CEMAS

Fe y Alegría

Ejercicios de Desarrollo de Aplicaciones:

Funciones y Recursividad- Conceptos Básicos 2

Prof. Lizandro Ramírez

1. ES_MULTIPLO: Escribir una función, isMultiple(n, m), esta debe tomar dos números enteros y retornar true si n es un multiplico de m, que es, n == m*x para un entero x, y falso de otra manera. Ejemplo:

   input: n = 6 , m = 3

   output: true, 3*2 = 6

   input: n = 9, m= 5

   output: false

2. ES_PAR: Escribe una función, isEven(k), que tome un entero como valor y retorne true si k es par, y falso si no lo es. Sin embargo tu función no puede usar operadores de multiplicación, modulo o división.

   input: k = 6

   outpu: true

3. MINMAX: Escribe una función, minMax(data), data es un arreglo de números enteros, debes retornar el otro arreglo de números enteros con el max y min del arreglo original, no puedes usar Math.max y Math.min.

   input: data = [8,1,3,7,28,45,0,6,-1]

   output: [-1, 28]

4. CONVERTIR_DEC_HEX: Dado un número en base Hexadecimal crea una función, decToHex(n), donde n es el número base 10, convertir este número a hexadecimal, sin usar funciones pre hechas de librerías como Math. Ejemplo:

   input: 255

   output: “FF”

5. Estructuras de Datos (Problema de investigación 20 pts): Investigar las propiedades esenciales e implementar las siguientes estructuras de datos usando su lenguaje de preferencia (es su caso Javascript) (Escribir su opinión personal

sobre cada estructura tomando en cuenta que problemas resuelven mejor que otras):

a. Pilas
b. Colas
c. Arreglos
d. Arboles
e. Listas enlazadas.
f. Grafos
g. Heap
h. Stack

6. TWOSUM: Realizar una función que resuelva el famoso problema de la suma de dos números dentro de un arreglo que coincida con un target dado. El algoritmo debe retornar el índice de estos dos elementos. Ejemplo:

   input : arr → [3, 1, 2, 4, 8, 11], target → 7

   output: [0, 3]

7. CONVERTIR_HEX_DEC: Dado un número en base Hexadecimal (en formato string) crea una función, hexToDec(st), donde st es el número base 16, convertir este número a decimal, sin usar funciones pre hechas de librerías como Math. Ejemplo:

   input: “FF”

   output: 255

8. UNIQUENUMBER: Escribe una función, uniqueNumber(data), que reciba una arreglo de números enteros y determine si todos los números en el arreglo son distintos uno del otro. Ejemeplo:

   input: data = [1,8,7,3,9,5,2]

   output: true

9. POSIBLES_STRING: Escribe una función, posibleStrings(st), que reciba el string st y de como salida todos las posibles strings que se pueden formar con estos caracteres sin repetirse. Ejemplo:

   input: cat

   output: [cta, cat, atc, act, tca, tac]

10. TECNICAS_PARA_SOLUCION_ALGORITMOS (Problema de investigación 20 pts): Explicar en que consisten cada una de las siguientes técnicas comunes en la solución de algoritmos.

a. Pattern: Sliding Window
b. Pattern: Two Pointers
c. Pattern: Fast & Slow pointers
d. Pattern: Merge Intervals
e. Pattern: Cyclic Sort
f. Pattern: In-place Reversal of a LinkedList
g. Pattern: Tree Breadth First Search
h. Pattern: Tree Depth First Search
i. Pattern: Two Heaps
j. Pattern: Subsets
k. Pattern: Modified Binary Search
l. Pattern: Bitwise XOR
m. Pattern: Top 'K' Elements
n. Pattern: K-way merge
o. Pattern : 0/1 Knapsack (Dynamic Programming)
p. Pattern: Topological Sort (Graph)

11. CUENTAVOCALES: Escribe una función, countVowels(st), que reciba un string y te retorne el número de vocales que contiende dicho string. (Investigar Tabla ASCII y UNICODE como usarlos en JS). Ejemplo:

    input: “COCOTEROSNO”

    output: 5

12. UNIQUESTRING: Crear una función que reciba un String (st), retorne (true) si el st posee todos sus caracteres distintos, o retorne (false) si existen repeticiones. Ejemplo:

    input : st → “consta”

    output: true

    input : st → “constan”

    output: false

13. FIND_DUPLICATE_NUM: Crear una función que reciba un arreglo de números de tamaño n+1, los elementos dentro del arreglo esta dentro del rango de 1 a n. Encontrar el valor duplicado de la manera eficiente. Siempre existir aun duplicado en el arreglo con estas características. Ejemplo:

    input : arr → [2,3,4,1,5,2]

    output: 2

14. CHECK_PERMUTATION: Dados dos Strings, escriba una función que decida si uno es una permutación de otro. El problema no es sensible a mayúsculas por ejemplo: (Dog) es permutación de (god). Las cadenas no deben poseer espacios en blanco. Ejemplo:

    input : st1 → “dog” y st2 → “god”

    output: true

    input : st1 → “gato” y st2 → “tog”

    output: false

15. STRING_COMPRESSION: Implementar una función, stringCompression(st) que comprima un string usando la cuenta de los caracteres repetidos. Ejemplo:

    input: st = “aabccccccaaa”

    output: “a2b1c5a3”

