**UNIVERSIDAD DEL NORTE**

DIVISIÓN DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

ALGORITMOS Y COMPLEJIDAD

CÓDIGO:\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/\_\_\_/

NOMBRE: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FECHA : \_\_\_\_/\_\_\_\_/\_\_\_\_/ ..

Resolución de las Prácticas de Computadores, Complejidad e Intratabilidad.

Ej 1: Complete la gráfica del modelo representativo de la resolución de problemas del mundo real aplicando la Teoría de Algoritmos:

MUNDO REAL

Problemas ( ∏ ): Buscar / Ordenar / Graficar / Calcular / Conectar / Agregar / Disgregar / Consultar / ….

TEORÍA DE ALGORITMOS :

Algoritmo : AFD / MT

Problema a ser solucionado ( ∏ ):

Búsqueda en un archivo

Buscar en una B. de D.

Impráctica:

Práctica :

Programa Optimo

Pi ++

SOLUCIÓN!!!

Tipos de Algoritmos ( ¿? )

Algoritmo OPTIMAL (¿?) : Ai ++

Tiempo / Espacio

Paradigma de Programación (¿?)

Programación Funcional =

Programación Lógica =

Ej 2: Ubique en el modelo representativo de la resolución de problemas del mundo real aplicando la Teoría de Algoritmos, el ciclo de vida del software: ANÁLISIS / DISEÑO / CODIFICACIÓN / PRUEBA / OPERACIÓN / MANTENIMIENTO.

1. Clasificar los siguientes problemas según su naturaleza.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **∏** | **N , ¬N** | **Op. B.** | **U. M.** | **E.D.** | **OC** |
| 1 | n! / n ∈ Z+ | N | multiplicacion | n | bit |  |
| 2 | Ordenar una lista de K números |  |  |  |  |  |
| 3 | MCD(n,m) / (n,m) ∈ Z+ |  |  |  |  |  |
| 4 | Search (x) en lista lineal de k nodos de longitud L |  |  |  |  |  |
| 5 | Con Mi de dimensión | N | Suma | (k-1)\*n^2 | Matriz |  |
| 6 | Recorrido Hamiltoniano del caballo de ajedrez | --N, N | Comparación | Tablero | tabla |  |
| 7 | Coloreado de un grafo G=(V,E), de V nodos y E arcos |  |  |  |  |  |
| 8 | Multiplicación de Polinomios |  |  |  |  |  |
| 9 | Dado G=(V,E) con pesos asociados en sus arcos w(e) ∈ Z+ para cada arco a e ∈E, y un K∈ Z+  P= Puede ser V particionado en dos conjuntos disyuntos V1,V2 / suma de los arcos de E en cada conjunto sea al menos K? |  |  |  |  |  |
| 10 | Traductor a Código Internacional Morse |  |  |  |  |  |
| **No.** | **∏** | **N , ¬N** | **Op. B.** | **U. M.** | **E.D.** | **OC** |

1. Chequear la efectividad de los siguientes segmentos de algoritmos, en el caso de que tengan alguna funcionalidad:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Proc: \_\_Divisores de n\_\_\_\_\_\_\_\_\_\_  Lea n // n , j definidos como enteros  j ← 1  HH  Si (n mod j = 0) entonces  Escriba j  Fin\_si  j←j+1  Fin\_HH (j > n/2)  Escriba n  Fin\_proc | 2. | Fun \_\_\_\_\_\_\_\_ (m.n)  r ← 0  HH  Si (m mod 2 < > 0 ) entonces  r ← r + n  Fin\_si  m ← m/2 // división entera  n ← n + n  Hasta\_que (m=1)  Devolver r  Fin\_Fun |
| 3. | Proc: \_\_\_\_Primo\_\_\_\_\_\_\_  Lea n // n,j son enteros  j←2  Sw←falso  Mq ((j<=n-1) y ( Sw=falso)) haga  Si ( n mod j = 0 ) entonces  Sw ← verdadero  Fin\_si  j ← j+1  Fin\_Mq  Si (Sw = falso) entonces  Escriba “Número \_\_\_\_\_\_\_”  Si\_no  Escriba “Número \_\_\_\_\_\_\_”  Fin\_si  Fin\_proc | 4 | Funcion Imax(A,n)  m←A[1];k←1  Para i =2.n,1 haga  Si (A[i]>m) entonces  m←A[i]  k←i  Fin\_si  Fin\_para  Devolver k  Fin\_Imax |
| 5 | Proc: Cálculo de \_\_\_\_\_\_\_\_\_\_  Lea n  s ← 1  Si (n>0) entonces  Para i=2,n,1 haga  s ← s\*i  Fin\_para  Si\_no  Escriba “\_\_\_\_\_\_\_\_\_\_\_\_\_\_”  Fin\_si  Fin\_proc | 6 | Proc: \_\_palindrome\_\_\_\_\_\_\_\_\_\_  Lea n  Para i=1,n,1 haga  Lea A[i]  Fin\_para  Sw←1;m←[n/2] ;j←i;i←1 // Op. parte entera[ ]  Mq((Sw=1) y (i<=m)) haga  Si (A[j] < > A[i]) entonces  Sw ← 0  Fin\_si  i ← i+1;j←j-1  Fin\_Mq  Si (Sw=1) entonces  Escriba “La estructura es\_\_\_\_\_\_\_\_\_\_\_\_”  Si\_no  Escriba “La estructura no es\_\_\_\_\_\_\_\_\_\_\_\_”  Fin\_si  Fin\_proc |

