IIC2283 — Diseño y Análisis de Algoritmos
Ayudantía 5

Juan Navarro

Problema 1

Dado un grafo $G = (V, E)$, se dice que $C \subseteq V$ es un *cubrimiento por vértices de* G, si para todo arco $(x, y) \in E$, se cumple que $x \in C$ o $y \in C$.

Describa y analice un algoritmo que permita encontrar un cubrimiento por vértices de tamaño mínimo de un árbol no dirigido.

Idea de solución Se puede elegir un nodo arbitrario como raíz. Luego por cada nodo del árbol se tiene el subproblema de encontrar el cubrimiento óptimo para el subárbol respectivo.

Dado un nodo arbitrario existen dos posibilidades: el nodo pertenece al cubrimiento óptimo, o no pertenece. Si pertenece, entonces el cubrimiento está formado por ese nodo, más los cubrimientos óptimos desde cada hijo; y si el nodo no pertenece entonces todos los hijos pertenecen, y el cubrimiento está formado por los hijos, más los cubrimientos óptimos desde los hijos de los hijos del nodo.

Como esas son las únicas alternativas, se pueden calcular los cubrimientos de forma bottom-up, usando programación dinámica en tiempo $O(|V|)$. La optimalidad se obtiene por inducción en la altura.

Problema 2

Se consideran expresiones booleanas, formadas por los siguientes símbolos:

$$\top, \bot, \wedge, \vee, \dot{\vee}$$

donde $\top$ indica verdadero, $\bot$ falso, $\wedge$ conjunción, $\vee$ disyunción, y $\dot{\vee}$ xor.

Luego, dada como input una secuencia de $2n - 1$ símbolos de ese conjunto, que comienza con $\bot$ o $\top$, y donde se alternan elementos de $\{\bot, \top\}$ con elementos de $\{\wedge, \vee, \dot{\vee}\}$, se busca determinar la cantidad de formas de evaluar la expresión tales que el resultado sea verdadero. Describa y analice un algoritmo para este problema.

Idea de solución Primero se enumeran los elementos de $\{\top, \bot\}$, en el orden en que aparecen. Luego, se definen las funciones *Verdadero* y *Falso*, donde *Verdadero*(i, j) corresponde a la cantidad de formas de hacer verdadera a la subfórmula entre los valores i y j, y similarmente *Falso*(i, j) es la cantidad de formas de hacerla falsa.

Luego, por cada forma de evaluar la subfórmula (i, j) debe existir un un conectivo que divide la formula en dos partes, tales que la evaluación sea equivalente a evaluar cada una de ellas, y luego utilizar el conectivo con los resultados. A este conectivo se le denominará *conectivo primario*.

Por lo descrito antes, *Verdadero*(i, j) y *Falso*(i, j) se definen inductivamente como la suma de las cantidades obtenidas al considerar cada conectivo como primario. Estas cantidades dependen del tipo de conectivo, y de las subfórmulas izquierda y derecha respectivas. Por ejemplo, si el segundo conectivo desde i es un $\wedge$, entonces se sumaría a *Verdadero*(i, j) el valor:

$$\textit{Verdadero}(i, i+1) \cdot \textit{Verdadero}(i+2, j),$$

y a $Falso(i, j)$:

$$Verdadero(i, i+1) \cdot Falso(i+2, j) +$$
$$Falso(i, i+1) \cdot Verdadero(i+2, j) +$$
$$Falso(i, i+1) \cdot Falso(i+2, j)$$

Finalmente, las funciones se pueden calcular usando programación dinámica, en el orden definido por los largos de las subsecuencias, y al terminar se retorna $Verdadero(1, n)$. El algoritmo funciona en tiempo $O(n^3)$.

Problema 3

Se considera el siguiente juego entre dos jugadores:

- Primero se genera una secuencia de n cartas, donde la carta i tiene valor $v_i \in \mathbb{N}$.
- Luego, los jugadores se turnan para sacar una carta de la secuencia, con la restricción que solo pueden elegir una carta de los extremos.
- Finalmente, gana el jugador cuyas cartas tengan la suma máxima.

Parte 1 Muestre que la estrategia *greedy*, de tomar la carta más alta no es siempre óptima.

Idea de solución 2,2,10,3.

Parte 2 Describa un algoritmo que dada la secuencia inicial permita construir una estrategia óptima para el primer jugador. La estrategia debe ser tal que en cada turno, el jugador decida cual extremo utilizar en tiempo $O(1)$.

En este problema se considera que una estrategia es óptima si maximiza el valor garantizado de la suma obtenida, independientemente de la estrategia del jugador 2.

Idea de solución Se consideran como subproblemas todos los subarreglos (i, j), y por cada uno se define $V(i, j)$ como la ganancia máxima que se puede asegurar cuando el jugador comienza desde ese subarreglo. Esta también debe considerar las opciones del jugador 2, y por lo tanto se puede definir inductivamente como:

$$V(i,j) = \begin{cases} v_j & j = i \\ \max\{v_i, v_j\} & j - i = 1 \\ \max\{v_i + \min\{V(i+2, j), V(i+1, j-1)\}, & \\ \quad v_j + \min\{V(i+1, j-1), V(i, j-2)\}\} & j - i > 1 \end{cases}$$

y luego calcular con programación dinámica de forma bottom-up.

La estrategia se puede calcular de forma simultánea en otra tabla, considerando la alternativa que maximiza cada caso. Por ejemplo, si

$$v_i + \min\{V(i+2, j), V(i+1, j-1)\} > v_j + \min\{V(i+1, j-1), V(i, j-2)\},$$

entonces la estrategia entre i y j es tomar la carta i.

El algoritmo funciona en tiempo $O(n^2)$, y en cada paso el jugador solo debe consultar una posición de la tabla con la estrategia, por lo que es $O(1)$.