Algoritmos y Estructuras de Datos

Clase de Práctica 15 - Matrices

Comisión A y F

Docente de práctica:

Tomás Assenza

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Ejercicios

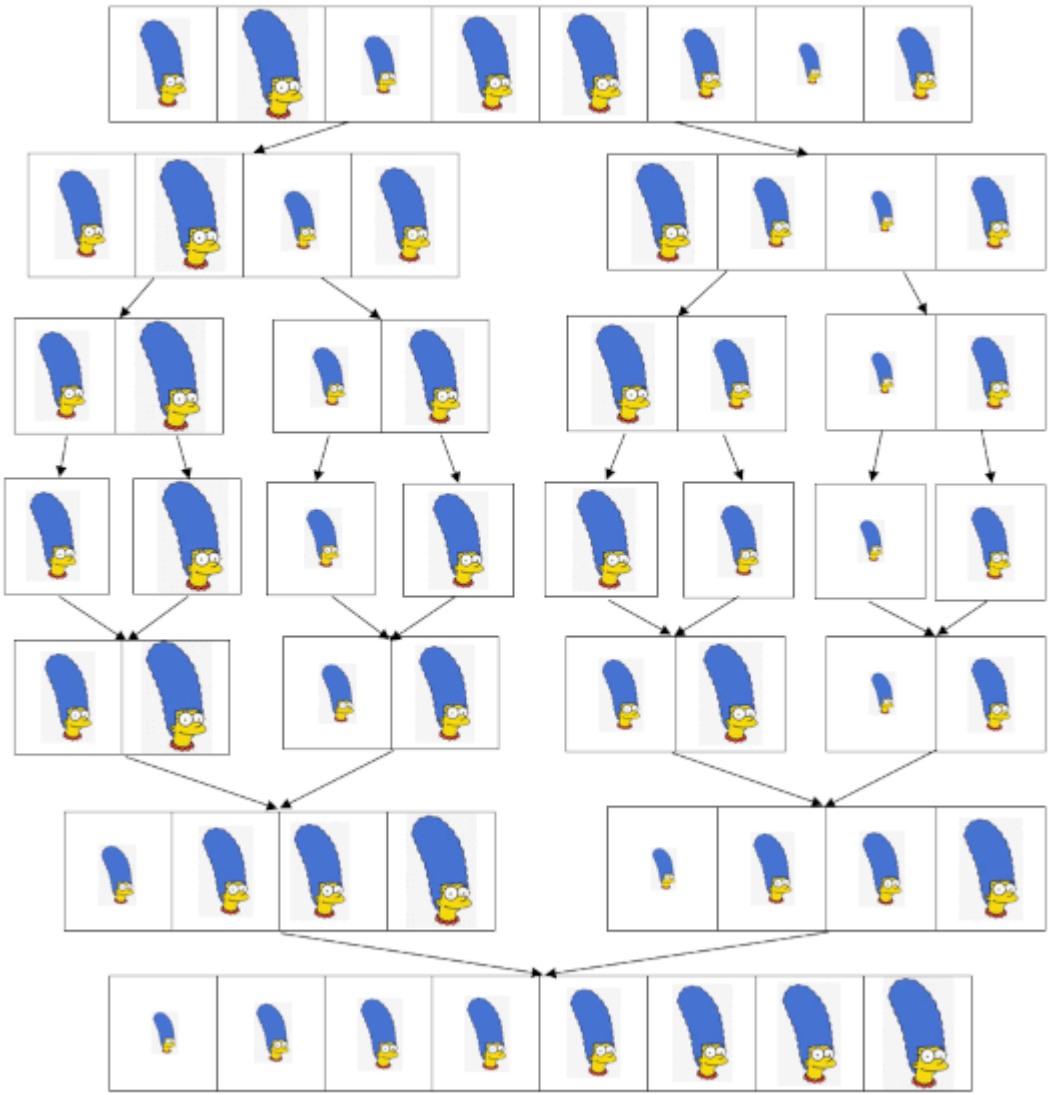
2) La función llamada elimOrdenar recibe como parámetros un arreglo A de 500 números enteros, el tamaño lógico TLA del arreglo y un valor entero X. El arreglo A está desordenado. La función deberá devolver el vector A actualizado de la siguiente forma:

- Se deben eliminar todos los valores que sean múltiplos de X.
- El vector debe quedar ordenado ascendentemente.
- La función deberá retornar la cantidad de elementos eliminados de A

Ejercicios

3) La función Uno() recibe dos arreglos de 500 números enteros (A y B), los tamaños lógicos de cada uno (TA y TB, TB <= TA) y un valor booleano (C). El arreglo A y el arreglo B están desordenados. La función Uno() debe retornar un vector V y su tamaño lógico TV, ordenado ascendentemente si C==True (descendentemente en otro caso) conteniendo solamente los valores de B que se encuentran en A, sin repeticiones. Los vectores A y B pueden contener elementos repetidos.

Marge Sort



```c
void mergesort(int v[], int inicio, int final) {
    if (final - inicio == 0)
        return;
    else {
        mergesort(v, inicio, (inicio + final) / 2);
        mergesort(v, (inicio + final) / 2 + 1, final);
        merge(v, inicio, (inicio + final) / 2, (inicio + final) / 2 + 1, final);
    }
}

void merge(int v[], int inicio1, int final1, int inicio2, int final2) {
    int i, j, k;
    int c[8];

    i = inicio1;
    j = inicio2;
    k = 0;

    while (i <= final1 && j <= final2) {
        if (v[i] < v[j])
            c[k++] = v[i++];
        else
            c[k++] = v[j++];
    }

    while (i <= final1)
        c[k++] = v[i++];

    while (j <= final2)
        c[k++] = v[j++];

    for (k = 0; k < final2 - inicio1 + 1; k++)
        v[inicio1 + k] = c[k];
}
```

Algunas operaciones básicas con matrices

Lectura de elementos

```
for (int i = 0; i < tamanioLogicoFilas; i++) {
    for (int j = 0; j < tamanioLogicoColumnas; j++) {
        cin >> matriz[i][j];
    }
}
```

Algunas operaciones básicas con matrices

Lectura de elementos

```
for (int i = 0; i < tamanioLogicoFilas; i++) {
    for (int j = 0; j < tamanioLogicoColumnas; j++) {
        cin >> matriz[i][j];
    }
}
```

Recorrer la diagonal principal

```
for (int i = 0; i < tamanioLogicoFilas; i++) {
    cout << matriz[i][i] << " ";
}
cout << endl;
```

Algunas operaciones básicas con matrices

Lectura de elementos

```
for (int i = 0; i < tamanioLogicoFilas; i++) {
    for (int j = 0; j < tamanioLogicoColumnas; j++) {
        cin >> matriz[i][j];
    }
}
```

Recorrer la diagonal principal

```
for (int i = 0; i < tamanioLogicoFilas; i++) {
    cout << matriz[i][i] << " ";
}
cout << endl;
```

Suma de la triangular inferior

```
int sumaTriangularInferior = 0;

for (int i = 0; i < tamanoLogicoFilas; i++) {
    for (int j = 0; j <= i; j++) {
        sumaTriangularInferior += matriz[i][j];
    }
}
```

Algunas operaciones básicas con matrices

Suma de la triangular superior

```
int sumaTriangularSuperior = 0;

for (int i = 0; i < tamanoLogicoFilas; i++) {
    for (int j = i; j < tamanoLogicoFilas; j++) {
        sumaTriangularSuperior += matriz[i][j];
    }
}
```

Ejercicios - Beecrowd

Sobre la Diagonal Principal - 1183

Lea un caracter en mayúscula que indica una operacion que será realizada sobre un arreglo bidimensional M[12][12]. Luego, calcule e imprima la suma o el promedio considerando solo los números que se encuentren por arriba de la diagonal principal del arreglo bidimensional, como se muestra en la siguiente figura (área verde).

Entrada

La primera línea de la entrada contiene un único caracter en mayúscula O ('S' o 'M'), que indica la operación Suma o Promedio (Média en portugués) a ser realizada sobre los elementos del arreglo bidimensional, seguido de 144 números de punto flotante que son los elementos del arreglo bidimensional.

Salida

Imprima el resultado del cálculo (de la suma o el promedio), con un solo dígito luego del punto decimal.

Ejercicios - OmegaUp

Matrices Simples - 10212

Este es un problema sencillo: dada una matriz de nxm enteros, tendrás que imprimir m enteros donde cada entero representa al menor de cada columna.

Entrada

En la primera línea los enteros n y m, que representan las dimensiones de la matriz. En la segunda línea recibirás nxm enteros: la matriz. Cada número de la matriz podrá tomar un valor entre -100000 y 100000, inclusive.

Salida

Imprime m enteros, donde cada uno corresponde a los menores de cada columna de la matriz.

Declaramos los métodos que vamos a utilizar, y los argumentos que le vamos a enviar

```
#include <iostream>
//Particularmente en Windows la declaracion de una matriz con tamaño grande puede
//fallar. Para probar este código se puede modificar TLFILAS y TLCOLUMNAS
#define TFFILAS 1000
#define TFCOLUMNAS 1000
using namespace std;

void leeMatriz(int [][TFCOLUMNAS], int, int);
void imprimeMenores(int [][TFCOLUMNAS], int, int);

int main(int argc, char *argv[]) {
  int n, m, matriz[TFFILAS][TFCOLUMNAS];

  cin >> n >> m;

  leeMatriz(matriz, n, m);
  imprimeMenores(matriz, n, m);

  return 0;
}
```

Codificamos las funciones

```
void leeMatriz(int matriz[][TFCOLUMNAS], int tlFilas, int tlColumnas) {
  for (int i = 0; i < tlFilas; i++) {
    for (int j = 0; j < tlColumnas; j++) {
      cin >> matriz[i][j];
    }
  }
}

void imprimeMenores(int matriz[][TFCOLUMNAS], int tlFilas, int tlColumnas) {
  for (int indiceColumna = 0; indiceColumna < tlColumnas; indiceColumna++) {
    int menorLocal = matriz[0][indiceColumna];
    for (int indiceFila = 0; indiceFila < tlFilas; indiceFila++) {
      if (matriz[indiceFila][indiceColumna] < menorLocal)
          menorLocal = matriz[indiceFila][indiceColumna];
    }
    cout << menorLocal << " ";
  }
  cout << endl;
}
```

Ejercicios - OmegaUp

DosFilasSolo2conElementosColumnasImparesImpares - 9505

Leer por teclado un valor N representando el tamaño de una matriz cuadrada. Determinar si existen en la matriz, dos filas (sólo 2) que tengan todos sus elementos impares. En caso de que sea así imprimir "SI", en otro caso imprimir "NO".

Entrada

La primer línea de entrada contendrá el valor N. Luego vendrán N líneas con los elementos separados por un espacio.

Salida

Si hay dos filas que tengan todos sus elementos impares imprimir "SI", en otro caso imprimir "NO"

Declaramos los métodos que vamos a utilizar, y los argumentos que le vamos a enviar

```
#include <iostream>
#define TLFILAS 10
#define TLCOLUMNAS 10
using namespace std;

void leeMatriz(int [][TLCOLUMNAS], int, int);
int columnas_impares(int matriz[][TLCOLUMNAS], int tlFilas, int tlColumnas);

int main(int argc, char *argv[]) {
  int matriz[TLFILAS][TLCOLUMNAS];
  int n;
  cin>>n;

  leeMatriz(matriz, n, n);
  if (columnas_impares(matriz, n, n) == 2)
    cout<<"SI";
  else
    cout<<"NO";

  return 0;
}
```

Codificamos las funciones

```
void leeMatriz(int matriz[][TLCOLUMNAS], int tlFilas, int tlColumnas) {
    for (int i = 0; i < tlFilas; i++) {
        for (int j = 0; j < tlColumnas; j++) {
            cin >> matriz[i][j];
        }
    }
}



int columnas_impares(int matriz[][TLCOLUMNAS], int tlFilas, int tlColumnas) {
    int cantfilas = 0, j = 0;
    for(int i = 0;i < tlFilas; i++){
        while(matriz[i][j] % 2 != 0 && j != tlColumnas){
            j++;
        }
        if(j==tlColumnas){
            cantfilas++;
        }
        j = 0;
    }
    return cantfilas;
}
```