Escuela de Ingeniería y Arquitectura - Depto. de Informática e Ingeniería de Sistemas

Examen práctico de Programación 2 - 4 de junio de 2015 - Turno 1º - Tiempo: hasta las 16:50

Este examen práctico individual en laboratorio forma parte de la evaluación de la asignatura. Su calificación tiene un peso del 15 % en la calificación final de la asignatura en la convocatoria de junio.

PRIMERA PARTE [6.0 puntos]

Se ha de diseñar la función ***contarRepetidos****(v,n,rep)* que se especifica a continuación sin programar ni un solo bucle. Podrán añadirse las funciones auxiliares que se precisen. Cada una de estas deberá estar convenientemente especificada.

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].rep[alfa]=(Núm beta EN [0,n-1].v[alfa]=v[beta]))
 */
void contarRepetidos (const int v[], int n, int rep[]);
```

SEGUNDA PARTE [4.0 puntos]

Se ha de diseñar la función ***contarRepetidosIter****(v,n,rep)* que se especifica a continuación. Su coste en tiempo ha de ser $\boldsymbol{O}(t(n)) = \boldsymbol{O}(n^2)$. Esta función no debe presentar invocaciones a funciones auxiliares ni invocaciones recursivas. En cambio se pueden programar instrucciones iterativas. Se pide anotar los datos y deducciones, sobre el coste en tiempo de ejecutar esta función en el computador ***hendrix-ssh***, que se describen en el reverso de esta hoja.

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].rep[alfa]=(Núm beta EN [0,n-1].v[alfa]=v[beta]))
 */
void contarRepetidosIter (const int v[], int n, int rep[]);
```

PRESENTACIÓN DEL TRABAJO

Como resultado del trabajo se entregará:

- Un único fichero cuyas primeras líneas sean un comentario con el nombre del alumno y, a continuación, almacene exclusivamente el código de las funciones que integren el diseño completo de ***contarRepetidos****(v,n,rep)* así como el diseño completo de la función ***contarRepetidosIter****(v,n,rep)*. La entrega se hará a través de la plataforma ***Moodle2*** (moodle2.unizar.es) antes de las 16:50. Se recomienda no hacerlo hasta que haya sido verificado de forma exhaustiva que el comportamiento de todo código diseñado es correcto.
- Rellenar la página posterior de esta hoja, que se entregará al profesor, en la que se deben anotar los datos de costes obtenidos en el computador ***hendrix-ssh*** y se deben responder las cuestiones planteadas en ella.

Examen práctico de Programación 2 - 4 de junio de 2015 - Turno 1º

Entregar esta hoja al profesor al finalizar el examen, debidamente cumplimentada.

Nombre y apellidos: _____

1. DATOS EXPERIMENTALES DE COSTE

Anote en la siguiente tabla los datos experimentales del coste, en tiempo $t(n)$, de ejecutar la función ***contarRepetidosIter****(v,n,rep)* para cuatro valores de n comprendidos en el intervalo [2.000, 50.000].

NÚMERO DE DATOS n	COSTE MEDIDO $t(n)$ EN HENDRIX-SSH EN SEGUNDOS

2. DEDUCCIÓN DE UNA FUNCIÓN DE COSTE APROXIMADA

A partir de los datos anteriores debe deducirse una función de coste aproximada que ha de tener la forma $t(n) = k \times n^2$. Presentar la deducción de dicha función.

3. DATOS DE COSTE ESTIMADOS

Anote en la siguiente tabla los costes estimados a partir de los resultados anteriores, en tiempo $t(n)$, de ejecutar la función ***contarRepetidosIter****(v,n,rep)* para los valores de n que se indican en la tabla.

NÚMERO DE DATOS n	COSTE ESTIMADO $t(n)$ EN HENDRIX-SSH EN HORAS, MINUTOS Y SEGUNDOS
100.000	
500.000	
1.000.000	

Una solución de las tareas propuestas en el turno 1º

```
/*
 * Pre: desde >= 0
 * Post: vecesEsta(d,v,desde,hasta) = VECES
 *         AND VECES = (Núm alfa EN [desde,hasta].v[alfa]=d))
 */
int vecesEsta (const int d, const int v[], int desde, int hasta) {
    if (desde<=hasta) {
        if (v[desde]==d) {
            return 1 + vecesEsta(d, v, desde+1, hasta);
        }
        else {
            return vecesEsta(d, v, desde+1, hasta);
        }
    }
    else {
        return 0;
    }
}

/*
 * Pre: n > 0 AND desde >= 0 AND desde <= n
 * Post: (PT alfa EN [desde,n-1].rep[alfa]=(Núm beta EN [0,n-1].v[alfa]=v[beta]))
 */
void contarRepetidos (const int v[], const int n, int desde, int rep[]) {
    if (desde<n) {
        rep[desde] = vecesEsta(v[desde], v, 0, n-1);
        contarRepetidos(v, n, desde+1, rep);
    }
}

/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].rep[alfa]=(Núm beta EN [0,n-1].v[alfa]=v[beta]))
 */
void contarRepetidos (const int v[], const int n, int rep[]) {
    contarRepetidos(v, n, 0, rep);
}
```

