AYUDANTÍA 0: ESTRUCTURAS DE DATOS Y ALGORITMOS $2^{\underline{\text{do}}}$ Semestre 2020

Profesor: Javier Carrión
Ayudante: Yerko Ortiz

Objetivo de la ayudantía: Introducción y motivación al curso.

Preguntas conceptuales

1. ¿Cómo definiría usted un algoritmo?
2. ¿Por qué cree usted que los algoritmos son importantes?
3. ¿Qué es una estructura de datos?, ¿conoce alguna?
4. ¿El lenguaje de programación importa para estudiar algoritmos y estructuras de datos?
5. ¿Cuál es la diferencia entre un algoritmo recursivo y uno iterativo?

Un poco de Java

En los problemas de algoritmos del curso, será común encontrarnos con enunciados que expliquen el problema, un conjunto de datos de entrada y un conjunto de datos de salida. Estos últimos son los resultados que sus algoritmos deben arrojar.

Ejercicio 1

Implemente un programa en Java que como entrada reciba lo siguiente:

- La primera linea contiene un entero N que corresponde al largo de un arreglo.
- La segunda linea contiene N enteros que corresponden a los elementos del arreglo.

Y como salida su programa debe entregar:

- La suma de los elementos en el arreglo

Ejercicio 2

Implemente un programa en Java que imprima un String en orden inverso.
Por ejemplo sea la palabra Hola, entonces su programa debe retornar aloH.
Formato de la entrada

- Una linea con un String S

Formato de la salida

- El string S invertido.

"Premature optimization is the root of all evil."

Donald Knuth

Ejercicio 3

Implemente un programa en Java que verifique si un número entero es par
Su programa debe imprimir PAR en caso que el entero sea divisible por dos, IMPAR en caso complementario.
Formato de la entrada

- Una linea con un entero N

Formato de la salida

- imprimir PAR en caso que el entero sea divisible por dos, IMPAR en caso complementario.

Compilador mental

Usando lapiz y papel como memoria y su imaginación como unidad de procesamiento. Ejecute los siguientes códigos con las entradas respectivas y describa lo que dichos códigos hacen.

Algoritmo 1

```
1 static int algo(int a, int b)
2 {
3   if(b == 0) return a;
4   return algo(b, a %b);
5 }
```

Entrada: 123, 51
Output:
Descripción:

Algoritmo 2

```
1  void algo(int arr[])
2     {
3         int n = arr.length;
4         for (int i = 0; i < n-1; i++) {
5             int min_idx = i;
6             for (int j = i+1; j < n; j++)
7                 if (arr[j] < arr[min_idx]) min_idx = j;
8             int temp = arr[min_idx];
9             arr[min_idx] = arr[i];
10            arr[i] = temp;
11        }
12    }
```

Input: 2, 3, 1, 5, 6, 7
Output:
Descripción:

Bit Strings

En el sistema numérico decimal se representan las cantidades haciendo uso de diez símbolos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} por ejemplo el número tres lo representamos con 3, o el número doce lo representamos con 12. En sistema binario por otro lado las cantidades se representan haciendo uso de los símbolos {0,1}, donde por ejemplo el número tres se representa con 11 y el número 12 se representa con 1100.
La codificación binaria en sistema decimal representa una suma de potencias de dos multiplicadas por el dígito correspondiente.
11 = 1 x 2 + 1 x 1 = 3
1100 = 1 x 8 + 1 x 4 + 0 x 2 + 0 x1
Es posible generalizar y decir que cualquier entero n de base 10 se puede escribir como:

$$\sum_{i=0}^{L} a_i 2^i$$

Donde L es el largo del bitstring y a_i corresponde a la i-ésima posición del bitstring leído de derecha a izquierda.
Diseñe e implemente un algoritmo en Java que transforme un entero N en su representación binaria.

Datos de entrada

- Una linea que contiene un entero N.

Dominio de la entrada

- $0 \leq N \leq 2^{30}$

Datos de salida

- Imprimir el bitstring que representa al entero N.

Casos de prueba

Caso 1	Caso 2	Caso 3
Entrada: 5	Entrada: 9	Entrada: 30
Salida: 101	Salida: 1001	Salida: 11110

Hints para el curso

- D. Knuth en su libro The Art Of Computer Programming menciona que la mejor forma de aprender un algoritmo nuevo es ejecutarlo con lápiz y papel; Es decir que antes de implementar y usar un algoritmo que desconoce, haga el ejercicio al menos una vez de ejecutarlo a mano, de esa forma lo podrá interiorizar.
- Resolver un problema nuevo la mayoría de las veces es difícil, no busque soluciones de otras personas hasta que no le haya dedicado un tiempo razonable!
- La solución a un problema de algoritmos se puede diseñar e implementar de diversas formas, tómese el tiempo de discutir su solución con la de sus compañeros; Esto le otorgará nuevas perspectivas del problema.
- Antes de implementar una solución es importante que la diseñe bien, no programe nada hasta que este seguro de que el diseño es correcto; Posteriormente no se quede solo con el diseño, implemente su solución y verifique que funciona experimentalmente.