Problema 1: Sucesión

Enunciado

Dada la siguiente secuencia de números enteros grandes P(n) definida por los siguientes valores iniciales

$$P(0) = 4;\ P(1) = 3;\ P(2) = 2;\ P(3) = 1$$

y la siguiente relación de recurrencia

$$P(n) = 2P(n-2) + P(n-4)$$

Los primeros valores de P(n) son: 4, 3, 2, 1, 8, 5, 18, 11, 44, 27, .. .

Preguntas

1. Decida cuál es la mejor opción para este problema concreto: ¿con o sin memoria? Justifíquelo brevemente. ¿Qué tipo de recursión puede realizarse? Razone la respuesta. ¿Es una recursividad simple o múltiple? ¿final o no final?

2. Realice la definición recursiva de la sucesión indicada.

3. Implemente (sin usar memoria) el algoritmo recursivo no final empleando el lenguaje de programación que le ha indicado su profesor que solucione el problema planteado.

4. Implemente si es posible una función recursiva final empleando el lenguaje de programación que le ha indicado su profesor que solucione el problema planteado.

5. Implemente el algoritmo iterativo empleando el lenguaje de programación que le ha indicado su profesor que solucione el problema planteado.

6. Defina el tamaño del problema, calcule el T(n) y la complejidad del algoritmo (con y sin memoria) considerando los casos mejor, peor y medio.

Problema 2: Comprobado de integridad

Enunciado

Un algoritmo, *comprobarIntegemplridad,* permita determinar la integridad de un texto en una comunicación. Para realizar dicha comprobación, las cadenas de texto que se reciben han sido marcadas de forma especial antes de ser enviadas. Veamos un ejemplo del texto:

```
"Esto es un text securizado$2516"
```

Como puede comprobarse la cadena tiene una marca $2516, la marca $ indica el final de la cadena y el 2516 es la suma del valor numérico ascii de cada carácter de la cadena de texto.

El algoritmo hace uso de una función *valorNumerico* que determine el valor numérico de la cadena; hasta el carácter $ (sin incluir este). Posteriormente comparará el valor obtenido por esta función con el valor tras el carácter $. Si ambos valores coinciden quiere decir que el texto está íntegro y no ha sido modificado en la comunicación y el algoritmo devolverá *true*, en cualquier otro caso el texto ha sido comprometido devolverá *false*.

Preguntas

1. Defina la función recursiva *valorNumerico* que dado una cadena de texto marcada devuelva su valor numérico entero.
2. Implemente una función **recursiva no final** empleando el lenguaje de programación que le ha indicado su profesor que solucione el problema planteado, partiendo de la función definida previamente.
3. Implemente el algoritmo *comprobarIntegridad;* empleando el lenguaje de programación que le ha indicado su profesor, que dado una cadena de caracteres de entrada y haciendo uso de la implementación de la función recursiva no final del apartado anterior, determine si la cadena es integra o no.
4. Implemente una función **recursiva final** a partir de la definición del apartado 2.
5. A partir de las funciones anteriores, diseñar e implementar un **algoritmo iterativo** que proporcione el valor deseado.
6. Indique el tamaño del problema, T(n) y la complejidad del algoritmo implementado en el apartado 2.

Nota:

- Asuma que dentro del texto no vendrán incluidos símbolos
- Asuma que dispone de la siguiente función que permiten calcular el valor de un carácter:

```
int valor(char c){
      return (int)c;
}
```

- Ejemplo de transformación cadena caracteres a entero en C:

```
int val;
val = atoi("12345");
```

Problema 3: El número Aureo (*φ*)

Enunciado

$$\varphi = 1 + \frac{1}{\varphi} \rightarrow \varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \ldots}}}}$$

φ es un número irracional y es considerado, entre otras cosas, como la proporción perfecta. No puede ser expresado con un número de decimales finito. Su valor aproximado es de 1,61803.

Para poder obtener aproximaciones de dicho número existen diferentes reglas que pueden ser aplicadas. Siguiendo un desarrollo decimal como el mostrado en la figura, puede obtenerse el número ***φ***.

Para poder realizar un algoritmo que realice dicho cálculo aproximado se requiere crear una función con el siguiente prototipo:

float f_aureo(int n)

Esta función devolverá la aproximación del número ***φ*** aplicando *n* desarrollos. Como puede suponer, mientras más grande sea *n* (más desarrollos serán aplicados), mejor será la aproximación. Considere los siguientes ejemplos:

f_aureo(0)=1

f_aureo(1)=1+ 1/1=2

f_aureo(2)=1+ 1/(1+ 1/1)= 1.5

f_aureo(5)=1+ 1/(1+ 1/(1+ 1/(1+ 1/(1+1/1)))))= 1.625

Preguntas

1. Defina la función recursiva no final para "el cálculo del número Aureo para *n* desarrollos".
2. Implemente una función **recursiva no final** empleando el lenguaje de programación que le ha indicado su profesor que solucione el problema planteado, partiendo de la función definida previamente.
3. Implemente una función **recursiva final** a partir de la definición anterior.

4. A partir de las funciones anteriores, diseñar e implementar un **algoritmo iterativo** que proporcione el valor deseado.
5. Indique el tamaño del problema, T(n) y la complejidad de los tres algoritmos.

