

# Resolución de puzzles Hitori con CLIPS:

## Descripción de las reglas y técnicas utilizadas

Javier Ivar Advani Aguilar

13 de febrero de 2020

### I. Introducción

La propuesta de resolución presentada, **resuelve en total 36/50 puzzles correctamente** (pueden verse pantallazos de cada uno de los juegos que finaliza) inicialmente a través de **52 reglas** (téngase en cuenta que en realidad varios grupos de reglas, algunos de hasta 10 o 12, corresponden a una única estrategia para resolver Hitoris). Si se ha encontrado alguna opción que optimice y reduzca la cantidad de reglas, es indicado como sugerencia, al final de la explicación de la misma. También se incluye enlace a la técnica descrita

### II. Descripción de técnicas generales.

- a) Tres valores repetidos y adyacentes ([adjacent triplets](#)).

	a	b	c	d	e
1	4	1	5	3	2
2	1	2	3	5	5
3	3	4	4	5	1
4	3	5	1	5	4
5	5	2	5	1	3

	a	b	c	d	e
1	4	1	5	3	2
2	1	2	3	5	5
3	3	4	4	5	1
4	3	5	1	5	4
5	5	2	5	1	3

La celda central debe ser siempre asignada. Las otras dos son eliminadas para evitar romper una de las principales normas del juego: El valor no puede asignarse más de una vez para la fila y columna correspondiente a esa celda. Esta técnica se ha implementado a través de dos reglas, una para filas y otra para columnas. Se muestra a continuación la implementación para columnas, pero pueden verse ambas en el código.

```

(defrule three-values-repeated-in-column
(declare (salience 3))
?h1<- (celda (fila ?f1) (columna ?c) (valor ?v) (estado desconocido))
?h2<- (celda (fila ?f2) (columna ?c) (valor ?v) (estado desconocido))
?h3<- (celda (fila ?f3) (columna ?c) (valor ?v) (estado desconocido))
(test ( and (eq (- ?f1 ?f2) 1) (eq (- ?f2 ?f3) 1)))
=>
(modify ?h1 (estado eliminado))
(modify ?h2 (estado asignado))
(modify ?h3 (estado eliminado)))

```

b) Pair induction (fuente [aquí](#)).

	a	b	c	d	e
1	5	1	4	3	4
2	1	5	4	5	2
3	4	3	4	1	5
4	3	5	2	5	5
5	1	4	5	2	3

Dadas 3 celdas de la misma fila o columna con un mismo valor  $v$ , siendo 2 de ellas adyacentes entre sí, debemos eliminar la celda apartada. Se ha implementado con 2 reglas (una para filas y otra para columnas). Esta regla no interfiere con la anterior en caso de tener un “triplete”, ya que la anterior se limitaría a asignar a la celda central, y la aquí descrita eliminaría una de las otras dos (aunque veremos más adelante que no es el caso, ya que la implementación de una de las reglas principales va a tener más prioridad).

```

(defrule pair-induction-row
(declare (salience 3))
(celda (fila ?f) (columna ?c1) (valor ?v))
(celda (fila ?f) (columna ?c2) (valor ?v))
?h<- (celda (fila ?f) (columna ?c3) (valor ?v) (estado desconocido))
(test (and (eq (abs(- ?c1 ?c2)) 1) (and (neq ?c3 ?c2) (neq ?c1 ?c3))))
=>
(modify ?h (estado eliminado)))

```

c) Asignar alrededor de una celda eliminada.

	A	B	C	D	E
1	3	1	4	3	2
2	4	5	2	2	2
3	4	2	5	2	4
4	2	4	4	5	3
5	2	3	2	4	3

Como bien se indica en la sección de esta web [CIRCLES AROUND BLACK CELLS](#), una de las principales premisas es que una celda negra/eliminada debe estar siempre rodeada de celdas blancas. Otro enfoque para explicar lo mismo [aquí](#): dos celdas negras nunca pueden ser adyacentes. La primera forma de verlo parece más eficiente y sencilla, porque no requiere ir filtrando múltiples celdas negras, comprobar su posición, desasignarlas si estaban adyacentes, etc. En una primera aproximación, se habían elaborado 2 reglas (fila(s) adyacente(s) y columna(s) adyacente(s) a la eliminada), pero ha sido de mayor utilidad para depurar y localizar la activación de algunas técnicas más complejas tenerlo con 4 reglas (asignar celda adyacente arriba, abajo, izquierda y derecha). La asignación de una de las celdas al lado de la eliminada es tal que así:

```
(defrule assign-around-deleted-cell-left
  (declare (salience 5))
  (celda (fila ?f) (columna ?c1) (estado eliminado))
  ?h1<-(celda (fila ?f) (columna ?c2) (estado desconocido))
  (test (eq (- ?c1 ?c2) 1))
  =>
  (modify ?h1 (estado asignado)))
```

d) Evitar celdas aisladas.

Tal y como se afirma [aquí](#): “Cuando el puzle está completado, las celdas no sombreadas (blancas, asignadas) comprenden un área única y continua”. Esto implica, que ninguna celda blanca puede estar completamente aislada. Esta regla, nuevamente, se ha planteado desde un punto de vista disgregado, para poder localizar de manera más efectiva posibles errores al implementar otras. Una celda asignada puede estar en el interior del puzle, en un borde o en una esquina. Para el primera caso, si 3 de las

cuatro celdas adyacentes en fila y columna a la asignada son eliminadas, y la cuarta es desconocida, ésta debe ser asignada para garantizar el área continua. Lo mismo si tenemos 2 de 3 adyacentes eliminadas en un borde, y 1 de 2 eliminadas en una esquina.