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].rep[alfa]=(Núm beta EN [0,n-1].v[alfa]=v[beta]))
 */
void contarRepetidosIter (const int v[], const int n, int rep[]) {
    for (int i=0; i<n;++i) {
        int cuenta = 0;
        for (int j=0; j<n; ++j) {
            if (v[i]==v[j]) {
                cuenta = cuenta + 1;
            }
        }
        rep[i] = cuenta;
    }
}
```

1. DATOS EXPERIMENTALES DE COSTE

Anote en la siguiente tabla los datos experimentales del coste, en tiempo $t(n)$, de ejecutar la función ***contarRepetidosIter(v,n,rep)*** para cuatro valores de n comprendidos en el intervalo [2.000, 50.000].

NÚMERO DE DATOS n	COSTE MEDIDO $t(n)$ EN HENDRIX-SSH EN SEGUNDOS.
5.000	1.31 seg
10.000	5.26 seg
20.000	21.19 seg
40.000	84.51 seg

2. DEDUCCIÓN DE UNA FUNCIÓN DE COSTE APROXIMADA

A partir de los datos anteriores debe deducirse una función de coste aproximada que ha de tener la forma $t(n) = k \times n^2$. Presentar la deducción de dicha función.

$t(n) = k \times n^2$

$k = \frac{t(n)}{n^2} = \frac{84.51}{40000^2}\ seg = 5.28 \times 10^{-8}\ seg$

Luego:

$t(n) = 5.28 \times 10^{-8} \times n^2\ seg$

3. DATOS DE COSTE ESTIMADOS

Anote en la siguiente tabla los costes estimados a partir de los resultados anteriores, en tiempo $t(n)$, de ejecutar la función ***contarRepetidosIter(v,n,rep)*** para cuatro los valores de n que se indican en la tabla.

NÚMERO DE DATOS n	COSTE ESTIMADO $t(n)$ EN HENDRIX-SSH EN HORAS, MINUTOS Y SEGUNDOS
100.000	528 seg = 8 min 48 seg
500.000	13.205 seg = 3 h 40 min 5 seg
1.000.000	52.820 seg = 14 h 40 min 20 seg

Escuela de Ingeniería y Arquitectura - Depto. de Informática e Ingeniería de Sistemas
Examen práctico de Programación 2 - 4 de junio de 2015 - Turno 2º - Tiempo: hasta las 18:50

Este examen práctico individual en laboratorio forma parte de la evaluación de la asignatura. Su calificación tiene un peso del 15 % en la calificación final de la asignatura en la convocatoria de junio.

PRIMERA PARTE [6.0 puntos]

Se ha de diseñar la función ***contarDuplicados****(v,n,rep)* que se especifica a continuación sin programar ni un solo bucle. Podrán añadirse las funciones auxiliares que se precisen. Cada una de estas deberá estar convenientemente especificada.

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].dup[alfa]=(Núm beta EN [alfa+1,n-1].v[alfa]=v[beta]))
 */
void contarDuplicados (const int v[], const int n, int dup[]);
```

SEGUNDA PARTE [4.0 puntos]

Se ha de diseñar la función ***contarDuplicadosIter****(v,n,rep)* que se especifica a continuación. Su coste en tiempo ha de ser $\boldsymbol{O}(t(n)) = \boldsymbol{O}(n^2)$. Esta función no debe presentar invocaciones a funciones auxiliares ni invocaciones recursivas. En cambio se pueden programar instrucciones iterativas. Se pide anotar los datos y deducciones, sobre el coste en tiempo de ejecutar esta función en el computador ***hendrix-ssh***, que se describen en el reverso de esta hoja.