Problema 4: Pico pala

Enunciado

Dada la siguiente definición recursiva:

$$pico(j, V) = \begin{cases} V[0] & si\ j <= 0 \\ 2 * V[j] & si\quad j <= 1 \\ pico(j-2, V) * pala(j, V) + V[j-1] & en\ otro\ caso \end{cases}$$

$$pala(j, V) = \begin{cases} V[0] & si\ j <= 0 \\ 2 * V[j] & si\quad j <= 1 \\ pico(j-1, V) & en\ otro\ caso \end{cases}$$

Donde `V` será un vector de enteros de tamaño `TAM`.

Preguntas

1. Implemente la función `pico(j,V)`.
2. Defina una función `picopala(j,V)` equivalente a la función `pico(j,V)` donde sólo se puedan realizar llamadas recursivas a la función `picopala(j,V)`. ¿La definición de la función `picopala(j,V)` se trata de una recursividad lineal final? ¿Sería útil el uso de un esquema con memoria?
3. Implemente la función `picopala(j,V)` del aparatado b).
4. Si es posible implemente una función recursiva lineal final.
5. Si es posible implemente una función iterativa.
6. Defina tamaños del problema y calcule los T(n) para las distintas funciones implementadas considerando los casos mejor, peores y medio.

NOTAS:

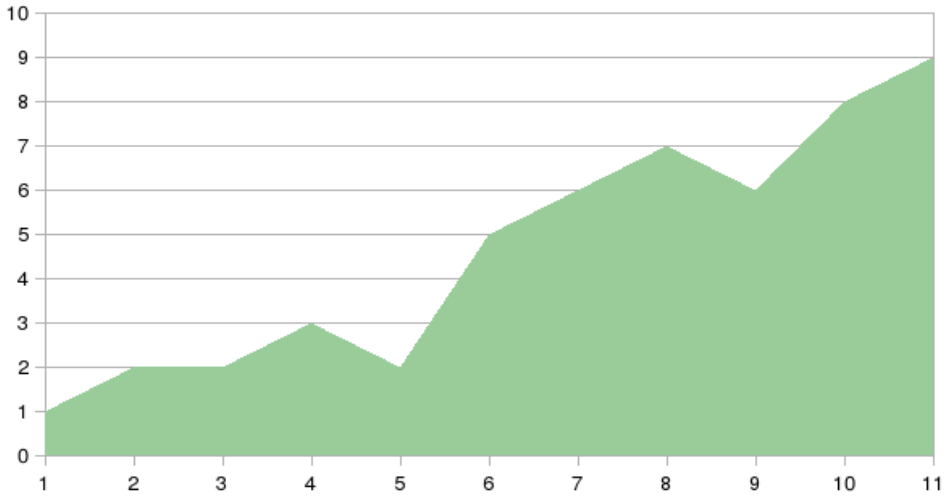
1) Para todos los apartados la llamada inicial será `pico(TAM-1,&V)` o `picopala(TAM-1,&V)`.

2) Emplee el lenguaje de programación que le ha asignado su profesor.

Problema 5: Etapa de Montaña

Enunciado

Se pretende implementar un algoritmo que permita determinar el desnivel existente en una etapa ciclista de montaña. Para ello, representamos la etapa mediante una serie de puntos y su altitud. Por ejemplo, con los siguientes valores de altitud para 10 puntos dentro de una etapa: (1,2,2,3,2,5,6,7,6,8,9), representaríamos el siguiente perfil de etapa, colocando los puntos en horizontal y las altitudes en vertical:



Según la imagen, podemos encontrar dos tipos de desniveles: en descenso, cuando un punto tiene mayor altura que el punto siguiente; y en ascenso si ocurre lo contrario. En el ejemplo existen dos *desniveles en descenso: del punto 4 al 5 y del punto 8 al 9.*

Preguntas

1. Defina la función recursiva no final *num_descensos* que permita determinar número de desniveles de descenso que se producen en una etapa
2. Implemente una función **recursiva no final** que solucione el problema planteado, partiendo de la función *num_descensos* definida previamente, empleando el lenguaje de programación que le ha indicado su profesor.
3. Implemente una función **recursiva final** a partir de la definición anterior, empleando el lenguaje de programación que le ha indicado su profesor.
4. A partir de las funciones anteriores, diseñar e implementar un **algoritmo iterativo** que solucione el problema planteado, empleando el lenguaje de programación que le ha indicado su profesor.
5. Defina el tamaño del problema, calcule el T(n) y la complejidad del algoritmo.

Problema 6: Divisores

Enunciado

La factorización de números enteros consiste en descomponer un número compuesto (no primo) en divisores no triviales que cuando se multiplican dan el número original.

Para nuestro propósito académico queremos implementar una función recursiva que devuelva el número de divisores distintos de un número dado. Por ejemplo, el número 40 tiene 8 divisores: 1, 2, 4, 5, 8, 10, 20 y 40.

Preguntas

1. Defina la función recursiva no final que permita determinar la cantidad de divisores distintos de un número dado (explicación del enunciado).
2. Implemente una función **recursiva no final** que solucione el problema planteado, empleando el lenguaje de programación que le ha indicado su profesor.
3. Implemente una función **recursiva final** a partir de la definición anterior, empleando el lenguaje de programación que le ha indicado su profesor.
4. A partir de las funciones anteriores, diseñar e implementar un **algoritmo iterativo** que solucione el problema planteado, empleando el lenguaje de programación que le ha indicado su profesor.
5. Defina el tamaño del problema, calcule el T(n) y la complejidad del algoritmo.