Cuestiones Cálculo de Complejidad

Fecha de Creación	@21 de septiembre de 2023 16:01
Asignatura	FAL
Fecha de Modificación	@21 de septiembre de 2023 20:35

Pregunta 1
Parcialmente correcta
Se puntúa 0,17 sobre 1,00
Marcar pregunta

Dado un número natural n, el siguiente algoritmo calcula el valor $\sum_{i=1}^{n}\sum_{j=1}^{i} j$:

```
int suma(int n) {
    int resul = 0;
    int i = 1;
    int j = 1;
    while (i <= n) {
        if (j > i) {
            i++;
            j = 1;
        }
        else {
            resul += j;
            j++;
        }
    }
    return resul;
}
```

El coste de este algoritmo está en:

Seleccione una o más de una:

- a. $O(1)$.
- b. $\Theta(1)$.
- c. $\Omega(1)$.
- d. $O(n^2)$. ✔ **Cierto**. A pesar de contener un único bucle, el coste del algoritmo es cuadrático, ya que, antes del k-esimo incremento de i (se excluye el último), j se incrementa k veces. Por tanto, como i se incrementa $n+1$ veces, el número total de incrementos es del orden de $\Theta(n^2)$. Como entre incremento e incremento de i y de j, el coste es constante, el coste total será también del orden de $\Theta(n^2)$.
- e. $\Theta(n)$. ✖ **Falso**. La función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(n)$. Por tanto, tampoco está en $\Theta(n)$.

a. **Falso**. La función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(1)$.
b. **Falso**. La función de tiempo no está en $O(1)$, por lo que tampoco está en $\Theta(1)$.
c. **Cierto**. La función de tiempo de este algoritmo es cuadrática, por lo que está en $\Omega(n^2) \subset \Omega(1)$.
d. **Cierto**. A pesar de contener un único bucle, el coste del algoritmo es cuadrático, ya que, antes del k-esimo incremento de i (se excluye el último), j se incrementa k veces. Por tanto, como i se incrementa $n+1$ veces, el número total de incrementos es del orden de $\Theta(n^2)$. Como entre incremento e incremento de i y de j, el coste es constante, el coste total será también del orden de $\Theta(n^2)$.
e. **Falso**. La función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(n)$. Por tanto, tampoco está en $\Theta(n)$.

Las respuestas correctas son: $\Omega(1)$.
, $O(n^2)$.

Pregunta **2**

Parcialmente correcta

Se puntúa 0,17 sobre 1,00

Marcar pregunta

El siguiente algoritmo permite restar un número natural b a cualquier número entero a utilizando únicamente decrementos:

```
int resta(int a, unsigned int b) {
  while (b != 0) {
     a--;
     b--;
  }
  return a;
}
```

El coste de este algoritmo está en:

Seleccione una o más de una:

- a. $\Theta(a)$.
- b. $O(b)$. ✔ **Cierto**. El algoritmo realiza b iteraciones, y el coste en cada iteración es $O(1)$. Por tanto, por la regla de repetición, el coste del bucles es $O(b)$, que es el coste dominante.
- c. $\Theta(1)$.
- d. $O(b^2)$.
- e. $\Theta(b^2)$. ✘ **Falso**. La función de tiempo de este algoritmo es lineal: no está en $\Omega(n^2)$, y, por tanto, tampoco está en $\Theta(n^2)$.

a. **Falso**. El algoritmo realiza b iteraciones, por lo que a no es adecuado como tamaño del problema.
b. **Cierto**. El algoritmo realiza b iteraciones, y el coste en cada iteración es $O(1)$. Por tanto, por la regla de repetición, el coste del bucles es $O(b)$, que es el coste dominante.
c. **Falso**. La función de tiempo de este algoritmo es lineal, y por tanto no está en $O(1)$, por lo que tampoco está en $Theta(1)$.
d. **Cierto**. El algoritmo realiza b iteraciones, y el coste en cada iteración es $O(1)$. Por tanto, por la regla de repetición, el coste del bucles es $O(b)$, y, por consiguiente, también es $O(b^2)$.
e. **Falso**. La función de tiempo de este algoritmo es lineal: no está en $\Omega(n^2)$, y, por tanto, tampoco está en $\Theta(n^2)$.

Las respuestas correctas son: $O(b)$.
, $O(b^2)$.

