UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I

2do parcial – 15 de Noviembre 2019

Nro. de orden: 30
LU: 351/17
Apellidos: YULITA
Nombres: FEDERICO
Nro. de hojas que adjunta: 4

1			2	3	4	TOTAL
a	b	c				
5	5	10	25	30	25	100

(A)

Aclaraciones: Se permite tener UNA hoja A4 con anotaciones durante el parcial. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos.
Entregar cada ejercicio en una hoja separada, numerada y que incluya el nro. de orden.

Ejercicio 1. [20 puntos]
Dado el siguiente programa para determinar si la suma de los elementos pares de un arreglo es mayor estricto al entero k:

```
bool sumaMayor(vector<int> v, int k){
  int suma = 0;
  for(int i = 0; i < v.size(); i++){
    if(v[i] % 2 == 1){
      suma = suma + v[i];
    }
  }
  return (suma > k);
}
```

a) **[5 puntos]** Describir el diagrama de control de flujo del programa.

b) **[5 puntos]** Este programa tiene un error. Decir cuál es y escribir un caso de test que exhiba el defecto.

c) **[10 puntos]** Solucionar el error del item anterior, y escribir un test suite que cubra todas las lineas (sentencias) pero no todas las ramas (branches) del programa.

Ejercicio 2. [25 puntos] Dada la siguiente especificación se pide escribir una implementación cuyo único ciclo respete el invariante I. Justificar todas las decisiones.

Especificación

proc hayInterseccion (in s: $seq\langle\mathbb{Z}\rangle$, in t: $seq\langle\mathbb{Z}\rangle$, out res: Bool) {
  Pre $\{creciente Estricta(s) \land creciente Estricta(t)\}$
  Post $\{res = \text{true} \leftrightarrow (\exists n : \mathbb{Z})(\exists m : \mathbb{Z})(0 \leq n < |s| \land 0 \leq m < |t| \land_L s[n] = t[m])\}$
  pred crecienteEstricta (s: $seq\langle\mathbb{Z}\rangle$) {
    $(\forall i : \mathbb{Z})(0 \leq i < |s| - 1 \rightarrow_L s[i] < s[i+1])$
  }
}

$$I = 0 \leq i \leq |s| \ \land 0 \leq j \leq |t| \land (res = \text{true} \leftrightarrow (\exists n : \mathbb{Z})(\exists m : \mathbb{Z})(0 \leq n < i \land 0 \leq m < j \land_L s[n] = t[m]))$$

Ejercicio 3. [30 puntos] Dada la siguiente especificación, escribir una implementación cuyo tiempo de ejecución de peor caso sea $O(n + |s|)$. Justificar.

proc algúnProductoEsDividido (in s: $seq\langle\mathbb{Z}\rangle$, in n: $\mathbb{Z}$, out res: Bool) {
  Pre { $|s| >= 2 \ \land \ n > 0 \ \land esProductoDeDosPrimos(n)$ }
  Post { $res = true \ \leftrightarrow \ (\exists i : \mathbb{Z})(\exists j : \mathbb{Z})(0 \leq i < j < |s| \ \land \ (s[i] * s[j]) \ mod \ n = 0)$ }
}
pred esProductoDeDosPrimos (n: $\mathbb{Z}$) {
  $(\exists p : \mathbb{Z})(\exists q : \mathbb{Z})((p \neq q) \land \ esPrimo(p) \ \land \ esPrimo(q) \ \land \ n = p * q \ )$
}

Ejercicio 4. [25 puntos] Se tienen dos arreglos, A y B. Se sabe que los elementos de B son todos distintos entre sí. Si A tiene exactamente n elementos, y B tiene exactamente n^2 elementos, dar un algoritmo que indique cuántos elementos de B también aparecen en A. El tiempo de ejecución del algoritmo debe ser estrictamente menor que $O(n^3)$. Por ejemplo, si A=(1,3,2) y B=(1,2,4,5,16,7,15,20,19), el algoritmo debe retornar 2 (pués coinciden en los elementos 1 y 2). Ayuda: Pueden usar la función ordenar sin programarla y suponer que es O(n*log(n))

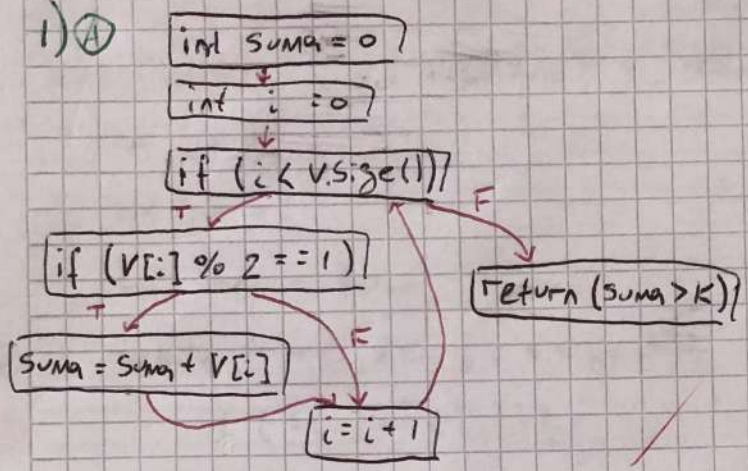
FEDERICO YULITA 351/17

(30)

HOJA N°

FECHA

1) (A)



Este CFG tiene 7 nodos y 8 ramas. Partí al for en una declaración - inicialización, un condicional y un i++ por separado.

