Comenzado el	jueves, 3 de diciembre de 2020, 08:47
Estado	Finalizado
Finalizado en	jueves, 3 de diciembre de 2020, 09:21
Tiempo empleado	34 minutos 29 segundos
Puntos	15/25
Calificación	**6** de 10 (**58**%)

Pregunta **1**

Finalizado

Puntúa 1 sobre 1

¿Cuántas comparaciones hace el algoritmo de Búsqueda Secuencial para encontrar un valor *x* en un arreglo de *n* componentes en el *peor caso posible*? (El peor caso es el que obliga a un algoritmo a hacer la máxima cantidad de trabajo. Obviamente, en el caso de la Búsqueda Secuencial ese peor caso se presenta si el valor buscado está exactamente en la última casilla, o bien, si el valr buscado no está en el arreglo).

Seleccione una:

a.
n comparaciones.

b.
Una sola comparación, siempre.

c.
$\log_2(n)$ comparaciones.

d.
n^2 comparaciones.

Pregunta **2**

Finalizado

Puntúa 0 sobre 1

El producto de dos números *a* y *b* mayores o iguales cero, es en última instancia *una suma*: se puede ver como sumar *b* veces el número *a*, o como sumar *a* veces el número *b*. Por ejemplo: 5 * 3 = 5 + 5 + 5 o bien 5 * 3 = 3 + 3 + 3 + 3 + 3. Sabiendo esto, se puede intentar hacer una definición recursiva de la operación producto(a, b) [a >= 0, b >= 0], que podría ser la que sigue:

$$\textbf{producto(a, b)} = \text{[a y b enteros, a >= 0 y b >= 0]} \begin{cases} 0 & \text{si a == 0 o b == 0} \\ a + producto(a, b-1) & \text{si a > 0 y b > 0} \end{cases}$$

¿Es correcto el siguiente planteo de la función *producto(a, b)*?

```
def producto(a, b):
    if a == 0 or b == 0:
        return 0
    return a + producto(a, b-1)
```

Seleccione una:

a.
No. La función sugerida siempre retornará 0, sean cuales fuesen los valores de *a* y *b*.

b.
No. La última línea debería decir *return a + producto(a-1, b-1)* en lugar de *return a + producto(a, b-1)*.

c.
Sí. Es correcto.

d.
No. La propia definición previa del concepto de producto es incorrecta: un producto es una multiplicación, y no una suma...

Pregunta **3**

Finalizado

Puntúa 0 sobre 1

Suponga que se tiene una pila *p* con capacidad para almacenar números enteros y que las operaciones básicas *pop()*, *peek()* y *push()* están correctamente implementadas en el módulo *stack.py* presentado en clases. Suponga que se han insertado los siguientes valores: [8 - 6 - 5 - 3 - 7] (el número 8 es el valor del frente o tope de la Pila). ¿Cuál de las siguientes secuencias de instrucciones permite retirar el valor 5 de la pila, pero dejando el 6 y 8 nuevamente arriba? (es decir: ¿cuál de las siguientes secuencias, dejaría la pila *p* en el estado [8 - 6 - 3 - 7]?)

Seleccione una:

a.
```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.peek(p)
stack.push(p, n2)
stcak.push(p, n1)
```

b.
```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
stack.push(p, n2)
stack.push(p, n1)
```

c.
```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
stack.push(p, n1)
stack.push(p, n2)
```

d.
```
n1 = stack.pop(p)
n2 = stack.pop(p)
x = stack.pop(p)
```

Pregunta **4**

Finalizado

Puntúa 0 sobre 1

¿Cuáles de los siguientes problemas pertenecen a la clase de complejidad NP (pero no a la clase de complejidad P (según lo que hasta hoy se sabe)? (Suponga que en todos los casos se está haciendo referencia a la versión *de decisión* del problema indicado) (Más de una respuesta puede ser válida. Marque todas las que considere correctas).