1. Colocar un nombre significativo a los siguientes algoritmos en el supuesto caso de que tengan alguna funcionalidad:

**Proc1**: ­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_cuenta los digitosde un numero\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lea n

n1 ← n

cd ← 0

Mq ( n1 > 0 ) haga

n1 ← n1/10 // División entera

cd ← cd + 1

Fin\_Mq

Escriba “El \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_”n

Escriba “\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ =”cd

**Fin\_proc1**

**Proc2**: \_\_\_\_\_\_\_converso de numero en binario\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lea n // n ∈ Z+

n1 ← n; j ← 0

Mq ( (n1/2) < > 1 ) haga

j ← j + 1

R(j) ← n1%2

n1 ← n1/2

Fin\_Mq

Si (n1=3) entonces

j ← j +1

R(j) ← 1

j ← j + 1

R(J) ← 1

Si\_no

j ← j+1

R(j) ← 0

j ← j + 1

R(j) ← 1

Fin\_si

Para i = j hasta 1 con incrementos de -1 haga

Escriba R(j)

Fin\_para

**Fin\_proc2**

**Proc3**:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (u,v)

Mq ( u > 0 ) haga

Si ( u < v ) entonces

m ← u

u ← v

v ← m

Fin\_si

u ← u – v

Fin\_Mq

Retorne v

**Fin\_proc3**

1. Diseñar la función de tiempo de corrida (Running Time) del siguiente segmento de programa, en función del número de datos de entrada, teniendo en cuenta que cada instrucción (Sti) consume ti unidades de tiempo.

Proc: Genérico1

St1

St2 // Lee el valor de n

Para i=1,n,1 haga

St3

St4

St5

Fin\_para

St6

St7

Fin\_proc

R/ Sea el tiempo de demora del algoritmo en función del número de datos de entrada

Si cada consume una unidad de tiempo, entonces el tiempo del algoritmo es

Tiepo lineal univeriado determinístico.

1. Construir la función de tiempo de corrida (Running Time) del siguiente segmento de programa, en función del número de datos de entrada, teniendo en cuenta que cada instrucción (Sti) consume una unidad de tiempo y la proposición tres se cambia según se muestra en el texto del algoritmo.

Proc: Genérico2

St1 // Lee el valor de n

St2 // Lee el valor de k

Para i=1,n,1 haga

Para j=1,k,1 haga

St4

St5

Fin\_para

Fin\_para

St6

St7

Fin\_proc

R/, Sea el tiempo de corrida de Genérico 2.

2\*n\*m+4

1. Construir la función de tiempo de corrida (Running Time) del siguiente segmento de programa, en función del número de datos de entrada, teniendo en cuenta que cada instrucción (Sti) consume una unidad de tiempo y la proposición tres se cambia según se muestra en el texto del algoritmo.

Proc: Genérico3

St1 // Lee el valor de n

St2

Para i=1,n,1 haga

Para j=1,i,1 haga

St4

St5

Fin\_para

Fin\_para

St6

St7

Fin\_proc

R/. Sea el tiempo de corrida de Genérico 3.

1. Diseñar la función de tiempo del siguiente segmento de algoritmo, que busca el elemento x en una vector L[`1..n] elementos, suponiendo que elemento x se encuentra en el arreglo.

Search(L,n,x)

i←1

Mq(x < > L(i) and i<=n) haga

i←i+1

Fin\_Mq

Si (x=L(i)) entonces

Devolver i

Si\_no

Devolver 0

Fin\_si

Fin\_Search

Nota: El algoritmo no es determinístico sino **probabilístico**´. Luego, se debe calcular el comportamiento promedio (p)

1. Dado el siguiente algoritmo. considerando que la instrucción Si consume una unidad de tiempo, calcule el tiempo T(¿?) de corrida del algoritmo validando el funcionamiento del algoritmo.

Proc: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Para (i=1,n-1,1) haga

m ← i

Para (j=i+1,n,1) haga

Si (A[j] < A[m] ) entonces

m ← j

Fin\_si

Temp ← A[m]

A[m] ← A[i]

A[i] ← Temp

Fin\_para

Fin\_proc

1. Comparación del orden de complejidad O(n) de los tres segmentos de programa. Concluya la complejidad de los segmentos de programa en función del cálculo de la función de tiempo del algoritmo.

|  |  |  |
| --- | --- | --- |
| Proc: Genérico1  St1  St2 // Lee el valor de n  Para i=1,n,1 haga  St3  St4  St5  Fin\_para  St6  St7  Fin\_proc | Proc: Genérico2  St1 // Lee el valor de n  St2 // Lee el valor de m  Para i=1,n,1 haga  Para j=1,m,1 haga  St4  St5  Fin\_para  Fin\_para  St6  St7  Fin\_proc | Proc: Genérico3  St1 // Lee el valor de n  St2  Para i=1,n,1 haga  Para j=1,i,1 haga  St4  St5  Fin\_para  Fin\_para  St6  St7  Fin\_proc |
|  |  |  |
|  |  |  |
|  |  |  |

1. Dado el siguiente texto algorítmico, diseñar la función de tiempo T(¿?) del algoritmo en función del número de datos de entrada considerando que la instrucción Si consume una unidad de tiempo, y calcular el tiempo cuando el valor de n = 100 .

Proc \_\_\_\_\_\_\_\_\_\_ ( A,n ) // A es un arreglo de n elemento

Para j = n hasta 2 con incrementos de -1 haga

Corrimiento (A,j)

fin\_para\_j

Fin\_pro;

Proc Corrimiento (A,j )

Para i = 1 hasta j-1 haga

Si A(i) < A(i+1) entonces

temp ← A(i)

A(i) ← A(i+1)

A(i+1) ← temp

fin\_si

fin\_para\_i

fin\_proc\_corrimiento;

Sea T(n) el tiempo de demora del procedimiento.

Recordando que:

+uv