Tecnológico de Monterrey, Campus Monterrey
Departamento de Ciencias Computacionales
ANALISIS Y DISEÑO DE ALGORITMOS - Ing. Román Martínez M.
PRIMER EXAMEN DE PROGRAMACIÓN - 13 de Octubre 2015

Nombre: _____Matrícula: _____

Instrucciones:

- Para resolver este examen puedes utilizar el compilador de C++ de tu preferencia para programar tus respuestas.
- Se te provee de un archivo llamado `ex1ad15.cpp` en donde se encuentra cierto código que utilizarás y en ese mismo archivo programarás tus respuestas.
- Identifica tu archivo con un comentario inicial con tu nombre y matrícula, y documenta el código con los comentarios que consideres importantes y necesarios para explicar lo que programas.
- El archivo final de entrega como respuesta de tu examen, deberá subirse a la página del curso en el espacio correspondiente.
- Está estrictamente PROHIBIDO el acceso a códigos previamente realizados y cualquier tipo de comunicación electrónica o presencial. Cualquier situación deshonesta será penalizada con una calificación de DA.

Revisa el código fuente que se encuentra en el archivo `ex1ad15.cpp`. Podrás observar que se encuentran definidos algunos arreglos que utilizarás y que ya está la implementación del algoritmo de Floyd y parcialmente el algoritmo de Godbole. Usa este código para resolver los problemas 2 y 3 que se presentan a continuación. El problema 1 no requiere de ningún código adicional. Para probar el programa, deberás seguir las instrucciones que se indican y responder las preguntas asociadas a cada problema. El ALGORITMARIO contiene únicamente el algoritmo de la búsqueda binaria por si te fuera útil.

PROBLEMA 1. *(35 puntos – Tiempo sugerido para solucionar el problema: 30 min).*

En cierta aplicación, se cuenta con un arreglo de datos ya ordenado al cual se le aplica cierto número de veces un proceso de **rotación circular a la derecha**. Este proceso consiste en recorrer todos los datos una posición hacia adelante en el arreglo, y que el último dato pase a ser el primero en el arreglo. Por ejemplo, el arreglo: 2, 4, 6, 8, 10, después de realizarle 3 rotaciones circulares a la derecha quedaría: 6, 8, 10, 2, 4.

Implementa una función que dado un arreglo ya rotado, sirva para identificar cuántas rotaciones circulares a la derecha se le aplicaron. El algoritmo que se implemente deberá tener un orden logarítmico O(log n). Considera que la cantidad de veces que se rotó el arreglo es menor al tamaño del arreglo.

Utiliza los arreglos `datos1` y `datos` 2 para probar tu programa y responder lo siguiente:

- ¿Cuántas rotaciones se realizaron al arreglo `datos1`? 146 ¿y al arreglo `datos2`? 1
- ¿Cuántas comparaciones realiza tu algoritmo para encontrar la respuesta con el arreglo `datos1`? 6 ¿y con el arreglo `datos2`? 6
- ¿Cuál es la técnica de diseño de algoritmos que utilizaste? busqueda binaria
- ¿Cuál es el orden de complejidad del proceso de rotar circularmente a la derecha? o(n)
- Si en la aplicación primero se hace el ordenamiento de los datos con el mejor algoritmo de ordenamiento que conoces, posteriormente se realiza un número aleatorio de rotaciones menor a la cantidad de datos, y finalmente se ejecuta tu función para encontrar ese número aleatorio, ¿cuál es el orden de complejidad temporal de toda la aplicación? Justifica tu respuesta.

n log n

PROBLEMA 2. *(35 puntos – Tiempo sugerido para solucionar el problema: 30 min).*

El algoritmo de Floyd es conocido en algunas referencias también con el nombre de **Floyd-Warshall**. Esto es debido a que la propuesta del algoritmo de Warshall, coincide en la aplicación que el propio algoritmo de Floyd realiza. De hecho el algoritmo de Floyd puede considerarse una generalización del algoritmo de Warshall. La propuesta original de Warshall fue proveer un algoritmo que procesara una matriz de adyacencias de un grafo NO PONDERADO para obtener su **cerradura transitiva**. La matriz de entrada en este caso es una **matriz de valores booleanos**, donde un valor de verdadero o 1 representa conexión entre los nodos, y un valor de falso o 0 indica que no hay conexión en el par de nodos correspondiente. La cerradura transitiva de esta matriz después de aplicar el algoritmo de Warshall es una matriz en donde un valor verdadero indica que SI es posible llegar del nodo origen al nodo destino, y el valor falso indica que no hay camino entre esos nodos.

El archivo `ex1ad15.cpp` contiene la implementación del algoritmo de Floyd. Modifica este algoritmo para convertirlo en el algoritmo de Warshall. Considera la mayor eficiencia en la implementación evitando hacer cálculos, pues en este caso no se requiere calcular el camino más corto.

