Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 1 - "Cotas inferiores: Adversario"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: Dustin Cobas

18 de marzo del 2019

P1. Mezclar 2 listas ordenadas

Considere el problema de mezclar 2 listas ordenadas de igual tamaño, $X[1,\dots n], Y[1,\dots n]$ de modo que el resultado final este también ordenado. En este problema estaremos interesados en las preguntas del estilo: ¿$X[i] < Y[j]$?

a) Muestre que el problema es $\mathcal{O}(2n-1)$

b) Muestre que el problema no puede ser resuelto en menos de $2n-1$ comparaciones

P2. Conectividad de un grafo

Considere el problema de responder si un determinado grafo no dirigido es conexo. En este problema estaremos interesados en las preguntas del estilo: ¿Existe una arista entre los nodos u y v?

a) Muestre que el problema es $\mathcal{O}(\binom{n}{2})$

b) Muestre que el problema no puede ser resuelto en menos de $\binom{n}{2}$

P3. Grafo crítico

Considere una propiedad P de los grafos no dirigidos sin loops; por ejemplo, conectividad, aciclicidad, no-planaridad. Sea $G = (V, E)$ un grafo no completo; es decir, no todo par en $V \times V$ corresponde a un arco en E. Decimos que G es crítico para P si G no tiene la propiedad P, pero al agregarle cualquier arco en $(V \times V) \setminus E$ obtenemos un grafo G' que satisface P. Definimos $x := |(V \times V) \setminus E|$.
Sea $\mathcal{H}$ el conjunto de los grafos que utilizan el mismo conjunto de vértices V que un grafo crítico G para P. Considere un algoritmo que verifica si un grafo $H \in \mathcal{H}$ tiene la propiedad P utilizando solo preguntas de la forma: ¿existe un arco entre los vértices u y v?.
Muestre que este algoritmo debe realizar a lo menos x preguntas.

P1. Mezclar 2 listas ordenadas

a) El algoritmo de "merge" ocupado en "MergeSort" realiza en el peor caso $2n-1$ comparaciones (una por cada elemento que pone en la lista de los ordenados).

b) Por $\Rightarrow\Leftarrow$ existe un algoritmo A que realiza $\leq 2n-2$ comparaciones.

El adversario construye el siguiente input:

- $X \rightarrow$ lista cuyos elementos son $x_i = 2i-1$
- $Y \rightarrow$ lista cuyos elementos son $y_i = 2i$

Si ejecutamos el algoritmo A sobre el input anterior encontraremos un i^* (además de x_1) tal que x_{i^*} no fue comparado con ambos y_{i^*-1} e y_{i^*} (si todos los i lo cumplieran se habrían hecho más de $2n-2$ comparaciones).
Supongamos sin mucha pérdida de generalidad que solo no fue comparado con y_{i^*}, entonces en el siguiente input:

- $X \rightarrow$ lista cuyos elementos son $x_i = 2i-1$, excepto $x_{i^*} = 2i^*$
- $Y \rightarrow$ lista cuyos elementos son $y_i = 2i$, excepto $y_{i^*} = 2i^* - 1$

El algoritmo recibe exactamente las mismas respuestas a las mismas preguntas que se hizo sobre el input anterior (pues el orden relativo de los elementos no se altera y no se compara x_{i^*} con y_{i^*}). Por lo tanto tengo 2 inputs válidos de listas para los cuales A retorna el mismo resultado lo que es una $\Rightarrow\Leftarrow$.

P2. Conectividad de un grafo

a) El problema puede ser resuelto aplicando un algoritmo de búsqueda en grafos (como BFS o DFS), los cuales son lineales en el tamaño del grafo, como un grafo puede tener a lo más $\binom{n}{2}$ aristas, el problema es $\mathcal{O}(\binom{n}{2})$

b) El adversario irá respondiendo las preguntas del algoritmo de manera de darle la menor cantidad de información posible. Para esto mantendrá dos grafos consistentes con las respuestas que le da al algoritmo, pero siendo uno conexo y el otro no, siendo imposible distinguir entre ambos grafos.

Para responder las preguntas del algoritmo el adversario sigue las siguientes reglas:

- Inicializa dos grafos C como grafo completo (todas las aristas) e I como el grafo vacío (ninguna arista).
- Si le preguntan si existe una arista que desconecta a C respondo que si está y la agrego a I.
- Si no, respondo que no está y la saco de C.

Observemos primero que I es un subgrafo de C. Además, si C tiene un ciclo, ninguna de las aristas del ciclo están en I (pues sacar una de ellas no desconecta C), y por lo tanto I es acíclico.

Veamos también que si $I \neq C$ entonces I es disconexo, pues si fuese conexo, sería un árbol (por ser acíclico), pero como $I \neq C$ hay una arista e en C que no está en I y que de hecho es parte de un ciclo en C (cuyas otras aritas están todas en I), pero ninguna de las aristas de ese ciclo puede estar en I lo que es una $\Rightarrow\Leftarrow$.

Finalmente, si un algoritmo termina haciendo menos de $\binom{n}{2}$ preguntas hay una arista que está en C, pero no en I por lo que C es conexo pero I no.

P3. Grafo crítico

Cada vez que el algoritmo pregunta si existe arista (u, v), el adversario responde sí en caso que (u, v) sea una arista del grafo crítico G, y no en caso contrario. Asumamos por contradicción que el algoritmo hizo $y < x$ preguntas. Entonces hay un par $(u, v) \in (V \times V) \setminus E$ para el cual el algoritmo no ha preguntado su existencia. Sea G' el grafo obtenido desde G agregando el arco (u, v). Notemos que G y G' son lógicamente correcto respecto a las respuestas que el adversario le dio al algoritmo. Sin embargo, G no cumple $\mathcal{P}$ pero G' si lo hace (ya que G es crítico para $\mathcal{P}$), lo que es una contradicción.