**Laboratorio S03. AYED-02**

Jefer Alexis González Romero

**DOCUMENTO TÉCNICO PROBLEMA A**

**Requisitos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **2** | **4** | **6** | **1** | **3** |

**Especificación**

)

sort\_array(

|  |
| --- |
| **Agregar con relleno sólido1** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **2** | **4** | **6** | **3** |

)

sort\_array(

sort\_array(

)

|  |
| --- |
| **2** |

|  |  |  |  |
| --- | --- | --- | --- |
| **5** | **4** | **6** | **3** |

|  |
| --- |
| **3** |

|  |  |  |
| --- | --- | --- |
| **5** | **4** | **6** |

****

)

sort\_array(

|  |
| --- |
| **Agregar con relleno sólido4** |

|  |  |
| --- | --- |
| **5**  sort\_array( | **6** |

)

)

sort\_array(

|  |
| --- |
| **5** |

|  |
| --- |
| **6** |

|  |
| --- |
| **6** |

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** |

**Entrada:**

Lista de enteros, donde cada número está separado por un espacio.

**Salida:**

La lista ordenada de menor a mayor.

**Diseño**

**Estrategia**

Se toma el número entero más pequeño y se añade a una lista, se hace esto recursivamente con subarreglos tomando su menor elemento.

Se utiliza un arreglo de una dimensión del cual se toman subarreglos con slices para tomar un valor pedido y añadirlo a una lista.

Adjunto el programa con nombre “A”.

**Invariante**

Por cada subarreglo se lleva añade el entero más pequeño a la lista.

* Iniciación: Arreglo con longitud mayor a 1.
* Estabilidad: Por cada subarreglo debe haber un número menor.
* Terminación: Solo hay un elemento en el arreglo.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 1 | Solo un valor en el arreglo | [1] |
| 4 3 | Dos valores | [3, 4] |
| 1 2 3 4 5 6 7 | Arreglo ordenado | [1, 2, 3, 4, 5, 6, 7] |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(n)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA B**

**Requisitos**

**Especificación**

Subarreglo limitado por los índices (1, 3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |

Suma = 0

sum\_sub\_array(

, 1, 3, 0)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |

, 1 + 1, 3, 0 + 2)

sum\_sub\_array(

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |

, 2 + 1, 3, 2 + 3)

sum\_sub\_array(

sum\_sub\_array(

, 3 + 1, 3, 5 + 4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** |

Suma = 9

**Entrada:**

Lista de enteros, donde cada número está separado por un espacio.

**Salida:**

Suma de los valores del subarreglo limitado por los índices dados.

**Diseño**

**Estrategia**

Se utiliza un contador i que tendrá como valor inicial el primer índice, este aumentará hasta que sea igual al segundo índice. Se toman los valores del subarreglo con i y se van sumando recursivamente con el acumulador “suma”.

Se utiliza un arreglo de una dimensión, al cual se le toman los valores del subarreglo pedido.

Adjunto el programa con nombre “B”.

**Invariante**

Por cada valor en el subarreglo se suma a un acumulador.

* Iniciación: Primer valor en el subarreglo con posición menor o igual al segundo índice dado.
* Estabilidad: Por cada valor su posición en el arreglo debe estar dentro de los limites dados para el subarreglo.
* Terminación: La posición del valor es igual al segundo índice.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 1 2 3 4 5 6 7  4 4 | Índices con el mismo valor | 5 |
| 1 2 3 4 5 6 7  6 3 | Prime índice mayor al segundo | [3, 4] |
| 3  0 0 | Arreglo con un solo valor | [1, 2, 3, 4, 5, 6, 7] |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(n)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA C**

**Requisitos**

**Especificación**

**Entrada:**

Número entero N.

**Salida:**

Suma de los enteros positivos pares desde N hasta 2.

**Diseño**

**Estrategia**

Si el número dado es impar se le resta 1, de modo que se comience con un número par, se resta de a dos para tomar los números pares y sumarlos a un acumulador, esto se hace hasta que el número sea igual a 2. En la suma si N es par se tendrá en cuenta, al igual que el número 2. Con los números negativos se hace igual, ya que para cualquier número negativo la suma de los pares positivos hasta 2 es 2.

Adjunto el programa con nombre “C”.

**Invariante**

Por cada número par positivo desde N hasta 2 se suma a un acumulador.

* Iniciación: Numero par.
* Estabilidad: Cada número debe ser mayor a 2.
* Terminación: El número es 2 o menor a 2.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| -42 | Número negativo | 2 |
| 2 | El número es 2 | 2 |
| 1 | El número no negativo menor a 2 | 2 |
| 7 | Número impar mayor a 2 | 12 |
| 10 | Número par mayor a 2 | 30 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(n/2)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA D**

**Requisitos**

**Especificación**

**Entrada:**

Dos números enteros (m y n) separados por un espacio.

**Salida:**

El máximo común divisor de los números dados.

**Diseño**

**Estrategia**

Se usa el algoritmo de Euclides de forma recursiva para hallar el MCD de los números.

Adjunto el programa con nombre “D”.

