

LISTA DE PROGRAMAS ALGORITMOS Y ESTRUCTURAS DE DATOS

De Luna Ocampo Yanina

Fecha de entrega: 19 de junio 2021

Programa 0

Problema 0

$U = \#0003 \rightarrow *U = 301$

$V = \#B002 \rightarrow *V = 201$

$W = \#A001 \rightarrow *W = 101$

$(*W) += a + b + c; \rightarrow 101 += 101 + 201 + 301; \rightarrow *W = 704$

$(*U) ++; \rightarrow 301 ++; \rightarrow 301 + 1; \rightarrow *U = 302$

$(*V) *= 4; \rightarrow 201 *= 4; \rightarrow *V = 804$

$q = w; \quad q = 704;$

$w = u; \quad w = 302;$

$u = v; \quad u = 804;$

$v = q; \quad v = 704;$

$(*u) -= (*q) \% 5 - a; \rightarrow 804 -= (704) \% 5 - 704; \rightarrow 804 -= 4 - 704; \rightarrow 804 -= -700; \rightarrow *u = 1504;$

$(*v) -= (*q) \% 3 - b; \rightarrow 704 -= (704) \% 3 - 1504; \rightarrow 704 -= 2 - 1504; \rightarrow 704 -= -1502; \rightarrow *v = 2206;$

$(*w) -= (*q) \% 2 - c; \rightarrow 302 -= (2206) \% 2 - 302; \rightarrow 302 -= 0 - 302; \rightarrow 302 -= -302; \rightarrow *w = 604;$

`printf("a = %d b = %d c = %d\n", a, b, c);`

Salida:

a = 2206

b = 1504

c = 604

Programa 1

Arreglo Dinamico

Ingrese tamano del arrelgo:5

Direccion de la memoria del arreglo= AF6BA0

arr[0]= 11474240

arr[1]= 0

arr[2]= 11469136

arr[3]= 0

arr[4]= 0

arr[0]= 0

arr[1]= 0

arr[2]= 0

arr[3]= 0

arr[4]= 0

arr= AF6BA0

arr= 0

Process exited after 2.239 seconds with return value 0

Presione una tecla para continuar . . . _

Programa 2

```
Digita una palabra de menos de 10 caracteres: hola
4
aloh  @%↓
hola
-----
Process exited after 1.285 seconds with return value 0
Presione una tecla para continuar . . .
```

Programa 3

Problema 5

```
**q = NULL , *p = NULL;
*a = NULL , *b = NULL , *x = NULL , *y = NULL;
N = 10 , K = 0;

10 ← a = (int *) malloc (N * sizeof (int));
10 ← b = (int *) malloc (N * sizeof (int));

for (x = a , y = b , K = 0 ; K < N ; K++ , x++ , y++) {
18 ← (*x) = 2 * K;
```

```

27 ←      (*y) = 3 * k;
      }

      for (k=0 ; k<N ; k++) {
          if (k%2 == 0)
10 ←      q = *a;
          else
10 ←      q = *b;

10 ←      p = *q;
19 ←      p = p + k;
-20 ←      (*p) * = -1;

-20 ←      (**q) += *p;
      }

      for (k=0 ; k<N ; k++)
          printf ("%X :: a[%d] = %d\n", &(b[k]), k, b[k]);

      printf ("\n");

      for (k=0 ; k<N ; k++)
          printf ("%X :: b[%d] = %d\n", &(b[k]), k, b[k]);

0 ←      free(a);
0 ←      free(b);

```

Programa 4

```

*dinosaurio[0]: 0          direccion: 711680
dinosaurio[1] : {0}
*dinosaurio[2]: 36          direccion: 7116f8
dinosaurio[3] : {222036}
*dinosaurio[4]: 7405614 direccion: 711690

Presione una tecla para continuar . . .