Para probar tu implementación, utiliza el arreglo de datos llamado `datos3` que ya contiene el archivo y agrega instrucciones que desplieguen la información para responder a las siguientes preguntas:

- ¿Es posible llegar del primer nodo del grafo al último nodo del grafo? si
- ¿Cuántos nodos no son alcanzables desde el segundo nodo del grafo? 3
- ¿Desde qué nodos no se puede llegar al penúltimo nodo del grafo? 6
- ¿Cuál es el orden de complejidad del algoritmo de Warshall? n^3
- Dado que el algoritmo de Floyd también resuelve este problema de detectar si hay camino no entre dos nodos, ¿cuál de los dos algoritmos utilizarías en una aplicación y por qué?

PROBLEMA 3. *(30 puntos – Tiempo sugerido para solucionar el problema: 30 min).*

En el archivo `ex1ad15.cpp` podrás encontrar la implementación del algoritmo de **Godbole** para encontrar el número óptimo de multiplicaciones escalares en la multiplicación encadenada de una secuencia de 'n' matrices. Sin embargo, hace falta la implementación de la función `minimo` para que pueda utilizarse para obtener resultados. Implementa la función `minimo`, prueba el algoritmo con el arreglo llamado `datos4` que contiene las dimensiones de las matrices a multiplicar, y responde lo siguiente:

- ¿Cuál es el número óptimo de multiplicaciones escalares en la multiplicación de las matrices? 576
- ¿Cuáles son las dimensiones de la matriz resultante? 7x7
- ¿Cuál es el número óptimo de multiplicaciones escalares en la multiplicación de la secuencia de la segunda a la penúltima matriz? 828
- ¿Cuál es el número óptimo de multiplicaciones escalares en la multiplicación de 5 matrices de 6 X 6? 216
- ¿Cuál es el orden de complejidad del algoritmo que implementaste para la función `minimo`? n

Una vez que termines el examen y lo hayas subido a la página del curso, entra a expresatec.mty.itesm.mx y escribe cuál es tu opinión del examen y tu resultado esperado.

ALGORITMARIO

Algoritmo recursivo en pseudocódigo de la búsqueda binaria

```
function busca (inicio, fin: index) : index
if (inicio > fin) then return 0;
else
   mitad = (inicio + fin) div 2;
   if (x == arreglo[mitad]) then return mitad;
   else if (x < arreglo[mitad]) then
            return(busca(inicio, mitad-1));
        else return(busca(mitad+1,fin));
```

ANEXO – código de inicio para el examen

```
#include <cstdlib>
#include <iostream>

using namespace std;

const int N = 21;

int datos1[300] = {50193, 50668, 51264, …, 50117, 50135};
int datos2[300] = {98733, 1097, 1287, …, 98214, 98462};

int datos3[7][7] = {{0, 0, 0, 1, 0, 0, 0},
                    {0, 0, 0, 0, 1, 0, 0},
                    {0, 0, 0, 0, 0, 1, 0},
                    {0, 0, 0, 0, 0, 0, 1},
                    {0, 0, 1, 0, 0, 0, 0},
                    {0, 1, 0, 0, 0, 0, 0},
                    {1, 0, 0, 0, 0, 0, 0}};

int datos4[N] = {8, 19, 4, 16, 18, 3, 5, 19, 17, 3, 2, 7, 8, 9, 11, 6, 16, 12, 4, 15, 11};

// Problema 1
// Escribe aqui la implementación de las funciones del problema 1




// Problema 2
// Algoritmo de Floyd

int min(int a,int b)
{
 if(a<b)
  return(a);
 return(b);
}

void floyd(int p[7][7],int n)
{
 for(int k=0;k<n;k++)
  for(int i=0;i<n;i++)
   for(int j=0;j<n;j++)
     p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
}
```

```
// Problema 3
// Algoritmo de Godbole

int minimo(int i, int j, int D[N][N], int d[N])
{
	// implementación a resolver en el problema 3
}

int godbole (int d[N], int n)
{
	int D[N][N];
	// se considera que no se usará el subíndice 0 y que se utilizará el subíndice n en la matriz
	int j;

	for (int i = 1; i<= n; i++) D[i][i] = 0;
	for (int diag = 1; diag <= n-1; diag++)
	  for (int i = 1; i <= n-diag; i++)
		{
			j = i+diag;
			D[i][j] = minimo(i,j,D,d);
			// calcula el valor mínimo entre los diversos valores de:
			// D[i,k] + D[k+1, j] + d[i-1]*d[k]*d[j]  para k desde i hasta j-1
		}
	return D[1][n];
}

int main()
{
    //Escribe aqui las llamadas para probar y responder preguntas del examen

    system("pause");
}
```