Seleccione una o más de una:

- a. El problema del Clique Máximo en un grafo no dirigido con n vértices.
- b. El problema de la Satisfactibilidad Booleana.
- c. El problema de la Inserción de un nuevo elemento en una pila de n componentes.
- d. El problema de la Multiplicación de Matrices (para simplificar, suponga matrices cuadradas y del mismo orden n).

Pregunta **5**

Finalizado

Puntúa 1 sobre 1

Suponga que se quiere desarrollar una función que tome como parámetro una secuencia (un string, una lista, una tupla, o cualquier otro tipo de colección que permita acceso mediante índices en Python) y que proceda a ordenar esa colección con diversos algoritmos conocidos, en forma ascendente o descendente a elección de quien invoca a la función.

En ese contexto, considere la siguiente definición para esa función, en la cual no es relevante su bloque de acciones a los efectos de la pregunta:

```
def ordenar(lista, ascendente=True, algoritmo='Quicksort'):
    # el bloque de acciones no importa ahora...
```

¿Cuáles de las siguientes invocaciones a esta función son correctas, y no provocarán un error de intérprete? (Observación: más de una respuesta puede ser válida. Marque todas las que considere adecuadas... y de nuevo: tómese su tiempo!!!)

Seleccione una o más de una:

- a. `ordenar(lista=(1,2,3), algoritmo='Heapsort', ascendente=False)`
- b. `ordenar('azbycx', algoritmo='Shellsort')`
- c. `ordenar('abcdef', ascendente=False, 'Insertionsort')`
- d. `ordenar('abcdef', False, metodo='Bubblesort')`

Pregunta **6**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características **correctas** del algoritmo *Shellsort*? (Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

- a.
  En el caso promedio, el algoritmo *Shellsort* es tan eficiente como el *Heapsort* o el *Quicksort*, con tiempo de ejecución *O(n*log(n))*.

- b.
  El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Inserción Directa* (o *Inserción Simple*), consistente en armar suconjuntos ordenados con elementos a distancia $h > 1$ en las primeras fases, y terminar con $h = 1$ en la última.

- c.
  El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Selección Directa*, consistente en buscar iterativamente el menor (o el mayor) entre los elementos que quedan en el vector, para llevarlo a su posición correcta, pero de forma que la búsqueda del menor en cada vuelta se haga en tiempo logarítmico.

- d.
  El algoritmo *Shellsort* es complejo de analizar para determinar su rendimientos en forma matemática. Se sabe que para la serie de incrementos decrecientes usada en la implementación vista en las clases de la asignatura, tiene un tiempo de ejecución para el peor caso de $O(n^{1.5})$.

Pregunta **7**

Finalizado

Puntúa 0 sobre 1

Suponga que en un programa en Python necesita abrir un archivo de tal forma que:

- El archivo se maneje como archivo binario (y no como archivo de texto).
- Si el archivo no existía NO sea creado.
- Se permita tanto leer como grabar su contenido.
- No se destruya el contenido si el archivo ya existía.
- Se pueda leer en cualquier posición del archivo y también grabar en cualquier posición del mismo.

¿Cuál de los posibles modos de apertura disponibles para *open()* tendría que emplear al abrir ese archivo?

Seleccione una:

- a. El modo rb.
- b. El modo a+b.
- c. El modo r+b.
- d. El modo r+t.

Pregunta **8**

Finalizado

Puntúa 0 sobre 1

Suponga *función test()* (que se muestra más abajo) toma como parámetro una matriz *mat* ya definida (y otras dos variables *x* e *y*) y correctamente cargada, con elementos de tipo *int*. ¿Qué hace concretamente esta función?

```
def test(mat, x, y):
    n = len(mat)
    m = len(mat[0])
    if x < 0 or x >= n:
        return False
    if y < 0 or y >= n:
        return False
    if x == y:
        return True

    for k in range(m):
        mat[x][k], mat[y][k] = mat[y][k], mat[x][k]

    return True
```

