Prácticas de Programación Resueltas: Semigrupos

1. *Realizar una subrutina que compruebe si una matriz de orden nxn, con n<100 define una ley de composición interna. Los datos de entrada de la subrutina son la matriz de entrada y la dimensión de la matriz n.*

Solución. La idea que se puede seguir para realizar esta subrutina es el siguiente:

Para cada $i, j \in \{1,...,n\}$ se estudia si $1 \leq a_{ij} \leq n$. Si existe un elemento que no cumple la condición, sabemos que la matriz no representa una ley de composición interna y si no existe tal elemento, sí lo es.

La subrutina que creamos devolverá un 0 si la matriz no representa una operación y un 1 si sí la representa. A modo de ejemplo, implementamos esta idea creando una subrutina en Lenguaje C:

```
int operacion(int n,int a[][100])
{
   int i,j,oper;
 /* la variable oper es una variable auxiliar que toma el valor 0 si la matriz a no define una
operación y 1 si sí la define. La inicializamos con el valor 1*/
   oper=1;
   for (i=1;i<=n;i++)
   {
      for (j=1;j<=n;j++)
      {
         if(a[i][j]>n||a[i][j]<1)
         {
            oper=0;
            i=n+10;
            j=n+10;
         }
      }
   }
  If(oper==0)
       printf("\n La matriz introducida no representa una operacion");
  else
       ("\n La matriz introducida representa una operacion");
   return(oper);
}
```

2. *Realizar una subrutina que dada una matriz de orden de orden nxn, con n<100 que define una ley de composición interna compruebe si ésta es conmutativa. Los datos de entrada de la subrutina son la matriz de entrada y la dimensión de la matriz n.*

Solución. El algoritmo que se puede seguir para realizar esta subrutina es el siguiente:

Para cada $i, j \in \{1,...,n\}$ se estudia si $a_{ij} = a_{ji}$. Si existe un elemento que no cumple la condición, sabemos que la matriz no representa una ley de composición interna conmutativa y si todos los elementos lo cumplen, sí lo es.

La subrutina que creamos devolverá un 0 si la matriz no representa una operación conmutativa y un 1 si sí la representa. A modo de ejemplo, implementamos esta idea creando una subrutina en Lenguaje C:

```
int asociativa(int dim, int a[][100])
{
 int i,j,k,conm;
/* conm  es una variable auxiliar que toma el valor 0 si la matriz a no define una operación
conmutativa y 1 si sí la define. La inicializamos con el valor 1*/
 conm=1;
 for(i=1;i<=dim;i++)
 {
    for(j=i+1;j<=dim;j++)
    {
       if(a[i][j]!=a[j] [i])
       {
          conm=0; /* Significa que no es conmutativa*/
          i=dim+10;
          j=dim+10;
        }
      }
    }
 if(conm==0)
   printf("\n La matriz no representa una ley de composicion interna conmutativa.");
else
   printf("\n La matriz representa una ley de composicion interna conmutativa.");
 return(conm);
}
```

3. *Realizar una subrutina que dada una matriz de orden de orden nxn, con n<100 que define una ley de composición interna compruebe si ésta es asociativa. Los datos de entrada de la subrutina son la matriz de entrada y la dimensión de la matriz n.*

Solución. La idea que se puede seguir para realizar esta subrutina es el siguiente:

Para cada $i, j \in \{1,...,n\}$ se estudia si $a_{ia_{jk}} = a_{a_{ij}k}$. Si existe un elemento que no cumple la condición, sabemos que la matriz no representa una ley de composición interna asociativa y si todos los elementos lo cumplen, sí lo es.

La subrutina que creamos devolverá un 0 si la matriz no representa una operación asociativa y un 1 si sí la representa. A modo de ejemplo, implementamos esta idea creando una subrutina en Lenguaje C:

```
int asociativa(int dim, int a[][100])
```

```
{
 int i,j,k,asoc;
/* asoc  es una variable auxiliar que toma el valor 0 si la matriz a no define una operación
asociativa y 1 si sí la define. La inicializamos con el valor 1*/
 asoc=1;
 for(i=1;i<=dim;i++)
 {
    for(j=1;j<=dim;j++)
    {
     for(k=1;k<=dim;k++)
     {
      if(a[i][a[j][k]]!=a[a[i][j]][k])
      {
         asoc=0; /* Significa que no es asociativa*/
         i=dim+10;
         j=dim+10;
         k=dim+10;
      }
     }
    }
 }
if(asoc==0)
  printf("\n La matriz no representa una ley de composicion interna asociativa.");
else
  printf("\n La matriz representa un semigrupo.");
 return(asoc);
}
```

4. *Realizar una subrutina que dada una matriz de orden de orden nxn, con n<100 que define una ley de composición interna localice los elementos idempotentes. Los datos de entrada de la subrutina son la matriz de entrada y la dimensión de la matriz n.*

Solución. La idea que se puede seguir para realizar esta subrutina es el siguiente:

Para cada $i, j \in \{1,...,n\}$ se estudia si $a_{ii} = i$. Si cumple la condición, entonces i es un elemento idempotente y en la matriz a colocaremos en la posición a_{0i} el valor 1. En otro caso, en a_{0i} guardaremos el valor 0. Asimismo, podemos realizar un contador que indique cuántos elementos idempotentes hay y sea lo que devuelva la subrutina. Observamos que al ejecutar esta subrutina se rellena la fila 0 de la matriz con los valores 1 y 0 que podrán ser utilizados en el programa principal.

A modo de ejemplo, implementamos esta idea creando una subrutina en Lenguaje C:

```
int idempotentes(int dim, int a[][100])
{
 int i,contador;
/* contador es una variable auxiliar que cuenta el número de elementos idempotentes. La
inicializamos con 0*/
 contador=0;
```

```
 for(i=1;i<=dim;i++)
{
   if (a[i][i]==i)
     {
        a[0][i]=1;
        contador=contador+1;
        printf("\n El elemento %d es un elemento idempotente",i);
     }
   else
     {
      a[0][i]=0;
     }
 }
return(contador);
}
```

5. *Realizar una subrutina que dada una matriz de orden de orden nxn, con n<100 que define una ley de composición interna y un elemento* i, *siendo* $1 \leq i \leq n$ *compruebe si éste es elemento neutro. Los datos de entrada de la subrutina son la matriz de entrada, la dimensión de la matriz n y el elemento* i.

Solución. La idea que se puede seguir para realizar esta subrutina es el siguiente:

Para cada $j \in \{1,...,n\}$ se estudia si $a_{ij} = j = a_{ji}$. Si cumple la condición para todo j, entonces i es un elemento identidad y la subrutina devolverá el valor 1. En otro caso, si encontramos un j para el que no lo cumple, sabemos que i no es un elemento identidad y no hace falta que sigamos comprobando los restantes elementos y devolverá la subrutina un 0.

A modo de ejemplo, implementamos esta idea creando una subrutina en Lenguaje C:

```
int esidentidad(int dim, int a[][100],int i)
{
 int j,identidad;
/* La variable identidad es una variable auxiliar que toma el valor 0 si no i no es elemento
identidad y 1 si sí lo es. Se inicializa con 1 y este valor cambia a 0 cuando se encuentra un j que
no cumple la condición*/
 identidad=1;
 for(j=1;j<=dim;j++)
 {
   if (a[i][j]!=j|| a[j][i]!=j)
     {
        identidad=0;
        printf("\n El elemento %d no es identidad. Falla para %d", i,j);
        j=dim+10;
      }
  }
If(identidad==1)
```

```
   printf("\n El elemento %d  es elemento identidad.", i);
return(identidad);
}
```