(B) El error de este programa es que suma los números impares en vez de los pares, debería decir if (V[i] % 2 == 0). Un posible test case que exhibe este error es (in: V=[1,1], K=1; out: False), ya que este programa va a sumar los impares, 1+1, y va a devolver True porque 2 > 1.

(C) El programa correcto es:

```
bool sumaMayor (vector<int> V, int K){
    int suma = 0
    for (int i = 0; i < V.size(); i++){
        if (V[i] % 2 == 0){
            suma = suma + V[i];
        }
    }
    return (suma > K);
}
```

; un posible test suite es:
; • (in: V = [2,2], K = 2; out: True)
; Este test suite recorre todas las líneas porque tiene números pares pero no todas las decisiones porque no hay números impares.

FEDERICO YULITA 351/17

(30)

HOJA N°

FECHA

2)

```
bool hayInterseccion(vector<int> s, vector<int> t){
    int i = 0;
    int j = 0;
    bool res = false;
(*) while (!res && i < s.size()){
(*)     if (s[i] == t[j]){
            res = true;
            i++;
            j++;
(*)     } else if (j == t.size() - 1){
            i++;
            j = 0;
        } else {
            j++;
        }

    }
    return res;
}
```

(*) Una vez que encuentre algún elemento común el ciclo termina o si no hay ninguno quiero que el ciclo termine cuando termine de recorrer todos los elementos. En este caso, eso sucede cuando i = s.size().

(*) Si encuentro un elemento común entonces hago que res = true y además incremento i y j para cumplir el invariante, ya que se fija en $n < i \not\wedge m < j$.

(*) El programa recorre todo t para cada elemento de s, por eso quiero que si j = t.size() - 1 que pase al siguiente elemento de s y que vuelva al principio de t. Por eso, de no haber un elemento común, el programa termina al recorrer todo s.

Ahora me doy cuenta que no hice uso del hecho de que las listas estén ordenadas, pero bue. Esto funca.

3) Voy a usar dos funciones: Una que busque un primo tal que n mod p = 0 y otra que se fije si existen números que tengan a p o a q = n/p como divisor. Con eso me alcanza, ya que entonces al multiplicarlos se que el resultado va a tener a p y a q al menos una vez en su producto de primos y entonces el resultado es divisible por n. La primera función será de tiempo $\Theta(n)$ y la segunda de tiempo $\Theta(|s|)$. La implementación es la siguiente:

```
int buscarPrimo(int n){
    for(int p = 2; p<n; p++){
        if(n%p==0){
            return p;
        }
    }
}
bool estanLosPrimos(vector<int> s, int p, int q){
    bool estaP = false;
    bool estaQ = false;
    for(int i=0; i<s.size(); i++){
        if(s[i]%p==0){
            estaP=true;
        }
        if(s[i]%q==0){
            estaQ=true;
        }
    }
    return (estaP&&estaQ);
}
```

```
bool algunProductoEsDivisible(vector<int> s, int n){
    int p = buscarPrimo(n);
    int q = n / p;
    return estanLosPrimos(s, p, q);
}
```

FEDERICO YULITA 351/17

HOJA Nº

FECHA

(3c)

i) La estrategia sería ordenar A y luego hacer una búsqueda binaria de cada elemento de B en A. El tiempo de ejecución sería:

$T \propto n\log(n) + n^2\log(n) = (n^2+n)\log(n)$

$\Rightarrow T \in \Theta(n^2\log(n))$

La implementación es la siguiente:

```
bool busquedaBinaria(vector<int> &S, int x){
    if (S.size() == 0){
        return false;
    }else if (S.size() == 1){
        return S[0] == x;
    }else if (x >= S[S.size()-1]){
        return x == S[S.size()-1];
    }else if (x <= S[0]){
        return x == S[0];
    }else{
        int low = 0;
        int high = S.size()-1;
        while (low+1 < high){
            int mid = (low+high)/2;
            if (S[mid] <= x){
                low = mid;
            }else{
                high = mid;
            }
        }
        return S[low] == x;
    }
}
```

NOTA

```
int cuantosIguales(vector<int> &A, vector<int> B){
    ordenar(A);
    int suma = 0;
    for(int i=0; i<B.size(); i++){
        if(busquedaBinaria(A, B[i]){
            suma++;
        }
    }
    return suma;
}
```