Algoritmos y Estructuras de Datos II - 25 de junio de 2014
Recuperatorio del Segundo Parcial

Alumno: .....

Siempre se debe explicar la solución, una respuesta correcta no es suficiente sino viene acompañada de una justificación que demuestre que la misma ha sido comprendida. Las explicaciones deben ser completas. La utilización de código c o de nivel de abstracción excesivamente bajo influirá negativamente. En los ejercicios con varios incisos, por favor, no resuelvas varios incisos simultáneamente, sino cada uno por separado (pero no hace falta que sea en hojas aparte).

1. Para el problema de la moneda, mostrá con ejemplos las siguientes afirmaciones:

   a) el criterio de elegir en cada paso una moneda de la mayor denominación posible que no exceda el monto que resta pagar, no garantiza que el algoritmo voraz obtenga la solución óptima,

   b) el algoritmo recursivo (basado en backtracking) puede repetir cálculos,

   c) el algoritmo basado en programación dinámica puede realizar cálculos innecesarios (es decir, que el algoritmo del inciso anterior no haría).

2. Respondé separadamente cada una de las siguientes preguntas.

   a) ¿Qué hace el algoritmo de Prim?

   b) ¿Cómo lo hace?

   c) ¿Qué hace el algoritmo de Kruskal?

   d) ¿Cómo lo hace?

   e) ¿Cuáles son entonces las semejanzas y cuáles las diferencias entre ambos algoritmos?

3. Considerá una cuadrícula de n filas por k columnas donde cada celda tiene un costo $c_{i,j}$. Se comienza en la posición $(1,1)$ (celda superior izquierda) de la cuadrícula y se debe viajar hasta la celda (n,k) (inferior derecha). Sólo hay tres movimientos permitidos: hacia abajo (de (i,j) a $(i+1,j)$) o hacia la derecha (de (i,j) a $(i,j+1)$) o ambas a la vez (de (i,j) a $(i+1,j+1)$). Estos movimientos se permiten siempre que no se salga de la cuadrícula. El costo de un viaje es la suma de los costos de las celdas por las que se pasa, incluyendo $c_{1,1}$ y $c_{n,k}$. Escribí un algoritmo que utilice backtracking para calcular el costo del viaje de menor costo desde $(1,1)$ hasta (n,k). (Ayuda: si querés, podés utilizar la siguiente definición $m(i,j)$ = "costo del viaje de menor costo desde $(1,1)$ hasta (i,j) con movimientos permitidos".)

   Justificá con detalle cada una de las ecuaciones.

4. Para el problema de la mochila, se cuenta con la siguiente solución basada en backtracking. Sean $v_1, v_2, \ldots, v_n$ y $p_1, p_2, \ldots, p_n$ los valores y pesos de los n objetos, y sea W la capacidad de la mochila. No se asume que haya ningún orden particular entre los objetos. Se debe hallar el máximo valor posible sin superar la capacidad de la mochila.

   La solución con que se cuenta consiste en definir $m(i,j)$ = "máximo valor alcanzable con los objetos $1, 2, \ldots, i$ sin superar la capacidad j de la mochila" que da lugar a la siguiente definición recursiva:

$$m(i,j) = \begin{cases} 0 & \text{si } i = 0 \vee j = 0 \\ \text{máx}\{v_k + m(k-1, j-p_k) | 1 \leq k \leq i \wedge p_k \leq j\} & \text{si } j > 0 \wedge i > 0 \end{cases}$$

   Para entender la última ecuación pensá en la siguiente variante:

$$m(i,j) = \text{máx}\{v_1 + m(0, j-p_1),\ v_2 + m(1, j-p_2),\ v_3 + m(2, j-p_3),\ \ldots, v_i + m(i-1, j-p_i)\}$$

   Observá que cada elemento del conjunto es de la forma $v_k + m(k-1, j-p_k)$, donde $1 \leq k \leq i$. Esta definición no es exactamente equivalente a la de la ecuación de arriba porque, en realidad, **deben descartarse los términos** $v_k + m(k-1, j-p_k)$ **tales que** $j < p_k$.

   El ejercicio consiste en transformar **esta** solución en una que utilice programación dinámica. **No es válido recurrir a otra** solución al mismo problema.