16. TORRES_HANOI (Problema de investigación 20 pts sobre recursividad): Imagina que tienes 3 postes llamados A, B y C. En el poste A tienes n discos de diferente diámetro, acomodados en orden creciente de diámetro desde lo más alto hasta lo más bajo. Solamente puedes mover un disco a la vez desde un poste hasta otro y no está permitido poner un disco más grande sobre otro más pequeño. Tu tarea es mover todos los discos desde el poste A hasta el poste C.
    i. Escribe una función, hanoi(n), que reciba como parámetro n y que imprima en pantalla todos los pasos a seguir para mover los discos del poste A al poste C.

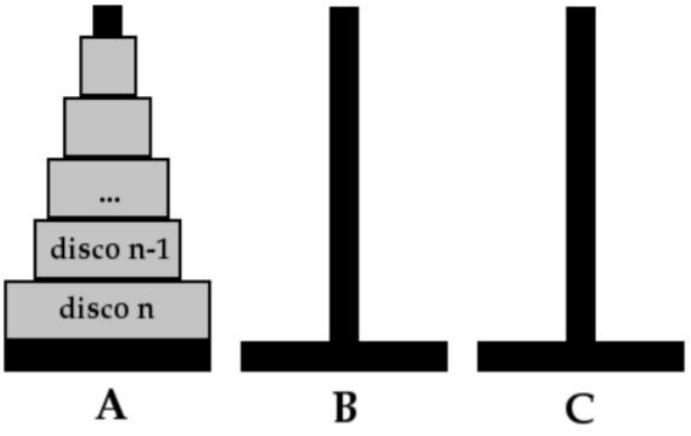

Figura 4.2: Tórres de Hanoi

Nota: Usando la estrategia divide y vencerás, crear una función mueve(n, “A”, “B”, “C”), donde ABC representan las distintas torres.

Una leyenda cuenta que en un monasterio en la ciudad de Hanoi hay 3 postes colocados de esa manera y unos monjes han estado trabajando para mover 48

discos del poste A al poste C, una vez que terminen de mover los 48 discos el mundo se acabará. Te parecen pocos 48 discos?, corre la solución a este problema con n=48 y verás que parece nunca terminar(ahora imagina si se tardaran 2 minutos en mover cada disco ¿Cuanto le tiempo durarían?).

17. ONE_WAY: Hay 3 formas de edición que pueden ser aplicadas en un string: insertar un carácter, remover un carácter, o reemplazar un carácter. Dados dos strings, escribe una función, oneWay(st, ts), que verifique si hay una edición aplicada entre un string y el otro. Ejemplo:

    pale, ple → true

    pales, pale → true

    pale, bale → true

    pale, bake → false

18. URLIFY: Escribe una función que remplace todos los espacios en blanco por '%20', debes suponer que el string tiene suficiente espacio para agregar los nuevos caracteres, el tamaño del string debe mantenerse igual que el string original. Ejemplo:

    input : st1 → "Mr John Smith " y tam → 13

    output: "Mr%20John%20Smith" y tam → 13

19. Escribe una función que dados dos números enteros n y c, cadenas(n,c), imprima todas las cadenas de caracteres de longitud n que utilicen solamente las primeras c letras del alfabeto(todas minúsculas), puedes asumir que n < 20. Ejemplo:

    cadenas(1, 3) = {a, b, c}

    cadenas(2, 3) = {aa, ab, ac, ba, bb, bc, ca, cb, cc }

    cadenas(3, 3) = {aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc, }

20. Algoritmos de Ordenamiento (Problema de investigación 20 pts): Investigar e implementar los siguientes algoritmos de ordenamiento usando su lenguaje de preferencia (es su caso Javascript) (Escribir su opinión personal sobre la estrategia utilizada en cada algoritmo):
    a. Algoritmo de Selección.
    b. Algoritmo de Inserción.
    c. El Odiado algoritmo de la Burbuja.
    d. Ordenamiento en O(n log n) por Mezcla (Merge).
    e. Ordenamiento con Heap.