Seleccione una:

a.
Intercambia los valores contenidos en la fila *x* con los valores contenidos en la fila *y*. Retorna *True* si el intercambio pudo hacerse (o si *x* era igual a *y*); y retorna *False* en caso contrario.

b.
Comprueba si la matriz es cuadrada. En caso de serlo, intercambia los valores contenidos en la fila *x* con los valores contenidos en la columna *y*. Retorna *True* si el intercambio pudo hacerse y retorna *False* en caso contrario.

c.
Intercambia los valores contenidos en la columna *x* con los valores contenidos en la columna *y*. Retorna *True* si el intercambio pudo hacerse (o si *x* era igual a *y*); y retorna *False* en caso contrario.

d.
Comprueba si alguno de los valores contenidos en la fila *x* es igual a alguno de los valores contenidos en la fila *y*. Retorna *True* en ese caso (o si *x* era igual a *y*); y retorna *False* en caso contrario.

Pregunta **9**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes estrategias de obtención del pivot *es la más recomendable* para evitar que el algoritmo *Quicksort* se degrade en su peor caso $O(n^2)$ en cuanto a su tiempo de ejecución?

Seleccione una:

a.
En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición).

b.
En cada partición, usar como pivot al valor de la casilla central.

c.
En cada partición, usar como pivot al valor de la última casilla.

d.
En cada partición, usar como pivot al valor de la primera casilla.

Pregunta **10**

Finalizado

Puntúa 0 sobre 1

*¿Cuáles de las siguientes afirmaciones **serían ciertas** si el problema **P** vs NP se resolviese por la negativa (es decir, si se demostrase que P y NP **no son** realmente iguales*? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

- [x] a.
  Todo problema actualmente en NP, tendría solución de tiempo polinomial en una máquina determinista.
- [ ] b.
  Todo problema actualmente en P se convertiría en intratable.
- [x] c.
  Todo problema actualmente considerado intratable tendría solución de tiempo de ejecución polinomial en una máquina determinista.
- [ ] d.
  Se tendría la certeza de que al menos para algunos problemas no existen buenas soluciones, lo que permitiría dejar de buscarlas inútilmente y pasar a concentrar los esfuezos en diseñar soluciones aproximadas o no óptimas.

Pregunta **11**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de listado completo de los registros de un archivo*, suponiendo que los registros *contienen un campo de marcado lógico* para gestionar eventuales bajas lógica?

Seleccione una:

- a.
  1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como *no válido* (*activo = False*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).
- b.
  1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).
- c.
  1) Abrir el archivo *m* en modo 'ab'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario ignorarlo (no mostrar su contenido).
- d.
  1) Abrir el archivo *m* en modo 'rb'. 2) Usar un ciclo para leer uno por uno los registros del archivo. 3) En cada vuelta del ciclo, si el registro leído está marcado como válido (*activo = True*), mostrar su contenido; en caso contrario no mostrar su contenido y detener el ciclo.

Pregunta **12**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones son ciertas, en relación al concepto de registro en Pyhton? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- [x] a.
  Un registro es un conjunto mutable de valores que pueden ser de distintos tipos. Cada componente de un registro se denomina campo. Pueden agregarse nuevos campos y cambiar los mismos de valor durante la ejecución de un programa.

- [ ] b.
  Un registro es un conjunto inmutable de valores. Cada componente de un registro se denomina campo. Pueden agregarse campos, pero la estructura resulta inmutable porque luego de ejecutarse el constructor ya no pueden agregarse campos nuevos.

- [ ] c.
  Un registro es un conjunto mutable de valores que deben ser todos del mismo tipo. Cada componente de un registro se denomina campo. Pueden agregarse campos y cambiar los mismos de valor durante la ejecución de un programa siempre y cuando se respeten los mismos tipos de datos.

- [ ] d.
  Un registro es un conjunto inmutable de valores que pueden ser de distintos tipos. Cada componente de un registro se denomina campo, estos no pueden cambiarse luego de que se les ha asignado un valor.

