Algoritmos y Estructuras de Datos II – 16 de Diciembre 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)

   Dado un grafo dirigido G con costos no negativos en sus aristas, representado por su matriz de adyacencia, y un vértice v del mismo, el algoritmo de Dijkstra calcula, para cada vértice w del grafo G, el costo del camino de costo mínimo de v a w.

   (a) De qué manera podés modificar el algoritmo de Dijkstra (llamémosle algoritmo de Artskjid) para que en lugar de calcular costos de caminos desde v, calcule costos de caminos hacia v. Es decir, para que dado G tal como se dijo, y dado un vértice v del mismo, calcule, para cada vértice w del grafo G, el costo del camino de costo mínimo de w a v. Escribí el algoritmo.

   (b) ¿Cómo podrías utilizar los algoritmos de Dijkstra y de Artskjid para calcular, para cada vértice w de G el costo del camino de costo mínimo de ida y vuelta de v a w. Incluso si no resolviste el inciso anterior, podés intentar resolver éste utilizando ambos algoritmos.

2. (Backtracking) Como sabés que esta medianoche aumentarán todos los precios, decidís gastar hoy mismo la mayor parte posible del dinero D de que disponés. Hay n objetos para comprar, cuyos precios hoy son $v_1, v_2, \ldots, v_n$. Suponemos que $D < \sum_{i=1}^{n} v_i$, por lo que deberás elegir cuáles objetos comprar, es imposible comprar todos. Se debe determinar el máximo monto que podés gastar sin exceder el dinero D disponible.

   Se pide lo siguiente:

   (a) Especificá precisamente qué calcula la función recursiva que resolverá el problema, indicando qué argumentos toma y la utilidad de cada uno.

   (b) Da la llamada o la expresión principal que resuelve el problema.

   (c) Definí la función en notación matemática.

3. (Programación Dinámica) Dados $c_1, c_2, \ldots, c_n$ y la siguiente definición recursiva de la función m, para $1 \leq j \leq i \leq n$, escribí un programa que utilice la técnica de programación dinámica para calcular el valor de $m(n, 1)$.

$$m(i,j) = \begin{cases} c_i & \text{si } i = j \\ m(i-1,j) + m(i,j+1) & \text{si } i > j \end{cases}$$

   Ayuda: Antes de comenzar, hacé un ejemplo para entender la definición recursiva. Podés tomar por caso $n = 4, c_1 = 3, c_2 = 1, c_3 = 2, c_4 = 5$, y calcular $m(4, 1)$.

4. 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).

```
fun f(a: array[1..n] of nat, i,j,x:nat) ret b: bool
    var d,e,f : nat
    d,f,b := i,j,false
    while (d ≤ f) do
          e:= d+f div 2
          if x < a[e] → f:= e-1
          [] x = a[e] → b:= true
          [] x > a[e] → d:= e+1
          fi
    od
end fun
```

```
fun g(a,b: array[1..n] of nat)
                                  ret c: array[1..n] of bool
    var i : nat
    i := 1
    while (i ≤ n) do
          c[i]:= f(a, 1, n, b[i])
          i:= i+1
    od
end fun
```

5. El TAD VipQueue es una variante del tipo abstracto Queue que cuenta con un constructor adicional **enqueueVip**, el cual modifica la VipQueue agregándole un elemento de modo "preferencial" o "vip". También cuenta con una operación adicional **hayVip** que indica si en la VipQueue hay algún elemento que se haya agregado de modo vip.

   El resto de las operaciones tienen el mismo tipo que en la versión Queue original pero su comportamiento es modificado:

   La operación **first** devuelve el primer elemento que haya ingresado como vip, o el primer elemento que haya ingresado de modo normal, en caso que no haya ningún vip. La operación **dequeue** devuelve la VipQueue que resulta de eliminar el elemento que la operación **first** devolvería.

   Se pide:

   (a) Escribí la especificación completa del tipo VipQueue.

   (b) Implementá el tipo VipQueue utilizando una estructura que contenga dos arreglos de tamaño N y dos números naturales.

6. (Para alumnos libres)

   (a) Graficá todos los árboles finitarios que al recorrerse en pre-orden o en BFS sus vértices resultan visitados en el siguiente idéntico orden: eva, maría, juan, josé, ana.

   (b) Graficá todos los árboles finitarios que al recorrerse en **pos-orden** (no confundir con pre-orden), sus vértices resultan visitados en el siguiente orden: juan, josé, ana, eva; pero que al recorrerse en BFS, sus vértices resultan visitados en el siguiente orden: eva, juan, josé, ana.