Estructuras de Datos y Algoritmos 1

Práctico 2

Tema: Recursividad

Escucho y olvido, veo y recuerdo, hago y aprendo.

Confucio

Objetivos:

- Familiarizarse con el desarrollo de programas recursivos.
- Comprender el funcionamiento de los programas recursivos, el papel de los parámetros pasados por valor, por referencia y las variables locales.
- Ejercitarse en el cálculo del tiempo de ejecución de los algoritmos recursivos.

Ejercicios:

Recursión Básica

1) Resolver los ejercicios pendientes del teórico es parte del práctico.
2) Diseñar una rutina listar1An(n) que liste los números del 1 al n. ¿Podría lograr que los liste en orden inverso? Calcule su orden. ¿Cómo compara esta implementación con la equivalente y más obvia en base a una estructura de control repetitiva?
3) Defina recursivamente las funciones:
   p(a,b) = a*b ; m(a,b)= a mod b ; c(a,b)= a div b; mcd(a,b).
1) Implementar recursivamente el cálculo de los números de Fibonacci. Los números de Fibonacci constituyen una serie donde el de orden 0 es 0, el de orden 1 es 1 y los siguientes se calculan como la suma de los dos anteriores. Realizar el diagrama de llamadas para fib(n). Calcule los números de Fibonacci de orden 2..50.
2) Escriba una función que imprima por pantalla cuántas veces se calcula fib(i) en el cálculo de fib(j) con j > i.
3) Diseñar una rutina recursiva promedio (VEC, N) que retorne el promedio de los valores del array VEC de N elementos. Una forma de determinar la suma de N elementos es iterar sobre ellos calculándola. Otra forma es partir el problema, por ejemplo en dos. Sumar la parte izquierda, sumar la parte derecha y calcular la suma total como la suma de la parte izquierda más la suma de la parte derecha. Implemente el cálculo de la suma de esta manera y calcule su orden.
4)

a) Defina una función recursiva int busco (VEC, izq, der, N, dato) que retorna la posición ocupada por dato sobre el array VEC entre las posiciones izq y der de vec o –1 si no se encuentra el dato.

b) Calcule el orden de la implementación de la parte a) Una forma de resolver el problema podría ser partir el problema en un sub problema izquierdo y uno derecho y buscar en la parte derecha y en la parte izquierda. Resuelva el problema de esta manera y calcule su orden.

c) ¿Ayudaría que los elementos de VEC se encontraran ordenados?

d) Implemente una versión mejorada basada en el supuesto de que los elementos de VEC se encuentran ordenados y calcule su orden.

5) Desarrollar una función que reciba un array y retorne su mínimo elemento.

6) Idem a 5 pero que retorne la posición del mínimo.

7) Usando 6 implemente una función recursiva que ordene un array.

Recursión Avanzada

8) Una manera de ordenar un array es dividirlo en una parte izquierda y una parte derecha, ordenar ambas y luego intercalarlas manteniendo el orden. Este algoritmo se conoce como clasificación por intercalación o Mergesort y será estudiado, junto a otros algoritmos para clasificar en profundidad en el próximo curso. Implemente una versión recursiva de este algoritmo y calcule su orden.

9) Implementar recursivamente el cálculo de a^b con a real y b entero. ¿Puede mejorar la cantidad de multiplicaciones necesarias? Realizar el diagrama de llamadas para pot(a,b).

10) Eratóstenes construyó una criba para determinar los números primos del 2 al 100 de la siguiente manera:

a) Sobre una madera escribió los números.

b) Recorría la misma secuencialmente buscando el primer número no perforado. Al encontrarlo, perforaba todos sus múltiplos y repetía el paso anterior a partir del siguiente, hasta terminar.

Implemente una versión estrictamente recursiva (sin usar estructuras de control repetitivas) de este algoritmo.

11) Diseñar una rutina inserto(VEC,N,CLAVE) que inserta CLAVE en el vector ordenado VEC de 100 elementos. El vector es de 100 elementos, pero contiene datos hasta el lugar N.

12) Una forma de determinar si un número n es múltiplo de 9 se basa en sumar sus cifras. Por ejemplo

N = 12345678901234567890123456756789432 1
1+2+3+4+5+6+7+8+9+0+1+2+3+4+5+6+7+8+9+4+3+2+1=100
1+0+0 = 1 no es un múltiplo de 9.

Programar una función booleana bool mult9 (int n) que indique si n es o no un múltiplo de 9.

13) Diseñar una función booleana primo(n) que devuelve true o false según su parámetro sea primo o no.

14) Desarrollar una función int ISLAS (MAPA, M, N) que recibe un mapa representado por una matriz de MxN donde hay un 0 en los lugares en que hay agua y un uno donde hay tierra y contesta la cantidad de ISLAS que contiene. Una isla es un conjunto de unos adyacentes (ortogonal u oblicuamente).

0	0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	0	0	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	1	0	1	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1
0	0	0	1	1	1	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	1	1	0	0	1	0	1	0
0	0	0	0	0	0	0	0	0	1

Por ejemplo en el “mapa” anterior la respuesta sería 6.

15) Desarrollar una función booleana que dados dos puntos del mapa definido en el ejercicio anterior conteste si es posible ir de uno al otro sin pasar por ningún punto de agua. ¿Podría responder en O(1)?

16) Desarrollar implementaciones recursivas que permitan realizar sumas de vectores y matrices de igual dimensión, y producto de matrices conformables.

17) Desarrolle una función:
void MinMax(VECTOR v,int izq,int der, float *min, float *max)
Que devuelva en *min y *max el mínimo y el máximo del vector v entre las posiciones izq y der.

18) Una antigua leyenda cuenta que en un monasterio de Hanoi, los monjes disponen de 64 discos perforados de diferente tamaño, que superpuestos en orden descendente de tamaño, conforman una torre. Tienen, además tres ejes uno de origen, otro de destino y un tercer eje que puede ser utilizado como auxiliar.
El problema que los monjes pretenden resolver es encontrar la secuencia de movimiento de discos que permita trasladar la torre del eje origen al destino con la condición de que en ningún momento se apoye un disco sobre uno de menor tamaño. Los monjes realizan un movimiento por día y esperan que cuando concluyan se acabe el mundo. ¿Debemos preocuparnos ante la eventualidad de que tengan razón?

19) Proporcione una implementación recursiva de los algoritmos de ordenación insert sort, quicksort y mergesort..

Demostraciones

20) Encontrar las funciones que definen las siguientes recurrencias:

1) q(0)=0 , y si n >=1 q(n)=q(n-1)+2n - 1
2) r(0)=1, y si n>=1 r(n)=2r(n-1)
3) s(0)=1, y si n>=1 s(n)=s(s(n-1)-1)+1

Nota: Se recomienda utilizar expansión de recurrencias

21) Mostrar que el enésimo número de Fibonacci fib(N) puede calcularse como f(0,1,n) definiendo f(a,b,0) = a y si c > 0, f(a,b,c) = f(b,a+b,c-1).

Adicionales

- **Los del final del capítulo 2 del libro:**
  Estructuras de Datos y Análisis de Algoritmos.
  Mark Allen Weiss.
- **Los del final del capítulo 1 del libro:**
  Estructuras de Datos y Algoritmos.
  A. Aho, J. E. Hopcroft & J. D. Ullman
  (Capítulo 9: recurrencias**)**
- Calcular el tiempo de ejecución y el orden (O) de los algoritmos de ordenación: merge-sort, insert-sort, quicksort y select-sort. Compararlos.