Pregunta **13**

Finalizado

Puntúa 1 sobre 1

¿Qué se entiende, en el contexto del Análisis de Algoritmos, por un *Orden de Complejidad*?

Seleccione una:

- (•) a.
  Un conjunto o familia de funciones matemáticas que se comportan asintóticamente de la misma forma.

- ( ) b.
  Un conjunto o familia de algoritmos que resuelven el mismo problema.

- ( ) c.
  Un conjunto o familia de subrutinas con similares objetivos (equivalente al concepto de *módulo*).

- ( ) d.
  Un conjunto de datos ordenados.

Pregunta **14**

Finalizado

Puntúa 1 sobre 1

El siguiente programa es una variante del que mostramos en la ***Ficha 7*** para resover el problema 20. En esta variante, hemos modificado la lógica para que al buscar el menor importe del vendedor 2 no se tenga que preguntar por el primer dato en cada vuelta, y eliminar la bandera que se aplicaba en la versión original. ¿Hay algún problema en el planteo de esta variante? *Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto*.

```
__author__ = 'Catedra de AED'


# pasó la primera venta del vendedor 2?
aviso = False

# si no se cargan ventas del vendedor 2, menor_importe
queda en None...
menor_importe = None

# acumuladores de cantidades...
c1 = c2 = 0

# acumuladores de importes...
i1 = i2 = 0

print('Ventas de un Comercio... ingrese los datos de cada
venta...')

# ingresar datos de la primera venta...
codigo = -1
while codigo < 0 or codigo > 2:
    codigo = int(input('Codigo de vendedor (1 o 2) (0 para
cortar): '))
    if codigo > 2 or codigo < 0:
        print('Error... se pidio 1 o 2 o 0 para
cortar...')

cantidad = int(input('Cantidad vendida: '))
importe = float(input('Importe: '))

menor_importe = importe

while codigo != 0:
    if codigo == 1:
        c1 += cantidad
        i1 += importe

    elif codigo == 2:
        c2 += cantidad
        i2 += importe

        # Aplicar mecanismo de cálculo del menor...
        if importe < menor_importe:
            menor_importe = importe

    # ingresar el siguiente codigo y volver al ciclo...
    codigo = -1
    while codigo < 0 or codigo > 2:
        codigo = int(input('Codigo de vendedor (1 o 2) (0
para cortar): '))
        if codigo > 2 or codigo < 0:
            print('Error... se pidio 1 o 2 o 0 para
cortar...')

    cantidad = int(input('Cantidad vendida: '))
    importe = float(input('Importe: '))

# Calcular el importe promedio...
promedio = (i1 + i2) / 2

print('Cantidad de productos vendida por el vendedor 1:',
c1)
print('Cantidad de productos vendida por el vendedor 2:',
c2)
print('Importe total facturado por el vendedor 1:', i1)
print('Importe total facturado por el vendedor 2:', i2)
print('Importe de la menor venta del vendedor 2:',
menor_importe)
print('Importe promedio entre los dos vendedores:',
promedio)
```

Seleccione una:

- a.
  Sí. Hay un único problema: los tres datos de la venta se cargan juntos. Si código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido.

b.
Sí. De hecho, hay dos problemas: el primero es que los tres datos de la venta se cargan juntos. Si el código del vendedor en ese momento fuese 0, el ciclo debería cortar pero de todos modos el programa pide al usuario los otros dos datos... que no tendrán ningún sentido. Y el segundo problema (y más grave) es que se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.

c.
Sí. Hay un único problema: se pedía el menor importe del vendedor 2, pero antes del ciclo la variable *menor_importe* se inicializa con el primer importe cargado, lo que sería un error si el código del vendedor en esa carga fuese el 1 y no el 2.

d.
No. No hay ningún problema. Funcionará correctamente en todos los casos.