Pregunta **3**

Parcialmente correcta

Se puntúa 0,33 sobre 1,00

Marcar pregunta

Dado un array de enteros a y un número n ($n \geq 0$, y n menor que el número de elementos de a), el siguiente algoritmo determina si en los n primeros elementos de a hay un número par:

```
bool contiene_par(int a[], int n) {
   bool hay_par=false;
   int i=0;
   while(i < n && ! hay_par) {
      hay_par = a[i]%2==0;
      i++;
   }
   return hay_par;
}
```

Entonces, el coste de este algoritmo está:

Seleccione una o más de una:

- a. En el peor de los casos, en $\Theta(1)$.
- b. En el mejor de los casos, en $O(n)$.
- c. En el peor de los casos, en $O(n)$. ✔ **Cierto**. En el peor de los casos, el bucle itera n veces, con coste constante en cada iteración. Por tanto, el resultado se sigue de la regla de repetición.
- d. En el mejor de los casos, en $\Theta(n)$.
- e. En el peor de los casos, en $O(n^2)$.

a. **Falso**. En el peor de los casos, la función de tiempo no está en $O(1)$, por lo que tampoco está en $\Theta(1)$.
b. **Cierto**. En el mejor de los casos, el coste del algoritmo es constante, por lo que está en $O(1) \subset O(n)$.
c. **Cierto**. En el peor de los casos, el bucle itera n veces, con coste constante en cada iteración. Por tanto, el resultado se sigue de la regla de repetición.
d. **Falso**. En el mejor de los casos, el coste del algoritmo es constante, por lo que no está en $\Omega(n)$, y por tanto, tampoco está en $\Theta(n)$.
e. **Cierto**. En el peor de los casos, la función de tiempo de este algoritmo es lineal, por lo que está en $O(n) \subset O(n^2)$.

Las respuestas correctas son: En el mejor de los casos, en $O(n)$.
, En el peor de los casos, en $O(n)$.
, En el peor de los casos, en $O(n^2)$.

Pregunta **4**
Correcta
Se puntúa 1,00 sobre 1,00
Marcar pregunta

Dado un número natural n, el siguiente algoritmo calcula el valor $\sum_{i=1}^{n} \sum_{j=1}^{i} j$:

```
int suma(int n) {
  int resul = 0;
  for (int i = 1; i <= n; i++) {
     int uno_mas = 0;
     for (int j = 1; j <= i; j++) {
         uno_mas += j;
     }
     resul += uno_mas;
  }
  return resul;
}
```

El coste de este algoritmo está en:

Seleccione una o más de una:

- [x] a. $O(n^2)$. ✔ **Cierto**. El bucle externo realiza n iteraciones. En la iteración k-esima, el bucle interno realiza k iteraciones. Por tanto, el bucle interno realiza $\frac{n(n+1)}{2}$ iteraciones, es decir, del orden de $\Theta(n^2)$ iteraciones. Como, en cada iteración, el coste realizado es constante, el coste atribuible a todas las iteraciones del bucle interno es del orden de $\Theta(n^2)$, que es el factor de coste dominante del algoritmo.
- [ ] b. $O(1)$.
- [x] c. $\Omega(n)$. ✔ **Cierto**. La función de tiempo de este algoritmo es cuadrática, por lo que está en $\Omega(n^2) \subset \Omega(n)$.
- [ ] d. $\Theta(1)$.
- [ ] e. $O(n)$.

a. **Cierto**. El bucle externo realiza n iteraciones. En la iteración k-esima, el bucle interno realiza k iteraciones. Por tanto, el bucle interno realiza $\frac{n(n+1)}{2}$ iteraciones, es decir, del orden de $\Theta(n^2)$ iteraciones. Como, en cada iteración, el coste realizado es constante, el coste atribuible a todas las iteraciones del bucle interno es del orden de $\Theta(n^2)$, que es el factor de coste dominante del algoritmo.
b. **Falso**. La función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(1)$.
c. **Cierto**. La función de tiempo de este algoritmo es cuadrática, por lo que está en $\Omega(n^2) \subset \Omega(n)$.
d. **Falso**. La función de tiempo no está en $O(1)$, por lo que tampoco está en $\Theta(1)$.
e. **Falso**. La función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(n)$.

Las respuestas correctas son: $O(n^2)$.
, $\Omega(n)$.

