Examen de Programación 3

15 de julio de 2016

La prueba es individual y sin material. La duración es 4 hs. Escriba su cédula y nombre en cada hoja.

Parte obligatoria

Esta parte es eliminatoria. Para la aprobación del examen debe obtenerse un mínimo del **50 % de esta parte (20 puntos)**. En caso de no llegar a dicho mínimo, **NO** se corregirán los problemas.

Ejercicio 1 (7 puntos)

Sea A un arreglo de enteros, con índices de 1 a n, que se pasa como parámetro al algoritmo que se muestra a la derecha. Se considera que la operación básica es la asignación.

(a) ¿Cuándo se da el peor caso en el algoritmo y por qué?

(b) Sea $T(n)$ el tiempo de ejecución del algoritmo. Indique para cada uno de las siguientes órdenes si $T(n)$ pertenece a ese orden: $\Omega(n)$, $O(2n)$, $O(n^2)$, $\Theta(n^2)$. No hace falta justificar.

```
a ← A[n]
FOR i = n - 1 DOWNTO 1
    IF A[i] < a
        a ← A[i]
b ← A[1]
FOR i = 2 TO n
    IF A[i] > b
        b ← A[i]
RETURN (a,b)
```

Ejercicio 2 (11 puntos)

(a) Describa brevemente el algoritmo `mergesort`. ¿Cuál es el orden exacto de su tiempo de ejecución?

(b) Mediante árboles de decisión se demuestra que $\log n!$ es una cota en los algoritmos de ordenación mediante comparaciones. Como consecuencia de esto, ¿qué se puede afirmar, en términos de orden, de la relación entre $\log n!$ y $T(mergesort)$?

(c) Demuestre, complementando el ítem b que $T(mergesort) \in \Theta(\log n!)$

Ayuda: Puede usar, asumiéndolas como demostradas, las siguientes expresiones:

$$\log n! \in \Omega(n) \tag{1}$$

$$n \log n \leq \log n! + 1{,}5n \tag{2}$$

Ejercicio 3 (11 puntos)

Sean $G = (V, E)$ un grafo dirigido y u un vértice de V.

(a) Defina la componente fuertemente conexa (CFC) de u.

(b) Considere las siguientes funciones del TAD Grafo (extendido):

- DFS: Grafo × Vértice → Grafo // Árbol
- Transpuesto: Grafo → Grafo
- Inducido: Grafo × Conjunto de Vértices → Grafo
- Vertices: Grafo → Conjunto de Vértices

I. Escriba el algoritmo de la función `Inducido_DFS(G,u)` que devuelve el subgrafo de G inducido por los vértices alcanzados por una recorrida `DFS` iniciada en u.

II. Escriba un algoritmo que obtenga la CFC de u en G.

En cada ítem se pueden usar las funciones de todos los ítems anteriores.

Ejercicio 4 (11 puntos)

Considere los siguientes problemas vistos en el curso: Dijkstra, Búsqueda binaria, Floyd, Suma de subconjuntos, Prim. Seleccione exactamente 3. Para cada uno explique qué es lo que resuelve y con cuál de las técnicas de diseño de algoritmos estudiadas en el curso.

Problemas

Problema A (25 puntos)

Florentina quiere comprar libros por internet. Tiene una lista de 1000 libros que desea (identificados del 1 al 1000), la cual va a usar para seleccionar los que va a comprar. De cada libro i se conoce su peso p_i (en kg), su costo c_i (en dólares) y un valor v_i del 1 al 100 que indica el interés que tiene Florentina por ese libro (a mayor número, mayor es el interés). Además se sabe que algunos de esos libros pertenecen a una colección dada, y que hay M colecciones en la lista (identificadas del 1 al M). Para cada libro se conoce a qué colección pertenece o se sabe que no pertenece a ninguna ($col_i \in \{0, \ldots, M\}$ siendo 0 cuando no pertenece a ninguna colección). Si un libro pertenece a una colección, para comprarlo se deben comprar los anteriores de la colección correspondiente. El orden de la lista respeta el orden de los libros en las colecciones (si aparece un libro en la lista, todos los anteriores de su colección aparecen previamente). La matriz $Orden$ indica el orden de los libros en las colecciones ($Orden[j, k] = i$ indica que el k-ésimo elemento en la colección j, es el libro i). A lo sumo se debe comprar una copia de cada libro.

Florentina tiene un presupuesto máximo de U$S200, y un límite de 5kg de peso total. Cada libro le cuesta U$S5 de flete.

Se quiere maximizar el valor (de interés) de los libros comprados, minimizando el costo (lo prioritario es maximizar el interés, y a igual interés, se elige la solución más económica).

Resuelva el problema usando la técnica **Backtracking**: exprese en lenguaje natural y en lenguaje formal la forma de la tupla, restricciones explícitas, restricciones implícitas y función objetivo en el caso que correspondan.

Problema B (20 puntos)

(I) Sea G un grafo simple no dirigido. Los vértices de G se identifican con enteros del 0 al $n-1$ y el grafo se representa mediante un arreglo de listas de adyacencia.

Implemente la siguiente función no recursiva:

```
bool asignar_paridad(int u, int n, Lista * adyacentes, int * & paridad);
```

En el arreglo `paridad` se debe indicar si la distancia desde u hasta los vértices de su misma componente es par o impar. Concretamente: $paridad[v]$ debe ser 0 si la distancia es par, 1 si es impar o -1 si v no pertenece a la misma componente de u.

Además, se debe devolver `true` si en la componente de u no hay aristas entre vértices de igual paridad; en otro caso se debe devolver `false`.

(II) Un grafo $G = (V, E)$ es bipartito si el conjunto de vértices se puede particionar en dos subconjuntos de tal manera que cada arista tenga un extremo en uno de los subconjuntos y el otro extremo en el otro subconjunto ¿Cómo se puede determinar si una componente de un grafo es bipartita usando la función de la parte anterior?

Problema C (15 puntos)

Martín, que es fanático de los programas de televisión, ganó un sorteo en el cual puede llevarse todo lo que quiera de un supermercado hasta un máximo $W = 10.000$ gramos. En el supermercado se sabe que hay n productos distintos, teniendo cada uno un precio d_i y un peso w_i (en gramos) conocidos (ambos números naturales), $i \in \{1, \ldots, n\}$. Para simplificar y sin pérdida de generalidad se asume que hay disponible un solo ítem de cada producto. Martín es astuto y desea revender lo que se lleve del supermercado, entonces quiere llevar la mayor cantidad de dinero en productos posible.

(I) ¿Qué técnica de diseño de algoritmos vista en el curso nos permite resolver de forma óptima y eficiente el problema de determinar cuánto es la ganancia máxima que puede obtener? Especificar matemáticamente el problema y la solución para resolverlo.

(II) Implementar en C* el algoritmo de la parte I. Asuma que se tienen disponibles las funciones *max* y *min*, que devuelven el máximo y mínimo de dos números enteros, respectivamente.

(III) Si se pudiera llevar partes fraccionadas de productos (por ejemplo, cantidad 0,243 de una botella con agua), indicar, justificando brevemente, una manera más eficiente de resolver el problema vista en el curso.