Pregunta **15**
Finalizado
Puntúa 1 sobre 1

Considere el problema del Cambio de Monedas analizado en clases, y la solución mediante un Algoritmo Ávido también presentada en clases ¿Cuáles de las siguientes afirmaciones ***son ciertas*** en relación al problema y al algoritmo citado? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

a.
Si el Problema de Cambio de Monedas no puede resolverse en forma óptima para un conjunto dado de monedas que incluya a la de 1 centavo, mediante el Algoritmo Ávido propuesto, entonces el problema no tiene solución.

b.
El Algoritmo Ávido sugerido para el problema del Cambio de Monedas funciona correctamente para cualquier conjunto de valores nominales de monedas, siempre y cuando ese conjunto incluya a la moneda de 1 centavo.

c.
El Algoritmo Ávido sugerido para el Problema del Cambio de Monedas falla si el valor x a cambiar tiene una moneda igual a x en el conjunto de valores nominales: en ese caso, el algoritmo provoca un error de runtime y se interrumpe.

d.
Sea cual sea el algoritmo que se emplee, es exigible que exista la moneda de 1 centavo, pues de otro modo no habrá solución posible para muchos valores de cambio.

Pregunta **16**

Finalizado

Puntúa 1 sobre 1

Suponga que para un problema dado se ha planteado un algoritmo basado en divide y vencerás cuya estructura lógica esencial es la siguiente:

```
proceso(partición):
   adicional() [O(n)]
   proceso(partición/3)
   proceso(partición/3)
   proceso(partición/3)
```

¿Cuál de las siguientes es la expresión en *notación Big O* del tiempo de ejecución para este proceso? (Aclaración: la base del logaritmo no es esencial en una expresión en notación Big O, pero en este caso es relevante a los efectos del planteo conceptual de la pregunta).

Seleccione una:

- a. $O(n^2 * \log_3(n))$
- b. $O(n^2 * \log_2(n))$
- c. $O(n * \log_2(n))$
- d. $O(n * \log_3(n))$

Pregunta **17**

Finalizado

Puntúa 0 sobre 1

¿Cuál es la ***principal*** característica de todos los métodos de ordenamiento conocidos como métodos simples o directos?

Seleccione una:

- a. Son muy fáciles de programar.
- b. Son muy veloces para cualquier tamaño del arreglo a ordenar.
- c. Tienen un tiempo de ejecución de orden cuadrático.
- d. Tienen un tiempo de ejecución de orden lineal.

Pregunta **18**

Finalizado

Puntúa 1 sobre 1

¿Cuál es el motivo por el cual la *dependencia externa* es un problema en cuanto a las posibilidades de reuso de una función?

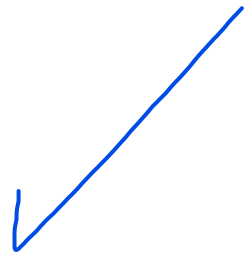
Seleccione una:

a.
Una función puede tener dependencia externa, pero esa dependencia externa no es un problema en cuanto a la reutilización de una función.

b.
El concepto de dependencia externa no es aplicable a funciones, sino a ciclos. Por lo tanto, la pregunta en sí misma carece de sentido.

c.
Si la función utiliza elementos definidos fuera de ella, entonces los programas donde se quiera usar esa función deberán haceer declaraciones especiales para poder usarla, y/o se deberá modificar la propia función

d.
Es peor que eso: si la función utiliza elementos definidos fuera de ella, no hay forma alguna de poder usarla en ningún programa: no sólo no es reutilizable, sino que simplemente no puede usarse.



