Instituto Politécnico Nacional

Programa de Licenciatura en Matemática Algorítmica

Trabajo Colaborativo

Primera Practica de Conjuntos

Autores:
Aguilar Romero Angel Uriel
Gordiano Rojas Carlos Vidal
Garcia Junquera Luis Eduardo
Martínez Huicochea Victor Hugo

Profesor:
Miguel Angel Valencia Bucio

Teoría de Conjuntos

México, Ciudad de México
October 6, 2023

Práctica 1

Objetivo:Realizar un programa que los parametros de entrada n= tamaño del conjunto A, R=conjunto R de pares ordenaados (R subconjunto de AxA) y que los datos de salida sea P=particion A inducida por R

1 Marco teorico

El programa proporcionado implementa un algoritmo de particionamiento de un conjunto A basado en relaciones binarias representadas en una matríz de adyacencia. El conjunto A esta representado por un número finito de elementos, y las relaciones binarias se ingresan como pares de enteros (a, b), donde a y b son elementos del conjunto A. El programa procesa estas relaciones y particiona el conjunto A en subconjuntos conexos.

2 Explicación del Programa

A continuación, se explican los pasos clave del programa:

- **Inicialización**: Se inicializan variables, como las variables a y b para leer las relaciones binarias, así como matrices y estructuras de datos para almacenar la matriz de adyacencia, el rastro de revisiones, y las particiones.
- **Entrada de Datos**: El usuario ingresa el tamaño del conjunto A (n).
- **Matriz de Adyacencia**: Se crea una matriz de adyacencia llamada "matriz" de tamaño n x n y se inicializa con ceros. Esta matriz se utiliza para representar las relaciones binarias entre elementos del conjunto A.
- **Ingreso de Relaciones Binarias**: El usuario ingresa las relaciones binarias hasta que ingrese (-1, -1) para indicar que no hay más relaciones que agregar. Las relaciones se almacenan en la matriz de adyacencia incrementando el valor en la posición correspondiente.
- **Matriz Simétrica**: Para evitar errores, se asegura que la matriz de adyacencia sea simétrica. Si hay una relación de a a b, también debe haber una relación de b a a en la matriz.
- **Impresión de Matriz**: Se imprime la matriz de adyacencia resultante para visualizar las relaciones binarias.
- **Rastreo de Visitas**: Se inicia un arreglo llamado "Revision" para rastrear si un elemento ha sido visitado en el proceso de particionamiento.
- **Particionamiento**: El programa inicia el proceso de particionamiento. Se utilizan dos estructuras de datos tipo pila (Elementos y Camino) para rastrear los elementos en la partición actual y los caminos explorados.
- **Recorrido de Grafos**: El programa explora el grafo a través de relaciones binarias. Comienza con un elemento y sigue los caminos posibles hasta que se agoten. A medida que se exploran los caminos, se marcan los elementos como visitados y se agregan a la partición actual.

- **Impresión de Particiones**: Después de explorar un conjunto conexo, se imprime la partición actual. Este proceso se repite hasta que se visitan todos los elementos del conjunto A.

- **Condición de Paro**: El programa verifica si todos los elementos han sido visitados. Si no, se establece la CondicionParo en 0, lo que indica que se debe continuar el proceso de particionamiento.

- **Finalización**: El programa termina y muestra un mensaje de finalización.

3 Pseudocodigo

1. Inicializar variables y matrices.

2. Solicitar al usuario el tamaño del conjunto A (n).

3. Crear una matriz de adyacencia llamada "matriz" de tamaño n x n e inicializarla con ceros.

4. Solicitar al usuario ingresar relaciones binarias hasta que ingrese (-1, -1) para terminar:

   (a) Leer a y b.

   (b) Si a y b son mayores que 0, incrementar el valor en matriz[a-1][b-1] en 1.

5. Hacer que la matriz sea simétrica para evitar errores.

6. Imprimir la matriz resultante.

7. Inicializar un arreglo "Revision" para rastrear si un elemento ha sido visitado en el proceso de particionamiento.

8. Inicializar índices y condiciones de paro.

9. Iniciar el proceso de particionamiento:

   (a) Establecer CondicionParo = 0.

   (b) Si el elemento actual (I) no ha sido visitado, agregarlo a la partición y al camino.

   (c) Recorrer los caminos posibles desde el elemento actual:

   (d) Si un elemento está conectado, no es el mismo y no ha sido visitado:

      i. Marcar el elemento como visitado.

ii. Agregarlo a la partición y al camino.

iii. Incrementar el tamaño de la partición.

iv. Cambiar la posición actual al nuevo elemento y reiniciar la búsqueda desde el inicio.

(e) Cuando se agotan los caminos, retroceder al camino anterior si es necesario.

(f) Imprimir la partición actual.

(g) Verificar si todos los elementos han sido visitados.

(h) Si no, establecer CondicionParo = 0 para continuar el proceso.

(i) Incrementar I para considerar el siguiente elemento.