```

Programa 5

```
Hipercubo de 6 [0]:  
    Hipercubo de 5 [0]:  
        Hipercubo de 4 [0]:  
            Volumen [0]:  
                Matriz [0]:0 0 0 0 0 0 0  
                Matriz [1]:0 0 0 0 0 0 0  
                Matriz [2]:0 0 0 0 0 0 0  
                Matriz [3]:0 0 0 0 0 0 0  
                Matriz [4]:0 0 0 0 0 0 0  
                Matriz [5]:0 0 0 0 0 0 0  
            Volumen [1]:  
                Matriz [0]:0 0 0 0 0 0 0  
                Matriz [1]:0 0 0 0 0 0 0  
                Matriz [2]:0 0 0 0 0 0 0  
                Matriz [3]:0 0 0 0 0 0 0  
                Matriz [4]:0 0 0 0 0 0 0  
                Matriz [5]:0 0 0 0 0 0 0  
            Volumen [2]:  
                Matriz [0]:0 0 0 0 0 0 0  
                Matriz [1]:0 0 0 0 0 0 0  
                Matriz [2]:0 0 0 0 0 0 0  
                Matriz [3]:0 0 0 0 0 0 0  
                Matriz [4]:0 0 0 0 0 0 0  
                Matriz [5]:0 0 0 0 0 0 0  
            Volumen [3]:  
                Matriz [0]:0 0 0 0 0 0 0  
                Matriz [1]:0 0 0 0 0 0 0  
                Matriz [2]:0 0 0 0 0 0 0  
                Matriz [3]:0 0 0 0 0 0 0  
                Matriz [4]:0 0 0 0 0 0 0  
                Matriz [5]:0 0 0 0 0 0 0  
            Volumen [4]:  
                Matriz [0]:0 0 0 0 0 0 0  
                Matriz [1]:0 0 0 0 0 0 0  
                Matriz [2]:0 0 0 0 0 0 0
```

```

Matriz [5]:0  20    80    180   320   500   720
Hipercubo de 4 [3]:

Volumen [0]:

Matriz [0]:0  0      0      0      0      0      0
Matriz [1]:0  6      24     54     96    150   216
Matriz [2]:0  12     48     108    192   300   432
Matriz [3]:0  18     72     162    288   450   648
Matriz [4]:0  24     96     216    384   600   864
Matriz [5]:0  30     120    270    480   750   1080

Volumen [1]:

Matriz [0]:0  0      0      0      0      0      0
Matriz [1]:0  6      24     54     96    150   216
Matriz [2]:0  12     48     108    192   300   432
Matriz [3]:0  18     72     162    288   450   648
Matriz [4]:0  24     96     216    384   600   864
Matriz [5]:0  30     120    270    480   750   1080

Volumen [2]:

Matriz [0]:0  0      0      0      0      0      0
Matriz [1]:0  6      24     54     96    150   216
Matriz [2]:0  12     48     108    192   300   432
Matriz [3]:0  18     72     162    288   450   648
Matriz [4]:0  24     96     216    384   600   864
Matriz [5]:0  30     120    270    480   750   1080

Volumen [3]:

Matriz [0]:0  0      0      0      0      0      0
Matriz [1]:0  6      24     54     96    150   216
Matriz [2]:0  12     48     108    192   300   432
Matriz [3]:0  18     72     162    288   450   648
Matriz [4]:0  24     96     216    384   600   864
Matriz [5]:0  30     120    270    480   750   1080

Volumen [4]:

Matriz [0]:0  0      0      0      0      0      0
Matriz [1]:0  6      24     54     96    150   216
Matriz [2]:0  12     48     108    192   300   432
Matriz [3]:0  18     72     162    288   450   648
Matriz [4]:0  24     96     216    384   600   864
Matriz [5]:0  30     120    270    480   750   1080

```

Programa 6

Problema 6

Considere un problema computacional X . ¿Siempre podrá encontrar un algoritmo para poder resolverlo?

Argumente su respuesta en términos de computabilidad.

R: No, ya que los problemas computacionales tienen clases de complejidad, tal como los NP que hasta la fecha no existe un algoritmo que los pueda resolver.

Programa 7

Problema 7

Describe las clases computacionales P, NP, NP-Completo y NP-Difícil. Por cada clase, dé un ejemplo de un problema que pertenezca a la misma.

R: Clase P: representa a los problemas tratables (fáciles) que una computadora pueda resolver, estas pueden encontrar soluciones en un tiempo razonable. Ejemplo: Multiplicación de matrices.

Clase NP: Son problemas que no son fáciles de encontrar una solución, pero una vez que se encuentran es fácil de comprobar. Ejemplo: problema de las 8 reinas.