Pregunta **19**

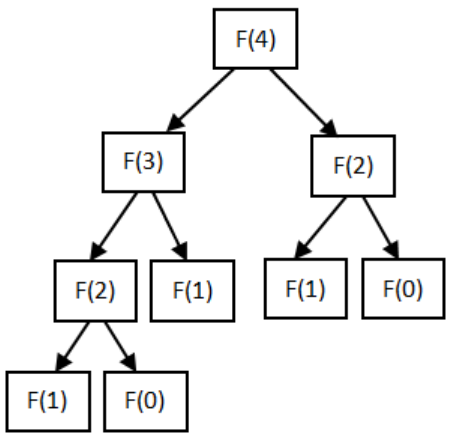
Finalizado

Puntúa 1 sobre 1

Sabemos que la recursión es una de las técnicas o estrategias básicas para el planteo de algoritmos y que una de sus ventajas es que permite el diseño de algoritmos compactos y muy claros, aunque al costo de usar algo de memoria extra en el stack segment y por consiguiente también un algo de tiempo extra por la gestión del stack. Algunos problemas pueden plantearse en forma directa procesando recursivamente diversos subproblemas menores. Tal es el caso de la *sucesión de Fibonacci*, en la cual cada término se calcula como la suma de los dos inmediatamente anteriores [esto se expresa con la siguiente relación de recurrencia: *F(n) = F(n-1) + F(n-2)* con *F(0) = 0* y *F(1) = 1*]. La función sencilla que mostramos a continuación que calcula el término n-ésimo de la sucesión en forma recursiva:

```
def fibo(n):
    if n <= 1:
        return 1
    return fibo(n-1) + fibo(n-2)
```

Sin embargo, un inconveniente adicional es que la *aplicación directa* de *dos o más invocaciones recursivas* en el planteo de un algoritmo podría hacer que un mismo subproblema se resuelva más de una vez, incrementando el tiempo total de ejecución. Por ejemplo, para calcular fibo(4) el árbol de llamadas recursivas es el siguiente:



y puede verse que en este caso, se calcula *2 veces* el valor de *fibo(2)*, *3 veces* el valor de *fibo(1)* y *2 veces* el valor de *fibo(0)*... con lo que la cantidad de llamadas a funciones para hacer *más de una vez el mismo trabajo* es de ***7*** (= 2 + 3 + 2).

Suponga que se quiere calcular el valor del sexto término de la sucesión. Analice el árbol de llamadas recursivas que se genera al hacer la invocación ***t = fibo(6)*** **¿Cuántas veces en total la función *fibo()* se invoca para hacer más de una vez el mismo trabajo, SIN incluir en ese conteo a las invocaciones para obtener *fibo(0)* y *fibo(1)*?**

Seleccione una:

- a. 12
- b. 10
- c. 0
- d. 11

Pregunta **20**

Finalizado

Puntúa 0 sobre 1

Suponga que en un archivo cuyo nombre está contenido en la variable **fd** se tiene grabada una secuencia de números, y que se quiere mostrar esos números por pantalla. Analice la siguiente función e indique cual es el resultado que se obtendría al ejecutarla (asuma que en el programa al que pertenece esta función están correctamente importados los módulos pickle y os.path).

```
def leer_todos(fd):
    if not os.path.exists(fd):
        print('El archivo', fd, 'no existe...')
        print()
        return

    m = open(fd, 'wb')
    t = os.path.getsize(fd)
    while m.tell() < t:
        num = pickle.load(m)
        print(num)
    m.close()
```

Seleccione una:

a.
Si el archivo está vacío, la función graba una secuencia de números en él, y luego lo recorre para mostrar los números que acaba de grabar.

b.
Si el archivo no está vacío, la función muestra correctamente todos los números que contiene.

c.
Si el archivo no está vacío, la función así como está planteada preserva los datos del archivo pero finaliza sin mostrar nunca ninguno de esos datos.

d.
Incluso si el archivo no estuviese vacío, esta función así como está planteada elimina todos los datos que el archivo tenía y finaliza sin mostrar nunca ninguno de esos números.

Pregunta **21**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes resume en forma correcta la idea general de la estrategia *Divide y Vencerás* para resolución de problemas?