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].dup[alfa]=(Núm beta EN [alfa+1,n-1].v[alfa]=v[beta]))
 */
void contarDuplicadosIter (const int v[], const int n, int dup[]);
```

PRESENTACIÓN DEL TRABAJO

Como resultado del trabajo se entregará:

- Un único fichero cuyas primeras líneas sean un comentario con el nombre del alumno y, a continuación, almacene exclusivamente el código de las funciones que integren el diseño completo de ***contarDuplicados****(v,n,rep)* así como el diseño completo de la función ***contarDuplicadosIter****(v,n,rep)*. La entrega se hará a través de la plataforma ***Moodle2*** (moodle2.unizar.es) antes de las 18:50. Se recomienda no hacerlo hasta que haya sido verificado de forma exhaustiva que el comportamiento de todo código diseñado es correcto.
- Rellenar la página posterior de esta hoja, que se entregará al profesor, en la que se deben anotar los datos de costes obtenidos en el computador ***hendrix-ssh*** y se deben responder las cuestiones planteadas en ella.

Examen práctico de Programación 2 - 4 de junio de 2015 - Turno 2º

Entregar esta hoja al profesor al finalizar el examen, debidamente cumplimentada.

Nombre y apellidos: _____

1. DATOS EXPERIMENTALES DE COSTE

Anote en la siguiente tabla los datos experimentales del coste, en tiempo $t(n)$, de ejecutar la función ***contarDuplicadosIter****(v,n,rep)* para cuatro valores de n comprendidos en el intervalo [2.000, 50.000].

NÚMERO DE DATOS n	COSTE MEDIDO $t(n)$ EN HENDRIX-SSH EN SEGUNDOS

2. DEDUCCIÓN DE UNA FUNCIÓN DE COSTE APROXIMADA

A partir de los datos anteriores debe deducirse una función de coste aproximada que ha de tener la forma $t(n) = k \times n^2$. Presentar la deducción de dicha función.

3. DATOS DE COSTE ESTIMADOS

Anote en la siguiente tabla los costes estimados a partir de los resultados anteriores, en tiempo $t(n)$, de ejecutar la función ***contarDuplicadosIter****(v,n,rep)* para los valores de n que se indican en la tabla.

NÚMERO DE DATOS n	COSTE ESTIMADO $t(n)$ EN HENDRIX-SSH EN HORAS, MINUTOS Y SEGUNDOS
100.000	
500.000	
1.000.000	

Una solución de las tareas propuestas en el turno 2º

```
/*
 * Pre: desde >= 0
 * Post: vecesEsta(d,v,desde,hasta) = VECES
 *         AND VECES = (Núm alfa EN [desde,hasta].v[alfa]=d))
 */
int vecesEsta (const int d, const int v[], int desde, int hasta) {
    if (desde<=hasta) {
        if (v[desde]==d) {
            return 1 + vecesEsta(d, v, desde+1, hasta);
        }
        else {
            return vecesEsta(d, v, desde+1, hasta);
        }
    }
    else {
        return 0;
    }
}

/*
 * Pre: n > 0 AND desde >= 0 AND desde <= n
 * Post: (PT alfa EN [desde,n-1].dup[alfa]=(Núm beta EN [alfa+1,n-1].v[alfa]=v[beta]))
 */
void contarDuplicados (const int v[], const int n, int desde, int
dup[]) {
    if (desde<n) {
        dup[desde] = vecesEsta(v[desde], v, desde+1, n-1);
        contarDuplicados(v, n, desde + 1, dup);
    }
}

/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].dup[alfa]=(Núm beta EN [alfa+1,n-1].v[alfa]=v[beta]))
 */
void contarDuplicados (const int v[], const int n, int dup[]) {
    contarDuplicados(v, n, 0, dup);
}
```

```
/*
 * Pre: n > 0
 * Post: (PT alfa EN [0,n-1].dup[alfa]=(Núm beta EN [alfa+1,n-1].v[alfa]=v[beta]))
 */
void contarDuplicadosIter (const int v[], const int n, int dup[]) {
    for (int i=0; i<n;++i) {
        int cuenta = 0;
        for (int j=i+1; j<n; ++j) {
            if (v[i]==v[j] ) {
                cuenta = cuenta + 1;
            }
        }
        dup[i] = cuenta;
    }
}
```

1. DATOS EXPERIMENTALES DE COSTE

Anote en la siguiente tabla los datos experimentales del coste, en tiempo $t(n)$, de ejecutar la función ***contarDuplicadosIter****(v,n,rep)* para cuatro valores de n comprendidos en el intervalo [2.000, 50.000].

NÚMERO DE DATOS n	COSTE MEDIDO $t(n)$ EN HENDRIX-SSH EN SEGUNDOS.
5.000	0.64 seg
10.000	2.61 seg
20.000	10.46 seg
40.000	41.95 seg

2. DEDUCCIÓN DE UNA FUNCIÓN DE COSTE APROXIMADA

A partir de los datos anteriores debe deducirse una función de coste aproximada que ha de tener la forma $t(n) = k \times n^2$. Presentar la deducción de dicha función.

$t(n) = k \times n^2$

$k = \frac{t(n)}{n^2} = \frac{41.95}{40000^2}\ seg = 2.62 \times 10^{-8}\ seg$

Luego:

$t(n) = 2.62 \times 10^{-8} \times n^2\ seg$

3. DATOS DE COSTE ESTIMADOS

Anote en la siguiente tabla los costes estimados a partir de los resultados anteriores, en tiempo $t(n)$, de ejecutar la función ***contarDuplicadosIter****(v,n,rep)* para cuatro los valores de n que se indican en la tabla.

NÚMERO DE DATOS n	COSTE ESTIMADO $t(n)$ EN HENDRIX-SSH EN HORAS, MINUTOS Y SEGUNDOS
100.000	261 seg = 4 min 22 seg
500.000	6.555 seg = 1 h 49 min 15 seg
1.000.000	26.219 seg = 7 h 16 min 19 seg