Pregunta **5**
Parcialmente correcta
Se puntúa 0,67 sobre 1,00
Marcar pregunta

Dado un array de enteros a y un número n ($n \geq 0$, y n menor que el número de elementos de a), el siguiente algoritmo determina si el array contiene elementos repetidos:

```
bool contiene_repetidos(int a[], int n) {
    bool repetidos = false;
    int i = 0;
    while (i < n && !repetidos) {
        int j = i + 1;
        while (j < n && !repetidos) {
            repetidos = a[j] == a[i];
            j++;
        }
        i++;
    }
    return repetidos;
}
```

Entonces, el coste de este algoritmo está:

Seleccione una o más de una:

- [ ] a. En el peor de los casos, en $O(n)$.
- [x] b. En el mejor de los casos, en $\Theta(1)$. ✔ **Cierto**. El mejor de los casos aparece cuando, bien no hay elementos, bien hay solo uno, o bien hay dos elementos repetidos al comienzo del array. En todos estos supuestos el coste es constante.
- [ ] c. En el mejor de los casos, en $O(n)$.
- [ ] d. En el mejor de los casos, en $\Omega(n)$.
- [x] e. En el peor de los casos, en $\Omega(n)$. ✔ **Cierto**. En el peor de los casos, la función de tiempo de este algoritmo es cuadrática, por lo que está en $\Omega(n)$.

a. **Falso**. En el peor de los casos, la función de tiempo de este algoritmo es cuadrática, por lo que no está en $O(n)$.
b. **Cierto**. El mejor de los casos aparece cuando, bien no hay elementos, bien hay solo uno, o bien hay dos elementos repetidos al comienzo del array. En todos estos supuestos el coste es constante.
c. **Cierto**. En el mejor de los casos, el coste del algoritmo es constante, por lo que está en $O(1) \subset O(n)$.
d. **Falso**. En el mejor de los casos, el coste del algoritmo es constante, por lo que no está en $\Omega(n)$.
e. **Cierto**. En el peor de los casos, la función de tiempo de este algoritmo es cuadrática, por lo que está en $\Omega(n)$.

Las respuestas correctas son: En el mejor de los casos, en $\Theta(1)$.
, En el mejor de los casos, en $O(n)$.
, En el peor de los casos, en $\Omega(n)$.

Pregunta **6**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Dado un número natural n, el siguiente algoritmo calcula el valor $\sum_{i=1}^{n} \sum_{j=1}^{i} j$:

```
int suma(int n) {
    int resul = 0;
    int i = 1;
    int s = 0;
    while (i <= n) {
        int j = 0;
        while (i < n && j < s) {
            j++;
            resul += i*(i+1)/2;
            i++;
        }
        s++;
        if (i <= n) {
            resul += i*(i+1)/2;
            i++;
        }
    }
    return resul;
}
```

El coste de este algoritmo está en:

Seleccione una o más de una:

- a. $\Theta(n)$.
- b. $\Theta(n^2)$. ✗ **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$, y por tanto, tampoco está en $\Theta(n^2)$.
- c. $\Omega(n^2)$. ✗ **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$.
- d. $O(1)$.
- e. $\Theta(1)$.

a. **Cierto**. A pesar de contener dos bucles anidados, i se incrementa $n+1$ veces. El coste entre incremento e incremento es constante. Por tanto, el resultado se sigue de una consideración análoga a la de la regla de repetición.
b. **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$, y por tanto, tampoco está en $\Theta(n^2)$.
c. **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$.
d. **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $O(1)$.
e. **Falso**. La función de tiempo no está en $O(1)$, por lo que tampoco está en $\Theta(1)$.

La respuesta correcta es: $\Theta(n)$.

Pregunta **7**

Parcialmente correcta

Se puntúa 0,50 sobre 1,00

Marcar pregunta

Dado un número natural n, el siguiente algoritmo calcula el valor $\sum_{i=1}^{n} \sum_{j=1}^{i} j$:

```
int suma(int n) {
   int resul = 0;
   for (int i = 1; i <= n; i++) {
     resul += i * (i + 1) / 2;
   }
   return resul;
 }
```

El coste de este algoritmo está en:

Seleccione una o más de una:

- a. $\Omega(1)$.
- b. $\Theta(n^2)$.
- c. $\Omega(n^2)$.
- d. $\Omega(n)$. ✔ **Cierto**. Basta aplicar la regla de repetición.
- e. $\Theta(1)$.

a. **Cierto**. La función de tiempo de este algoritmo es lineal, por lo que está en $\Omega(n) \subset \Omega(1)$.
b. **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$, y, por tanto, tampoco en $\Theta(n^2)$.
c. **Falso**. La función de tiempo de este algoritmo es lineal, por lo que no está en $\Omega(n^2)$.
d. **Cierto**. Basta aplicar la regla de repetición.
e. **Falso**. La función de tiempo no está en $O(1)$, por lo que tampoco está en $\Theta(1)$.

Las respuestas correctas son: $\Omega(1)$.
, $\Omega(n)$.