Seleccione una:

a.
Se usa una tabla para almacenar los resultados de los subproblemas que se hayan calculado, y luego cuando algún subproblema vuelve a aparecer se toma su valor desde la tabla, para evitar pérdida de tiempo.

b.
El conjunto de *n* datos se divide en subconjuntos de aproximadamente el mismo tamaño (n/2, n/3, n/4, etc.). Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

c.
Se aplica una regla simple que parece ser beneficiosa, sin volver atrás ni medir las consecuencias de aplicar esa regla, con la esperanza de lograr el resultado óptimo al final.

d.
El conjunto de *n* datos se divide en subconjuntos de cualquier tamaño, sin importar si los tamaños de cada subconjunto coinciden entre sí. Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

Pregunta **22**

Finalizado

Puntúa 0 sobre 1

Considere el programa para el *Juego del Número Secreto* que se presentó en la Ficha 7. Ese programa se planteó originalmente sin funciones, y aquí lo mostramos redefinido para emplear funciones. Pero además, en la versión original del programa se usaba una *bandera* para marcar en el algoritmo si el número secreto fue encontrado o no; y nos proponemos ahora tratar de eliminar el uso de esa bandera y simplificar la estructura del programa. Sugerimos la modificación que se muestra más abajo empleando una *instrucción* ***return*** *para cortar el ciclo y la función apenas se encuentre el número secreto*. ¿Funciona correctamente el programa que estamos sugiriendo? *Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto.*

```
__author__ = 'Catedra de AED'

import random


def play_secret_number_game(limite_derecho,
cantidad_intentos):

    # limites iniciales del intervalo de búsqueda...
    izq, der = 1, limite_derecho

    # contador de intentos...
    intentos = 0

    # el numero secreto...
    secreto = random.randint(1, limite_derecho)

    # el ciclo principal... siga mientras no
    # haya sido encontrado el número, y la
    # cantidad de intentos no llegue a 5...
    while intentos < cantidad_intentos:
        intentos += 1
        print('\nEl numero está entre', izq, 'y', der)

        # un valor para forzar al ciclo a ser [1, N]...
        num = izq - 1

        # carga y validación del número sugerido por el
usuario...
        while num < izq or num > der:
            num = int(input('[Intento: ' + str(intentos) +
'] => Ingrese su numero: '))
            if num < izq or num > der:
                print('Error... le dije entre', izq, 'y',
der, '...')

        # controlar si num es correcto, avisar y cortar el
ciclo...
        if num == secreto:
            print('\nGenio!!! Acertaste en', intentos,
'intentos')
            return

        # ... pero si no lo es, ajustar los límites
        # del intervalo de búsqueda... y seguir...
        elif num > secreto:
            der = num
        else:
            izq = num

    print('\nLo siento!!! Se te acabaron los intentos. El
número era:', secreto)


def test():
    print('Juego del Número Secreto... Configuración
Inicial...')
    ld = int(input('El número secreto estará entre 1 y:
'))
    ci = int(input('Cantidad máxima de intentos que tendrá
disponible: '))
    play_secret_number_game(ld, ci)


# script principal...
test()
```

Seleccione una:

a.
No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará el mensaje avisándole que ganó pero el ciclo continuará pidiendo que se ingrese un número, hasta agotar el número de intentos disponible.

b.

No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto la función *play_secret_number_game()* mostrará correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *return*. Pero luego la función continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.

c.
No. No funciona bien: en la forma en que está planteado, se muestran en forma incorrecta los mensajes informando los límites del intervalo que contiene al número secreto en cada vuelta del ciclo, ya que las variables *izq* y *der* se actualizan en forma incorrecta.

d.
Sí. Funciona correctamente para todos los casos.

Pregunta **23**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de los siguientes son conocidos algoritmos basados en la estrategia *Divide y Vencerás*? (Más de una respuesta puede ser correcta, por lo que marque todas las que considere válidas)

Seleccione una o más de una:

a.
Algoritmo *Quicksort* para ordenamiento de arreglos.

b.
Algoritmo *Mergesort* para ordenamiento de arreglos.

c.
Algoritmo *Shellsort* para ordenamiento de arreglos.

