Algoritmos y Estructuras de Datos II
Segundo parcial – 20/06/2009

Aclaraciones

- El parcial **NO** es a libro abierto.
- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Incluya el número de orden asignado (léalo cuando circule la planilla), apellido y nombre en cada hoja.
- Al entregar el parcial complete los datos faltantes en la planilla.
- Cada ejercicio se calificará con B, R ó M. Una B no significa que el ejercicio está "perfecto", sino que cumple con los requisitos necesarios para aprobar. En los parciales promocionados se asignará una nota numérica más precisa a cada ejercicio.
- Para aprobar el parcial debe obtenerse B en el ejercicio 1 y en alguno de los ejercicios 2 y 3. Un parcial se considera promocionado si está aprobado y su puntaje es 70 o superior.

Ej. 1. Diseño

Se desea diseñar un sistema para monitorear una planta industrial que cuenta con un conjunto de alarmas asociadas a distintos sensores. Cada sensor está asociado a una única alarma pero cada una de estas puede ser activada por distintos sensores. Una alarma está activa cuando la medición de al menos uno de sus sensores asociados supera un valor umbral definido para ese sensor. Los sensores y las alarmas se identifican con un código alfanumérico (para los sensores puede asumir que este es de longitud acotada, mientras que para las alarmas no)

El siguiente TAD es una especificación para este problema.

TAD PLANTA

géneros planta

observadores básicos

esAlarma : planta × alarma ⟶ bool

esSensor : planta × sensor ⟶ bool

alarmaSensor : planta p × sensor s ⟶ alarma — esSensor(p,s)

umbral : planta p × sensor s ⟶ nat — esSensor(p,s)

medicion : planta p × sensor s ⟶ nat — esSensor(p,s)

generadores

crear : conj(alarma) ⟶ planta

agSensor : planta p × sensor s × nat u × alarma a ⟶ planta — ¬ esSensor(p,s) ∧ esAlarma(p,a) ∧ $u > 0$

nuevaMedicion : planta p × sensor s × nat ⟶ planta — esSensor(p,s)

otras operaciones

encendidosPorAlarma : planta p × alarma a ⟶ conj(sensor) — esAlarma(p,a)

encendida : planta p × alarma a ⟶ bool — esAlarma(p,a)

axiomas

esAlarma(crear(c), a) ≡ $a \in c$

esAlarma(agSensor(p, s, u, a'), a) ≡ esAlarma(p, a)

esAlarma(nuevaMedicion(p, s, n), a) ≡ esAlarma(p, a)

esSensor(crear(c), s) ≡ false

esSensor(agSensor(p, s', u, a), s) ≡ $s = s' \vee$ esSensor(p, s)

esSensor(nuevaMedicion(p, s', n), s) ≡ esSensor(p, s)

alarmaSensor(agSensor(p, s', u, a), s) ≡ **if** $s = s'$ **then** a **else** alarmaSensor(p, a) **fi**

alarmaSensor(nuevaMedicion(p, s', n), s) ≡ alarmaSensor(p, s)

umbral(agSensor(p, s', u, a), s) ≡ **if** $s = s'$ **then** u **else** umbral(p, a) **fi**

umbral(nuevaMedicion(p, s', n), s) ≡ umbral(p, s)

medicion(agSensor(p, s', u, a), s) ≡ **if** $s = s'$ **then** 0 **else** medicion(p, a) **fi**

medicion(nuevaMedicion(p, s', n), s) ≡ **if** $s = s'$ **then** n **else** medicion(p, a) **fi**

encendidosPorAlarma(crear(c), a) ≡ $\emptyset$

encendidosPorAlarma(agSensor(p, s, u, a'), a) ≡ encendidosPorAlarma(p, a)

encendidosPorAlarma(nuevaMedicion(p, s, n), a) ≡ **if** alarmaSensor(p, s) $= a \wedge n >$ umbral(p, s) **then** $Ag(s,$ encendidosPorAlarma(p, a)) **else** encendidosPorAlarma(p, a) $- \{s\}$ **fi**

encendida(p, a) ≡ #encendidosPorAlarma(p, a) > 0

Fin TAD

Se desea diseñar el sistema propuesto, teniendo en cuenta que la operación *nuevaMedicion* debe realizarse con complejidad temporal $O(1)$ y *encendida* con complejidad temporal $O(l * a)$ en el peor caso, donde a es la cantidad de alarmas de la planta y l es la longitud del identificador de alarma más largo.

Se pide:

a) Describir la estructura a utilizar, documentando claramente cómo la estructura resuelve el problema y cómo cumple con los requerimientos de eficiencia. El diseño debe incluir sólo la estructura de nivel superior. (De ser necesario para justificar los órdenes de complejidad, describa las estructuras soporte.)

b) Escribir una versión en lenguaje imperativo del algoritmo *nuevaMedición*, indicando la complejidad de cada uno de los pasos. Tener en cuenta que esta operación puede activar una alarma (cuando la medición supera el umbral) o desactivarla (cuando la medición pasa a ser menor que el umbral y no hay otros sensores activándola).

Ej. 2. Sorting

Se desean ordenar los datos generados por un sensor industrial que monitorea la presencia de una determinada sustancia en un proceso químico. Cada una de estas mediciones es un número entero positivo. Dada la naturaleza del proceso se sabe que, dada una secuencia de n mediciones, a lo sumo $\lfloor \sqrt{n} \rfloor$ valores están fuera del rango $[20, 40]$.

a) Proponer un algoritmo que permita ordenar ascendentemente una secuencia de mediciones con las características anteriores y cuya complejidad temporal sea $O(n)$ en el peor caso, donde n es la longitud de la secuencia.
b) Mostrar que la complejidad del algoritmo propuesto es $O(n)$ en el peor caso, justificando claramente su respuesta.
c) Suponer que se cuenta con un registro histórico de n mediciones ordenado ascendentemente (pero no se tiene información sobre la distribución de los datos) y una secuencia no ordenada de $\lfloor\sqrt{n}\rfloor$ valores que se distribuyen como en el inciso *a*. Proponer un algoritmo que permita mostrar por pantalla en orden ascendente los $n + \lfloor\sqrt{n}\rfloor$ elementos y cuya complejidad temporal sea $O(n)$ en el peor caso.
d) Mostrar que la complejidad del algoritmo propuesto en el punto anterior es efectivamente $O(n)$.

Ej. 3. Divide & Conquer

a) Dada una matriz cuadrada A de orden 4×4 y un número entero n, proponer, utilizando la técnica de Divide & Conquer, un algoritmo *potencia* que permita calcular A^n y cuya complejidad temporal sea sublineal respecto de n.
b) Dar la complejidad temporal del algoritmo propuesto, justificando su respuesta.
c) Dada una matriz cuadrada A de orden 4×4, y un número entero n que es una potencia de 2 (esto es $n = 2^k$), desarrollar, utilizando la técnica de Divide & Conquer y el algoritmo *potencia* propuesto anteriormente, un algoritmo que permita calcular

$$A^1 + A^2 + \ldots + A^n$$

y cuya complejidad temporal sea sublineal respecto de n.
d) Estimar el número de veces que se aplicará la operación *potencia* cuando se calcule $A^1 + A^2 + \ldots + A^n$ usando el algoritmo propuesto en el punto *c*).