Algoritmos y Estructuras de Datos II – 29 de Julio de 2021

Examen Final Teórico-Práctico

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

Siempre se debe explicar la solución. Una respuesta correcta no es suficiente si no viene acompañada de una justificación lo más clara y completa posible. Los algoritmos no deben escribirse utilizando código c o de bajo nivel, sino el código de la materia y evitando la utilización innecesaria de punteros. La no observación de estas recomendaciones resta puntaje.

1. (Algoritmos voraces) Es viernes a las 18 y usted tiene ganas de tomar limonada con sus amigos. Hay n bares cerca, donde cada bar i tiene un precio P_i de la pinta de limonada y un horario de happy hour H_i, medido en horas a partir de las 18 (por ejemplo, si el happy hour del bar i es hasta las 19, entonces $H_i = 1$), en el cual la pinta costará un 50% menos. Usted toma una cantidad fija de 2 pintas por hora y no se considera el tiempo de moverse de un bar a otro. Escribir un algoritmo que obtenga el menor dinero posible que usted puede gastar para tomar limonada desde las 18 hasta las 02 am (es decir que usted tomará 16 pintas) eligiendo en cada hora el bar que más le convenga.

   Se pide lo siguiente:

   (a) Indicar de manera simple y concreta, cuál es el criterio de selección voraz para construir la solución?

   (b) Indicar qué estructuras de datos utilizarás para resolver el problema.

   (c) Explicar en palabras cómo resolverá el problema el algoritmo.

   (d) Implementar el algoritmo en el lenguaje de la materia de manera precisa.

2. Te encontrás frente a una máquina expendedora de café que tiene un letrero que indica claramente que la máquina "no da vuelto". Buscás en tu bolsillo y encontrás exactamente n monedas, con las siguientes denominaciones enteras positivas: $d_1, d_2, \ldots, d_n$. Una rápida cuenta te transmite tranquilidad: te alcanza para el ansiado café, que cuesta C. Teniendo en cuenta que la máquina no da vuelto, dar un algoritmo que determine el menor monto posible que sea mayor o igual al precio del café.

   (a) (Backtracking) Resolvé el problema utilizando la técnica de backtracking dando una función recursiva. Para ello:

   - Especificá precisamente qué calcula la función recursiva que resolverá el problema, indicando qué argumentos toma y la utilidad de cada uno.
   - Da la llamada o la expresión principal que resuelve el problema.
   - Definí la función en notación matemática.

   (b) (Programación dinámica) Implementá un algoritmo que utilice Programación Dinámica para resolver el problema.

   - ¿Qué dimensiones tiene la tabla que el algoritmo debe llenar?
   - ¿En qué orden se llena la misma?
   - ¿Se podría llenar de otra forma? En caso afirmativo indique cuál.

3. Para cada uno de los siguientes algoritmos determinar **por separado** cada uno de los siguientes incisos.

   (a) ¿Qué hace? ¿Cuáles son las precondiciones necesarias para haga eso?

   (b) ¿Cómo lo hace?

   (c) El orden del algoritmo, analizando los distintos casos posibles.

   (d) Proponer nombres más adecuados para los identificadores (de variables, funciones y procedimientos).

```
proc q(in/out a : array[1..N] of int, in x : nat)
    for j:= 1 to x do
        m:= j
        for k:= j+1 to x do
            if a[k] < a[m] then m:= k fi
        od
        swap(a,j,m)
    od
end proc
proc p(in/out a : array[1..N] of int, in i : nat)
    q(a, i-1)
    r(a, i+1)
end proc
```

```
proc r(in/out a : array[1..N] of int, in y : nat)
    for j:= y to n do
        m:= j
        while m > y ∧ a[m] < a[m-1] do
            swap(a,m,m-1)
            m:= m-1
        od
    od
end proc
```

4. Considere la siguiente especificación del tipo Listas de algún tipo T.

```
spec List of T where

constructors
    fun empty() ret l : List of T
    {- crea una lista vacía. -}

    proc addl (in e : T, in/out l : List of T)
    {- agrega el elemento e al comienzo de la lista l. -}

destroy
    proc destroy (in/out l : List of T)
    {- Libera memoria en caso que sea necesario. -}

operations
    fun is_empty(l : List of T) ret b : bool
    {- Devuelve True si l es vacía. -}

    fun head(l : List of T) ret e : T
    {- Devuelve el primer elemento de la lista l -}
    {- PRE: not is_empty(l) -}

    proc tail(in/out l : List of T)
    {- Elimina el primer elemento de la lista l -}
    {- PRE: not is_empty(l) -}

    proc addr (in/out l : List of T,in e : T)
    {- agrega el elemento e al final de la lista l. -}

    fun length(l : List of T) ret n : nat
    {- Devuelve la cantidad de elementos de la lista l -}

    proc concat(in/out l : List of T,in l0 : List of T)
    {- Agrega al final de l todos los elementos de l0
      en el mismo orden.-}

    fun index(l : List of T,n : nat) ret e : T
    {- Devuelve el n-ésimo elemento de la lista l -}
    {- PRE: length(l) > n -}

    proc take(in/out l : List of T,in n : nat)
    {- Deja en l sólo los primeros n
      elementos, eliminando el resto -}

    proc drop(in/out l : List of T,in n : nat)
    {- Elimina los primeros n elementos de l -}

    fun copy_list(l1 : List of T) ret l2 : List of T
    {- Copia todos los elementos de l1 en la nueva lista l2 -}
```

   (a) Utilizando como representación un arreglo de N elementos de tipo T y un natural, definí la estructura que representa la lista, implementá los constructores y las operaciones tail, concat, length y drop.

   (b) ¿La representación elegida para implementar el tad tiene alguna limitación? En caso afirmativo indicá cuál es. ¿Alguna operación debe tener una precondición extra?

   (c) ¿Qué orden tiene la operación tail implementada en el inciso anterior? ¿Se podría modificar la estructura de datos utilizada para que tail sea constante? Explicá cómo.

(d) Implementá una función que reciba una lista de enteros y devuelva otra lista que contenga sólo los elementos que sean pares. Para ello utilizá el tipo **abstracto**, sin acceder a su representación interna.

5. (Para alumnos libres) En un ABB cuyos nodos poseen valores entre 1 y 1000, interesa encontrar el número 363. ¿Cuáles de las siguientes secuencias NO puede ser una secuencia de nodos examinados según el algoritmo de búsqueda? ¿Por qué?

   (a) $2, 252, 401, 398, 330, 344, 397, 363$.

   (b) $924, 220, 911, 244, 898, 258, 362, 363$.

   (c) $925, 202, 911, 240, 912, 245, 363$.

   (d) $2, 399, 387, 219, 266, 382, 381, 278, 363$.

   (e) $935, 278, 347, 621, 299, 392, 358, 363$.