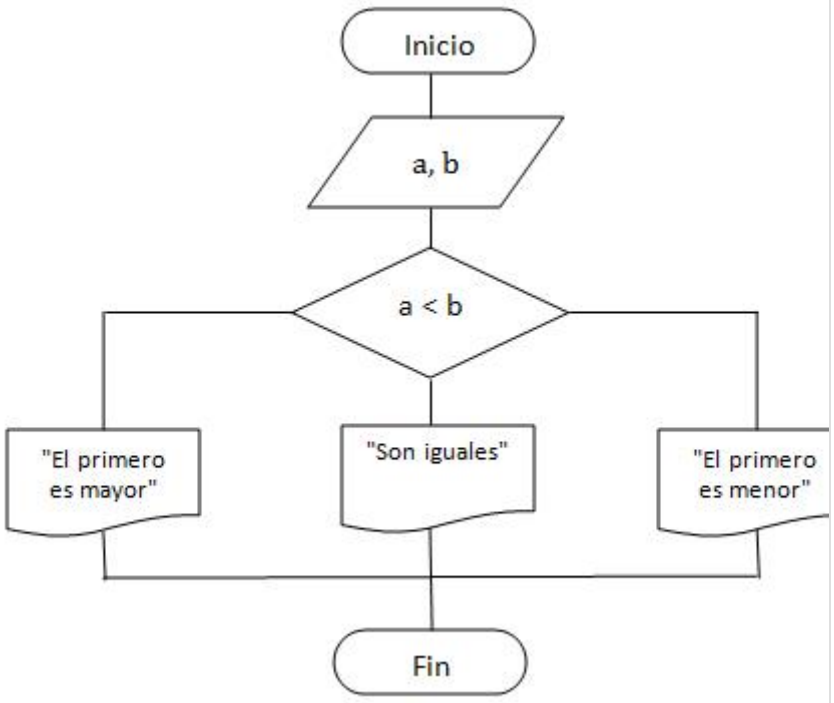
d.
Algoritmo *Heapsort* para ordenamiento de arreglos.

Pregunta **24**

Finalizado

Puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números y muestre un mensaje indicando si el primero es menor, igual o mayor al segundo. ¿Está bien planteado el siguiente diagrama de flujo?



Seleccione una:

- a. Está mal planteado: la condición está mal dibujada, ya que una condición no puede tener tres salidas o ramas.
- b. Sí. El diagrama está correctamente planteado.
- c. Está mal planteado: en la condición, se debía preguntar si a >= b.
- d. Está mal planteado: el símbolo usado para la carga de datos, debió ser un rectángulo y no un paralelogramo.

Pregunta **25**

Finalizado

Puntúa 1 sobre 1

El proceso de búsqueda del mayor en el siguiente programa, contiene una ligera modificación a las variantes analizadas en la Ficha 7: la variable *may* se inicializa en 0 antes del ciclo, y luego simplemente se aplica la idea de comparar el número cargado *num* contra *may* en cada vuelta, sin preocuparse por el valor de *may* frente al primer dato. ¿Hay algún problema en el planteo de esta estrategia?

```
__author__ = 'Catedra de AED'

print('Determinacion del mayor de una sucesion (variante:
may inicializada en 0)...')
n = int(input('Ingrese n: '))

may = 0
for i in range(1, n+1):
    num = int(input('Numero: '))
    if num > may:
        may = num

print('El mayor es:', may)
```

Seleccione una:

a.
No. No hay ningún problema. Funcionará correctamente en todos los casos.

b.
Sí, hay un problema: si todos los números del conjunto de entrada fuesen cero, se mostraría un *None* como resultado, en lugar de lo que correspondería retornar que es un 0.

c.
Sí, hay un problema: si todos los números del conjunto de entrada fuesen positivos o cero, se mostraría un 0 como resultado, en lugar del mayor del conjunto cargado.

d.
Sí, hay un problema: si todos los números del conjunto de entrada fuesen negativos, se mostraría un 0 como resultado, en lugar del mayor de los negativos cargados.

◄ Avisos

Ir a...

Examen Final - Registro de Calificación Final (Libres, Regulares y Promocionados - Cualquier año) ►