10. Finalizar el programa.

4 Pruebas

A continuación mostramos algunas pruebas que realizamos al programa para poder verificar que su ejecución procede sin problemas, esto meadiante algunas relaciones únicas cuyas soluciónes puedan representar un problema para nuestro programa. Algunos de estos casos son:

- **Relacion vacia**: Sea una relacion en $n = 5$ vacia, entonces tenemos que:
  Datos de entrada:
  n=5
  $R = \emptyset$
  Partición Prevista:$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
  Solucion planteada del programa:

```
Ingrese el tama±o del conjunto A:
5
Ingrese las relacion #1 separadas por espacio(digite [-1 -1] para acabar):
-1 -1
a= -1, b=-1
El particionamiento de A es:
{1, },{2, },{3, },{4, },{5, },
 Fin del programa :D
--------------------------------
Process exited after 35.89 seconds with return value 0
Presione una tecla para continuar . . .
```

- **Relacion de un solo elemento**: Sea una relacion en $n = 1$ tal que:
  Datos de entrada:
  n=1
  $R = \{(1, 1)\}$
  Partición Prevista:$\{1\}$
  Solucion planteada del programa:

```
Ingrese el tama±o del conjunto A:
1
Ingrese las relacion #1 separadas por espacio(digite [-1 -1] para acabar):
1 1
a= 1, b=1
Ingrese las relacion #2 separadas por espacio(digite [-1 -1] para acabar):
-1 -1
a= -1, b=-1
El particionamiento de A es:
{1, },
 Fin del programa :D
--------------------------------
Process exited after 5.216 seconds with return value 0
Presione una tecla para continuar . . .
```

- **Relacion árbol**: Sea una relacion en $n = 8$ tal que:
  Datos de entrada:
  n=8
  $R = \{(1,2),(1,3),(1,4),(4,5),(4,6),(3,7),(3,8)\}$
  Partición Prevista:$\{1,2,3,4,5,6,7,8\}$
  Solucion planteada del programa:

```
Ingrese el tama±o del conjunto A:
8
Ingrese las relacion #1 separadas por espacio(digite [-1 -1] para acabar):
1 2
a= 1, b=2
Ingrese las relacion #2 separadas por espacio(digite [-1 -1] para acabar):
1 3
a= 1, b=3
Ingrese las relacion #3 separadas por espacio(digite [-1 -1] para acabar):
1 4
a= 1, b=4
Ingrese las relacion #4 separadas por espacio(digite [-1 -1] para acabar):
4 5
a= 4, b=5
Ingrese las relacion #5 separadas por espacio(digite [-1 -1] para acabar):
4 6
a= 4, b=6
Ingrese las relacion #6 separadas por espacio(digite [-1 -1] para acabar):
3 7
a= 3, b=7
Ingrese las relacion #7 separadas por espacio(digite [-1 -1] para acabar):
3 8
a= 3, b=8
Ingrese las relacion #8 separadas por espacio(digite [-1 -1] para acabar):
-1 -1
a= -1, b=-1
El particionamiento de A es:
{6, 5, 4, 8, 7, 3, 2, 1, },
 Fin del programa :D
```

Nota: Aunque no estan ordenados, es evidente que la solución dada concuerda con lo previsto

- **Relacion de todos los elementos**: Sea una relacion en $n = 3$ relacionada en su totalidad, entonces tenemos que:
  Datos de entrada:
  n=3
  $R = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1)(3,2),(3,3)\}$
  Partición Prevista:$\{1,2,3\}$
  Solucion planteada del programa:

```
Ingrese las relacion #3 separadas por espacio(digite [-1 -1] para acabar):
1 3
a= 1, b=3
Ingrese las relacion #4 separadas por espacio(digite [-1 -1] para acabar):
2 1
a= 2, b=1
Ingrese las relacion #5 separadas por espacio(digite [-1 -1] para acabar):
2 2
a= 2, b=2
Ingrese las relacion #6 separadas por espacio(digite [-1 -1] para acabar):
2 3
a= 2, b=3
Ingrese las relacion #7 separadas por espacio(digite [-1 -1] para acabar):
3 1
a= 3, b=1
Ingrese las relacion #8 separadas por espacio(digite [-1 -1] para acabar):
3 2
a= 3, b=2
Ingrese las relacion #9 separadas por espacio(digite [-1 -1] para acabar):
3 3
a= 3, b=3
Ingrese las relacion #10 separadas por espacio(digite [-1 -1] para acabar):
 1  1
a= -1, b=-1
El particionamiento de A es:
{3, 2, 1, },
 Fin del programa :D
--------------------------------
Process exited after 53.6 seconds with return value 0
Presione una tecla para continuar . . .
```

Nota: Aunque no estan oredenados, es evidente que la solución dada concuerda con el programa

- **Relacion Disconexa**: Sea una relacion en $n = 5$ tal que:
  Datos de entrada:
  n=5
  $R = \{(1,1),(1,2),(2,1),(1,5),(5,1),(5,2),(2,5),(3,3)\}$
  Partición Prevista:$\{1,2,5\},\{3\},\{4\}$
  Solucion planteada del programa:

```
Ingrese las relacion #2 separadas por espacio(digite [-1 -1] para acabar):
1 2
a= 1, b=2
Ingrese las relacion #3 separadas por espacio(digite [-1 -1] para acabar):
2 1
a= 2, b=1
Ingrese las relacion #4 separadas por espacio(digite [-1 -1] para acabar):
1 5
a= 1, b=5
Ingrese las relacion #5 separadas por espacio(digite [-1 -1] para acabar):
5 1
a= 5, b=1
Ingrese las relacion #6 separadas por espacio(digite [-1 -1] para acabar):
5 2
a= 5, b=2
Ingrese las relacion #7 separadas por espacio(digite [-1 -1] para acabar):
2 5
a= 2, b=5
Ingrese las relacion #8 separadas por espacio(digite [-1 -1] para acabar):
3 3
a= 3, b=3
Ingrese las relacion #9 separadas por espacio(digite [-1 -1] para acabar):
-1 -1
a= -1, b=-1
El particionamiento de A es:
{5, 2, 1, },{3, },{4, },
 Fin del programa :D
--------------------------------
Process exited after 40.99 seconds with return value 0
Presione una tecla para continuar . . .
```