**Invariante**

* Iniciación: Numero n dado, diferente de 0.
* Estabilidad: n toma como valor el residuo entre m y n, el cual debe ser diferente de 0.
* Terminación: n es igual a 0.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 360 450 | m < n | 90 |
| 30 30 | m = n | 30 |
| -200 -13 | m < 0, n < 0 | -1 |
| 6 0 | m = 0 ∨ n = 0 | 6 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

**Código**

**Documentación**

Dentro del código.

**DOCUMENTO TÉCNICO PROBLEMA E**

**Requisitos**

**Especificación**

**Entrada:**

Número entero positivo.

**Salida:**

Notación binaria del número dado.

**Diseño**

**Estrategia**

Se halla el logaritmo en base dos del número y se toma su parte entera, la cual representa la longitud del binario. Se resta lo obtenido al número y se vuelve a sacar el logaritmo en base dos a la diferencia obtenida, si el logaritmo obtenido antes difiere mayor a 1 con respecto al último, al binario se le agregan la diferencia menos uno de ceros. Así se hace recursivamente hasta que el numero sea igual a 0.

Si el número dado es par se le agregan los ceros necesarios hasta que se complete su longitud debida.

Adjunto el programa con nombre “E”.

**Invariante #1 (ver código)**

Por cada número entre el logaritmo anterior y el nuevo se agrega un cero al binario.

* Iniciación: La resta entre los logaritmos obtenidos.
* Estabilidad: La diferencia entre los logaritmos es mayor a 1.
* Terminación: La diferencia es igual a 1.

**Invariante #2 (ver código)**

Por cada logaritmo en base dos del número se le agregan los unos y ceros correspondientes al binario.

* Iniciación: El número dado y su logaritmo en base dos.
* Estabilidad: El número resultante es diferente de 0.
* Terminación: El número es igual a 0.

**Invariante #3 (ver código)**

Si el número es par, por cada número entre la parte entera del logaritmo en base dos de este y la longitud del binario obtenido recursivamente se le agrega un 0.

* Iniciación: El número binario.
* Estabilidad: La resta entre la parte entera del logaritmo en base dos del número dado y la longitud del binario son diferentes.
* Terminación: La longitud y el logaritmo son iguales.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 86 | Número par | 1010110 |
| 105 | Número impar | 1101001 |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(1)

En el peor de los casos T(n) = O(log2n)

**Código**

**Documentación**

Dentro del código

**DOCUMENTO TÉCNICO PROBLEMA F**

**Requisitos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **5** | **2** | **4** | **6** | **1** | **3** |

**Especificación**

invert(

)

|  |
| --- |
| **Agregar con relleno sólido3** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **2** | **4** | **6** | **1** |

)

invert(

invert(

)

|  |
| --- |
| **1** |

|  |
| --- |
| **6** |

|  |  |  |  |
| --- | --- | --- | --- |
| **5** | **2**  invert( | **4** | **6** |

|  |  |  |
| --- | --- | --- |
| **5** | **2** | **4** |

****

)

|  |
| --- |
| **Agregar con relleno sólido4** |

|  |  |
| --- | --- |
| **5** | **2** |

)

invert(

invert(

)

|  |
| --- |
| **2** |

|  |
| --- |
| **5** |

|  |
| --- |
| **5** |

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **3** | **1** | **6** | **4** | **2** | **5** |

**Entrada:**

Lista de enteros, donde cada número está separado por un espacio.

**Salida:**

La lista de enteros invertida.

**Diseño**

**Estrategia**

Se toma el último número y se añade a una lista, se hace esto recursivamente con subarreglos tomando suúltimo elemento.

Se utiliza un arreglo de una dimensión del cual se toman subarreglos con slices para tomar un valor pedido y añadirlo a una lista.

Adjunto el programa con nombre “F”.

**Invariante**

Por cada subarreglo se lleva el último elemento a la primera ubicación de esta.

* Iniciación: Contador con valor diferente de 0.
* Estabilidad: El contador es diferente a la longitud del arreglo menos dos.
* Terminación: La longitud del arreglo es igual al contador mas dos.

**Casos prueba**

|  |  |  |
| --- | --- | --- |
| Entrada | Justificación | Salida |
| 7 | Solo un valor en el arreglo | [7] |
| 2 65 | Dos valores | [65, 2] |
|  | Arreglo vacío | [] |

**Análisis**

**Temporal**

En el mejor de los casos T(n) = Ω(n)

En el peor de los casos T(n) = O(n)

**Código**

**Documentación**

Dentro del código.

**Ejercicio H**

**Peor caso**

Los arreglos dados tienen longitudes muy grandes  
len(one), len(two) ---> N  
Tomaré el caso en que todos los valores del primer arreglo son menores a los del segundo. Da igual que si se toman todos diferentes, ya que dentro del while se ejecuta (2n -1) veces y el slice del final se ejecuta solo una vez. Y en el caso que tomo dentro del while se ejecuta n veces y el slice tendrá O(n), así que da lo mismo.

Tabla

Descripción generada automáticamente

**Mejor caso**

Los arreglos dados están vacíos

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja