \leq \Theta(n) + t_{MST}(n) + \Theta(n)$$

   y por lo tanto $t_{MST}(n) \in \Omega(n \log n)$.