Algoritmos y Estructuras de Datos II – 28 de Julio de 2022
Examen Final Teórico-Práctico

Alumno: María Emilia Caldara Email: maemilia.caldara@mi.unc.edu.ar

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.

Realizar los ejercicios EN HOJAS SEPARADAS.

1. (Algoritmos voraces) Es el siglo XX y no existe Netflix ni ningún servicio de TV por demanda. En esta época la gente contrata servicio de TV por cable donde te envían un librito con la programación de cada canal día por día.

   Como el fin de semana estará lloviendo, planeás encerrarte a ver películas. Del librito de programación seleccionaste n películas que te interesan (que se transmiten en distintos canales) y para cada película i, con $1 <= i <= n$, tenés el horario de comienzo c_i y de final f_i. Por supuesto no podés ver dos películas a la vez.

   Debés encontrar cuáles de las n películas vas a ver, de manera que la cantidad sea máxima. 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. Ya pasaste los 35 años y al fin te ponés las pilas para hacer ejercicio físico. En el gimnasio tenés un plan con n ejercicios donde cada ejercicio i, con $1 <= i <= n$ tiene asociado un "valor de entrenamiento general" e_i. Además, cada ejercicio i requiere un esfuerzo de brazos b_i, un esfuerzo de zona media z_i y un esfuerzo de piernas p_i.

   Se debe encontrar el máximo valor de entrenamiento general obtenible al elegir ejercicios sin que el esfuerzo total en brazos supere el monto B, el esfuerzo total en zona media supere el monto Z y el esfuerzo total en piernas supere el monto P.

   (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 Pila de elementos de tipo T:

```
spec Stack of T where

constructors
    fun empty_stack() ret s : Stack of T
    {- crea una pila vacía. -}

    proc push (in e : T, in/out s : Stack of T)
    {- agrega el elemento e al tope de la pila s. -}

destructors
    proc destroy(in/out s : Stack of T)

copy
    fun copy(s : Stack of T) ret s : Stack of T

operations
    fun is_empty_stack(s : Stack of T) ret b : Bool
    {- Devuelve True si la pila es vacía -}

    fun top(s : Stack of T) ret e : T
    {- Devuelve el elemento que se encuentra en el tope de s. -}
    {- PRE: not is_empty_stack(s) -}

    proc pop (in/out s : Stack of T)
    {- Elimina el elemento que se encuentra en el tope de s. -}
    {- PRE: not is_empty_stack(s) -}
```

   (a) Utilizando como representación un arreglo de N elementos y un número natural, implementá los constructores y todas las operaciones indicando el orden de complejidad de cada una.

   (b) ¿La representación elegida tiene alguna limitación? Si es así, ¿cuál? Justificá la respuesta.

   (c) ¿Podría implementarse el tipo Pila utilizando como representación interna un conjunto de elementos? Justificá la respuesta.

   (d) Utilizando el tipo **abstracto**, implementá un procedimiento *invert* que invierta los elementos de una pila de elementos de tipo T. ¿Qué orden de complejidad tiene la implementación?