Teoría de la computación - Práctica 4

Grupo Lunes 8:00-10:00 semanas B
— Práctica 4 —
Autor: Fabrizio Duarte
NIP:736857
Autor: Fernando Pastor Peralta
NIP:897113

Ejercicio 1:

Resumen:

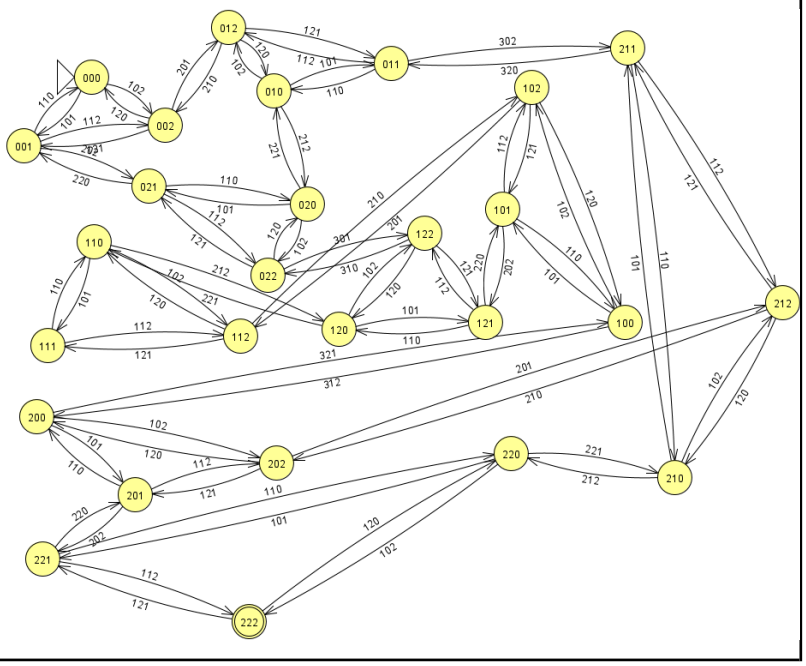
El problema de las Torres de Hanoi establece que tenemos una serie de discos (con mayor radio los discos de abajo y menor radio a medida que aumentan los discos) apilados en un palo y debemos mover esa pila de discos al palo final, en orden para poder ganar. Hay que seguir 2 reglas:

1. Mover un disco a la vez: Sólo se puede trasladar el disco superior de una de las pilas en cada movimiento.
2. Nunca colocar un disco más grande sobre uno más pequeño: Esto garantiza que el orden de los discos en cualquier varilla sea siempre de mayor a menor desde la base hasta la cima.

Nos centraremos en resolver el caso particular de 3 palos con 3 discos.

Para ello, crearemos un autómata en el que cada estado representa la posición de cada uno de los discos (en qué palo está cada uno), el número más característico refiriéndose a qué palo ocupa el disco 3 y el menos característico refiriéndose a qué palo ocupa el disco 1.
Los discos irán del 1 al 3, siendo el 1 el disco menos ancho y el 3 el más ancho, sin embargo, nos referiremos a los palos del 0 al 2, siendo el palo 0, el palo donde empezamos el juego.
Las transiciones se formarán de esta forma (disco_usado[1-3], posición_inicial_del_disco[0-2], posicion_final_del_disco[0-2]).

FOTO AUTÓMATA:



Ejercicios 2 y 3:

1.Resumen:

-Lenguaje:
El autómata anterior lo traduciremos al fichero de texto thP3D3.txt mediante esta codificación:

```
000 -> 001(101), 002(102);
001 -> 000(110), 002(112), 021(202);
002 -> 000(120), 001(121), 012(201);
010 -> 011(101), 012(102), 020(212);
011 -> 010(110), 012(112), 211(302);
012 -> 002(210), 010(120), 011(121);
020 -> 010(221), 021(101), 022(102);
021 -> 001(220), 020(110), 022(112);
022 -> 020(120), 021(121), 122(301);
100 -> 101(101), 102(102), 200(312);
101 -> 100(110), 102(112), 121(202);
102 -> 100(120), 101(121), 112(201);
110 -> 111(101), 112(102), 120(212);
111 -> 110(110), 112(112);
112 -> 102(210), 110(120), 111(121);
120 -> 110(221), 121(101), 122(102);
121 -> 101(220), 120(110), 122(112);
122 -> 022(310), 120(120), 121(121);
200 -> 100(321), 201(101), 202(102);
201 -> 200(110), 202(112), 221(202);
202 -> 200(120), 201(121), 212(201);
210 -> 211(101), 212(102), 220(212);
211 -> 011(320), 210(110), 212(112);
212 -> 202(210), 210(120), 211(121);
220 -> 210(221), 221(101), 222(102);
221 -> 201(220), 220(110), 222(112);
222 -> 220(120), 221(121);
```

