Algoritmos y Estructuras de Datos II – 6 de agosto de 2014

# Examen Final Teórico-Práctico

Alumno: ................................................. Email: ...................................................

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).

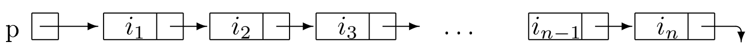
1. El TAD calcu_lista, define una calculadora de lista de la siguiente manera:

**TAD** calcu_lista
**constructores**
  vacía : calcu_lista
  agregar : entero × calcu_lista → calcu_lista
**operaciones**
  es_vacía : calcu_lista → booleano
  cabeza : calcu_lista → entero  {sólo se aplica a listas no vacías}
  cola : calcu_lista → calcu_lista  {sólo se aplica a listas no vacías}
  largo : calcu_lista → natural
  elemento : calcu_lista × natural → entero  {sólo se aplica a pares $(p, n)$ con $n <$ largo$(p)$}
  insertar : calcu_lista × natural × entero → calcu_lista  {sólo se aplica a ternas $(p, n, e)$ con $n \leq$ largo$(p)$}
  borrar : calcu_lista × natural → calcu_lista  {sólo se aplica a pares $(p, n)$ con $n <$ largo$(p)$}
  opuesto : calcu_lista × natural → calcu_lista
  mas : calcu_lista × natural → calcu_lista
**ecuaciones**
  es_vacía(vacía) = verdadero
  es_vacía(agregar(e,p)) = falso
  cabeza(agregar(e,p)) = e
  cola(agregar(e,p)) = p
  largo(vacía) = 0
  largo(agregar(e,p)) = 1 + largo(p)
  elemento(agregar(e,p),0) = e
  elemento(agregar(e,p),n+1) = elemento(p,n)
  insertar(p,0,e) = agregar(e,p)
  insertar(agregar(e',p),n+1,e) = agregar(e',insertar(p,n,e))
  borrar(agregar(e,p),0) = p
  borrar(agregar(e,p),n+1) = agregar(e,borrar(p,n))
  opuesto(vacía,n) = vacía
  opuesto(agregar(e,p),0) = agregar(-e,p)
  opuesto(agregar(e,p),n+1) = agregar(e,opuesto(p,n))
  mas(vacía,n) = vacía
  mas(agregar(e,vacía),n) = agregar(e,vacía)
  mas(agregar(e,agregar(e',p)),0) = agregar(e+e',p)
  mas(agregar(e,agregar(e',p)),n+1) = agregar(e,mas(agregar(e',p),n))

Implementá el TAD calcu_lista con una lista enlazada de enteros. Es decir, se define

**type** list_calc = **pointer to** node
**type** node = **tuple**
      value: **int**
      next: **pointer to** node
    **end**

Con esta representación, una calcu_lista $p$ de largo $n$ se puede graficar como




donde los $i_k$ son números enteros y la cabeza de la calcu_lista $p$ es $i_1$.

La implementación de las operaciones agregar, cola, insertar, borrar, opuesto y mas deben modificar la calcu_lista que reciben como argumento.

2. Dada la siguiente definición de la función $m(n, k)$:

$$m(i,j) = \begin{cases} 0 & i = 0 \\ 5 & i > 0 \wedge i \text{ es par} \wedge j = 0 \\ 4 & i > 0 \wedge i \text{ es impar} \wedge j = k \\ \text{mín}(2 + m(i-1, j), 3 + m(i, j \div 2)) & i > 0 \wedge \text{ es par} \wedge j > 0 \\ \text{mín}(2 + m(i-1, j), 3 + m(i, (j+k) \div 2)) & i > 0 \wedge i \text{ es impar} \wedge j < k \end{cases}$$

   donde $\div$ es la división entera (por ejemplo, $7 \div 2 = 3$). Escribí un algoritmo que calcule el valor de $m(n, 0) + m(n, k)$ utilizando programación dinámica. La dificultad consiste en hallar un orden de llenado de la matriz de manera tal que las celdas que se requieren para calcular el valor de la celda $m(i, j)$ hayan sido calculadas previamente.

   En caso de ser necesario, no dudes en ensayar la ejecución del algoritmo para $n$ y $k$ pequeños.

3. Resolvé la siguiente recurrencia

$$t(n) = \begin{cases} 0 & \text{si } n = 0 \\ n^2 - 2n + 1 + t(n-1) & \text{si } n > 0 \end{cases}$$

4. Se plantea la siguiente variante del problema de la mochila. Se cuenta con $n$ objetos de peso positivo $p_1, \ldots, p_n$ y se los quiere embalar en cajas de capacidad $P$. Se cuenta con $n$ cajas, que son suficientes ya que se asume que $p_i \leq P$ para todo $i$. De todas formas, se desea minimizar el número de cajas a emplear, colocando cuando sea posible varios o muchos objetos en cada caja, ya que los $n$ objetos serán enviados al mismo destinatario. El problema que se pide resolver, entonces, es dar un algoritmo que utilice backtracking para encontrar el menor número de cajas que se necesitan para embalar los $n$ objetos.

   Una posibilidad es definir $m(i, j_1, \ldots, j_n) =$ "menor número de cajas que se necesitan emplear para embalar los objetos $1, \ldots, i$ cuando las capacidades restantes de las cajas son $j_1, \ldots, j_n$."

   Observá que la función $m$ tiene, además del primer parámetro que se refiere al número de objetos considerados, $n$ parámetros (donde $n$ es el número de objetos totales) indicando cuánto resta ocupar de las $n$ cajas disponibles. Si uno de los $j_k$ es igual a $P$, significa que la $k$-ésima caja no fue aún utilizada.

   Podés utilizar esta definición para justificar una definición recursiva de $m$ y determinar cómo se utilizaría dicha definición para resolver el problema. No es imprescindible que lo pienses exactamente de esta manera, hay otras posibilidades.

5. (Para alumnos libres) Explicá el algoritmo de ordenación por intercalación. Cuáles son sus ventajas y cuáles sus desventajas. Cuál es su principal dificultad. Comentá sobre la posibilidad de definirlo recursivamente o iterativamente. Compararlo con otros algoritmos de ordenación que conozcas.