Clase NP-completo: Contienen los problemas más difíciles en NP, en el sentido de que son los que están más lejos de estar en P, indica que no se conoce la solución en tiempo polinomial. Ejemplo: Problema del viajero.

NP-Difícil: Cuando el método de solución de un problema se puede convertir en un método de solución NP-Completo se dice que es "NP-difícil" como cualquier problema de NP o tal vez más difícil. Ejemplo: Problema de satisfactibilidad booleana.

Programa 8

Problema 8

¿Existe algún problema que se encuentre tanto en la clase P como en la clase NP? Si es así, dé un ejemplo y proponga un algoritmo para resolverlo. ¿Esto significa que ambas clases de complejidad son iguales o son diferentes? ¿Su respuesta puede considerarse como una solución general para responder el problema P vs NP?

R: P y NP son 2 clases de complejidad que agrupan problemas distintos. Un problema que se encuentra dentro podría ser el ordenamiento QuickSort ya que en la mayoría de casos su complejidad es $O(n \log n)$, pero existen raros casos donde su complejidad es $O(n^2)$.

- No considero que mi respuesta anterior sea una solución general ya que existen diferentes clases NP (NP-completos y NP-Difíciles)

Programa 9

Problema 9

```
K=0, x=0, i=0, j=0;
```

```
for(k=0, x=0; k<N; k++)
```

```
    x += k;
```

$$2 + N + 1 + N = 3 + 2N = 2N + 3$$

```
printf("x = %.d\n");
```

$$O(N)$$

```
    x = 1643424
```

```
for(k=1, x=0; k<N; k*=6)
```

```
    x += k;
```

$$2 + N + 1 + \log_6 N = 3 + 2N + \log_6 N$$

```
printf("x = %.d\n");
```

$$O(\log_6 N)$$

```
    x = 6
```

```
for(i=0, x=0; i<N; i+=2)
```

$$2 + N + 1 + N/2$$

```
    for(j=0; j<N; j+=3)
```

$$3 + 2N/2$$

```
        x += i*j;
```

$$1 + N + 1 + N/3$$

```
printf("x = %.d\n");
```

$$3 + 2N/2 + 2 + 2N/3$$

$$5 + \frac{4}{5}N$$

$$O(N)$$

```
    x = 630585904
```

```
for(i=N, x=0; i>0; i/=2)
```

$$2 + N + 1 + N/2$$

```
    for(j=N; j>0; j--)
```

$$3 + 2N/2$$

```
        x += i*j;
```

$$1 + N + 1 + N$$

```
printf("x = %.d\n");
```

$$2 + 2N$$

$$5 + 4N/2$$

$$O(N)$$

```
    x = 0
```

$$3 + 2N/2 + 2 + 2N$$

Programa 10

Demuestre que la complejidad en el peor de los casos del Algoritmo de Kadane, para encontrar la suma más grande de un arreglo de N enteros, es $O(N)$.

Programa 11

```
Bienvenido al programa de ordenamineto
Su arreglo es el siguiente:
8 4 1 6 0 3 25 7 9

    OrdenamientoSeleccion-> 1
    OrdenamientoInsercion-> 2
    OrdenamientoBurbuja-> 3
    OrdenaminetoMezcla-> 4

Por favor digite el numero para el metodo que quiera utilizar:2

el arreglo se ordeno por insercion:
0 1 3 4 6 7 8 9 25

-----
Process exited after 1.09 seconds with return value 0
Presione una tecla para continuar . . .
```

Programa 12

Problema 12

Búsqueda Binaria Iterativa

Para llegar al conjunto de búsqueda de tamaño 1, hacemos " x " iteraciones. Entonces:

$$\frac{N}{2^{x-1}} = 1$$

$$N = 2^{x-1}$$

$$x = 1 + \log_2 n$$

$$\text{Complejidad} = O(\log n)$$

```
Ingrese el numero a buscar: 4
```

```
el numero esta en la celda: 2
```

```
-----
```

```
Process exited after 2.944 seconds with return value 0
```

```
Presione una tecla para continuar . . . █
```

```
Ingrese el numero a buscar: 83
```

```
el numero esta en la celda: 16
```

```
-----
```

```
Process exited after 1.364 seconds with return value 0
```

```
Presione una tecla para continuar . . .
```