Mediante Flex transformaremos la entrada en una serie de tokens que luego podrá interpretar Bison, creamos los siguientes tokens:

- \n (fín de línea): token EOL
- [0-9]+ (cualquier número): token NUMBER, además reserva memoria dinámica para guardar el número (formato texto).
- “->”: token FLECHA
- “(“: token OP
- “)”. token CP
- “,”: token COMA
- “;”: token PUNTOCOMA
- Si encontramos espacios o tabuladores los ignoraremos.
- Si encontramos un carácter diferente a los anteriores se realizará un syntax error.

-Gramática:
Bison recibirá estos tokens y los interpretará mediante las siguientes reglas:

- linea: se cumple si no encuentra nada o encuentra una línea completa que cumple la gramática, desde esta regla empezamos a analizar.
  *linea: /*nada*/ | linea estado FLECHA conjuntotransiciones PUNTOCOMA EOL*
- estado: es el primer número de cada línea, se asignará su valor a fila_tablaTransiciones.
  estado: NUMBER
- conjuntotransiciones: coge todos los estados a los que podemos ir con sus correspondientes transiciones.
  conjuntotransiciones: transicion | conjuntotransiciones COMA transicion
- transicion: toma un estado y su correspondiente transición (que está entre paréntesis), añadirá la transición a tablaTransiciones en la fila correspondiente al estado inicial y la columna correspondiente al estado al que llega siguiendo la transición.
  transicion: NUMBER OP NUMBER CP

-Codificación:
Nuestro programa puede usarse para abrir el archivo de texto que se quiera y elegir un nodo inicial y otro final que el usuario desee al ejecutar el programa mediante parámetros (no permitiremos el mismo nodo de inicio y fin). En cualquier caso, si no ponemos parámetros nuestro programa indicará cómo ponerlos y lanzará el comportamiento por defecto, en el que se abre el fichero thP3D3.txt con el nodo inicial 000 y el nodo final 222.

Para determinar la cantidad mínima de movimientos necesarios para ir entre estados, hemos utilizado un enfoque que consiste en elevar la matriz de transiciones a distintas potencias. Multiplicamos la matriz por sí misma repetidamente, obteniendo así la información que nos permite encontrar el camino más eficiente.

Después de encontrar el camino, imprimimos por pantalla el nodo inicial, el camino de transiciones tomado, el nodo final y la cantidad de estados y transiciones por las que hemos pasado.

2.Pruebas:

```
lab000:~/xddd/897113Pr4/ ./th
Uso: ./th <fichero_entrada> <nodo_inicial> <nodo_final>

Tomando comportamiento por defecto:
-----------------------------------
* Fichero: thP3D3.txt
* Nodo inicial: 000
* Nodo final: 222
-----------------------------------

Nodo inicial  : 000
Movimientos   : 102-201-121-302-110-212-102
Nodo final    : 222
Se han pasado por 8 estados (7 transiciones).
lab000:~/xddd/897113Pr4/ ./th thP3D3.txt 222 000

Parámetros tomados:
-----------------------------------
* Fichero: thP3D3.txt
* Nodo inicial: 222
* Nodo final: 000
-----------------------------------

Nodo inicial  : 222
Movimientos   : 120-221-101-320-112-210-120
Nodo final    : 000
Se han pasado por 8 estados (7 transiciones).
lab000:~/xddd/897113Pr4/ ./th thP3D3.txt 021 220

Parámetros tomados:
-----------------------------------
* Fichero: thP3D3.txt
* Nodo inicial: 021
* Nodo final: 220
-----------------------------------

Nodo inicial  : 021
Movimientos   : 110-221-101-302-110-212
Nodo final    : 220
Se han pasado por 7 estados (6 transiciones).
lab000:~/xddd/897113Pr4/ ./th thP3D3.txt 120 120

Parámetros tomados:
-----------------------------------
* Fichero: thP3D3.txt
* Nodo inicial: 120
* Nodo final: 120
-----------------------------------

No se permite mismo nodo inicial y final.
lab000:~/xddd/897113Pr4/ pwd
/home/a897113/xddd/897113Pr4
```