Segundo Parcial de Programación 3

30 de noviembre de 2015

- Este parcial dura **4 horas** y contiene 3 carillas. El total de puntos es **60**.
- **NO** se puede utilizar ningún tipo de material de consulta. Salvo que se indique lo contrario podrá usarse todo lo visto en el teórico, práctico y laboratorio sin demostrarlo, indicando claramente lo que se está usando.

Se requiere:

I) Numerar todas las hojas e incluir en cada una el **nombre, cédula de identidad, número de página y cantidad de hojas entregadas**.

II) Utilizar las hojas de **un solo lado** y escribir con lápiz.

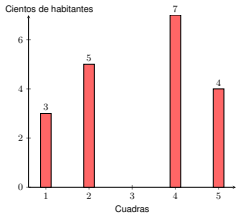
III) Iniciar cada ejercicio en hoja nueva.

IV) Poner en la carátula la cantidad de hojas entregadas y un índice indicando el número de página en la que comenzó a resolver cada problema.

Ejercicio 1 (20 puntos)

En la administración municipal se consideran las solicitudes para construir edificios en una nueva avenida de largo n cuadras, numeradas de 1 a n. Cada edificio estaría en el centro de una cuadra diferente. El edificio de la cuadra i-ésima alojaría h_i habitantes y si no hay una solicitud para esa cuadra se define $h_i = 0$. Tal vez no se puedan aceptar todas las solicitudes porque por razones de infraestructura (saneamiento, transporte, etc.) la distancia entre dos edificios, medida en cuadras, debe ser por lo menos d, con $1 \leq d \leq n$. Se pretende aceptar solicitudes de tal manera que se maximice la cantidad de habitantes en la avenida.

Ejemplo:
Sea n = 5, $h_1 = 300$, $h_2 = 500$, $h_3 = 0$, $h_4 = 700$, $h_5 = 400$.
Si $d = 3$ la solución óptima es $t = \langle 1, 4 \rangle$ que permite un total de $T = 1000$ habitantes.
Si $d = 1$ una solución óptima es $\langle 1, 2, 4, 5 \rangle$.



1. Se propone el siguiente algoritmo, en el que los edificios se consideran en orden decreciente de la cantidad de habitantes: aceptar el primero y a cada uno de los otros aceptarlo si cumple la restricción de la distancia con los edificios ya aceptados. En el ejemplo anterior, el orden en que se considerarían es $4, 2, 5, 1$.

   a) ¿De qué técnica vista en el curso es un ejemplo este algoritmo? Explique brevemente.

   b) Demuestre que siempre se puede obtener una solución óptima o exhiba un contraejemplo, lo más sencillo posible, en el que no ocurre.

2. De manera independiente a la respuesta del punto anterior, se debe resolver el problema mediante Programación Dinámica asumiendo que se cumple el Principio de Optimalidad. Se define $T(i)$ como el total de habitantes de una solución que optimice el subproblema definido para las cuadras $1, \ldots, i$. Se puede calcular $T(i)$ mediante una recurrencia.

   a) Exprese la recurrencia.

   b) Implemente la recurrencia para el problema $T(n)$ corrigiendo el siguiente algoritmo:

```
1  int total (int * h, int n) {
2     int T[n+1]; // El resultado del subproblema T(i) se mantiene en T[i]
3     T[0] = 0;
4     T[1] = h[1];
5     for (int i = 1; i <= n; i++)
6        if (h[i] > T[i-1])
7           T[i] = h[i] + T[i-d];
8        else
9           T[i] = T[i-1];
10    return T[n];
11 }
```

   Solamente puede agregar, suprimir o modificar líneas entre las líneas 3 y 9 incluidas. No puede usarse recursión ni estructuras de datos auxiliares.

Ejercicio 2 (20 puntos)

Un canal de televisión desea planificar el orden y contenido de las tandas publicitarias para el programa con más rating del canal. Para esto cuenta con un conjunto de reclames a trasmitir. La información se mantiene en el arreglo *reclames* entre los índices entre 1 y n. Los atributos de cada reclame son:

- *identificador*, que es la posición en el índice, está entre 1 y n;
- *duracion*, en cantidad de segundos, no mayor a 1 minuto;
- *minimo*, cantidad mínima de apariciones en todo el programa;
- *maximo*, cantidad máxima de apariciones en todo el programa, menor que 10.

La planificación debe cumplir lo siguiente:

A) Debe haber 5 tandas en el trancurso de todo el programa.

B) Cada tanda tiene que durar entre 4 y 5 minutos.

C) La cantidad total de minutos destinados a las tandas durante el programa no pueden superar los 23 minutos.

D) No puede emitirse el mismo reclame dos veces consecutivas en la misma tanda.

E) En cada tanda se deben emitir los reclames en orden decreciente (no estricto) de duración.

1. Defina una forma de tupla para el problema, considerando que cada componente represente una tanda.
2. Defina por extensión el dominio de las componentes de la tupla (liste los elementos que pertenecen al dominio), dado que los reclames son los siguientes:

		Apariciones	
Identificador	Duración (seg)	Minimo	Máximo
1	50	2	5
2	58	1	5
3	40	2	6
4	45	2	5
5	60	1	5
6	55	2	5

   Por ejemplo, ¿puede el reclame 2, seguido del 5, seguido del 4 ser una tanda válida?

3. Indique para cada una de las expresiones *A*-*E* si se trata de una Restricción Explícita, una Restricción Implícita, un Predicado de Poda o si queda considerada dentro de la forma de la solución. Exprese cada una en lenguaje formal.
4. Enuncie otras restricciones implícitas o explícitas y expréselas en lenguaje formal. Indique qué tipo de restricción es.
5. Proponga una función objetivo que maximice la facturación generada por el canal, considerando que el mismo cobra una cantidad F por segundo y que realiza descuentos según la cantidad de apariciones del reclame. El descuento para la aparición i-ésima es $(i-1) \times 10\,\%$ (por ejemplo, si un reclame aparece tres veces por la primera vez paga el 100 %, por la segunda 90 % y por la tercera 80 %).

Ejercicio 3 (20 puntos)

1. Explique cómo se construye el árbol de decisión de un algoritmo.
2. ¿Qué representan las hojas de un árbol de decisión de un algoritmo dado?
3. ¿Cuál es la altura mínima que puede tener un árbol de decisión que tiene m hojas?
4. ¿Qué representa un camino hasta una hoja en un árbol de decisión de un algoritmo dado?
5. ¿Qué representa la altura de un árbol de decisión de un algoritmo dado?
6. Dado un problema y todos los árboles de decisión correspondientes a los algoritmos que lo resuelven, ¿qué representa la mínima de las alturas de los árboles?

A partir de ahora se consideran solamente algoritmos de ordenación de n elementos no repetidos que comparan elementos de a pares.

7. Probar que para cualquier árbol de decisión de m hojas asociado a un algoritmo de ordenación se cumple: $m \geq n!$
8. Probar que cualquier algoritmo de ordenación debe en el peor caso hacer al menos $\lceil \log n! \rceil$ comparaciones.
9. ¿Cuál es la altura mínima de un árbol de decisión de un algoritmo que ordene 5 elementos?