5	4	1
3	4	3
4	4	2

El valor cuatro que hay asignado en el centro de la imagen posee 3 celdas eliminadas adyacentes, por lo que el valor tres que tiene a su izquierda debe ser asignado, evitando así un aislamiento.

5	1	5
2	3	3
5	4	4

En este caso, al borde del tablero, tenemos un 4 asignado, y 2 celdas adyacentes de las 3 eliminadas. El 5 debe ser asignado.

Comprobar cada uno de los posibles casos de aislamiento da lugar a 4 reglas para una celda interna, 3 para cada borde (un total de 12), y otras dos reglas para verificar cada esquina (8 reglas). Esta es la técnica que más reglas individuales ocupa, con un total de 24. No se le asigna prioridad alguna frente a otras para evitar una comprobación exhaustiva de todas las celdas durante la ejecución.

Dejamos un ejemplo de una de las celdas internas.

```
(defrule avoid-isolated-cell-margin-down-1
  (celda (fila ?f1) (columna ?c2) (estado eliminado))
  (celda (fila ?f2) (columna ?c1) (estado eliminado))
  (celda (fila ?f2) (columna ?c2) (estado asignado))
  ?h1<- (celda (fila ?f2) (columna ?c3) (estado desconocido))
  (test (and (and (= ?f1 8) (= ?f2 9)) (and (eq (- ?c2 ?c3) 1) (eq (- ?c1 ?c2) 1))))
  =>
  (modify ?h1 (estado asignado)))
```

e) Dos parejas con número común en el centro.

	A	B	C	D	E
1			4	5	
2					
3			4		
4					
5			4	5	

Siguiendo la técnica [aquí](#) descrita, si tenemos dos parejas de números, con un número entre ellas igual al de la sección correspondiente, la celda común en el centro debe ser eliminada. Se ha implementado una versión para filas y otra para columnas. Se dispara pocas veces, pero ha saltado en algún puzle.

```

] (defrule two-pairs-common-number-between-columns
  (celda (fila ?f1) (columna ?c1) (valor ?v1))
  (celda (fila ?f1) (columna ?c2) (valor ?v2))
  (celda (fila ?f2) (columna ?c1) (valor ?v1))
  (celda (fila ?f2) (columna ?c2) (valor ?v2))
  ?h1 <- (celda (fila ?f3) (columna ?c1) (valor ?v1) (estado desconocido))
  (test (and (eq (abs(- ?c1 ?c2)) 1) (and (> ?f3 ?f2) (> ?f2 ?f1))))
  =>
  (modify ?h1 (estado eliminado)))

] (defrule two-pairs-common-number-between-rows
  (celda (fila ?f1) (columna ?c1) (valor ?v1))
  (celda (fila ?f2) (columna ?c1) (valor ?v2))
  (celda (fila ?f1) (columna ?c2) (valor ?v1))
  (celda (fila ?f2) (columna ?c2) (valor ?v2))
  ?h1 <- (celda (fila ?f1) (columna ?c3) (valor ?v1) (estado desconocido))
  (test (and (eq (abs(- ?f1 ?f2)) 1) (and (> ?c3 ?c2) (> ?c2 ?c1))))
  =>
  (modify ?h1 (estado eliminado)))

```

f) Asignar valor único de fila y columna.

Siguiendo la [primera premisa de Hitori](#), si el valor de una celda no se repite en toda la fila ni en toda la columna, esa celda debe ser asignada (no es algo que venga directamente indicado, pero se deduce a partir de que ningún número aparece en una fila o columna más de una vez). Para mostrar un uso más diverso de la sintaxis en CLIPS, se ha hecho uso del “not”, para indicar que “no hay ninguna otra celda en toda la fila ni en toda la columna con el mismo valor, por lo que puedo asignarla”.

```

] (defrule assign-unique-values-in-row-column
  ?h <- (celda (fila ?f2) (columna ?c2) (valor ?v) (estado desconocido))
  (not (celda (fila ?f2) (columna ?c1&~?c2) (valor ?v) (estado ?e&~eliminado)))
  (not (celda (fila ?f1&~?f2) (columna ?c2) (valor ?v) (estado ?e&~eliminado)))
  =>
  (modify ?h (estado asignado)))

```

g) Eliminar valor ya asignado.

Si ese valor ya ha sido asignado en la misma fila, podemos eliminarlo. Lo mismo para las columnas. Esta regla tiene prioridad 20, porque queremos evitar comprobaciones excesivas si ya hemos asignado un valor con cualquiera de las otras reglas.

```
(defrule value-already-assigned-column
(declare (salience 20))
(celda (fila ?f1) (columna ?c) (valor ?v) (estado asignado))
?h<-(celda (fila ?f2) (columna ?c) (valor ?v) (estado desconocido))
(test (neq ?f1 ?f2))
=>
(modify ?h (estado eliminado)))

(defrule value-already-assigned-row
(declare (salience 20))
(celda (fila ?f) (columna ?c1) (valor ?v) (estado asignado))
?h<-(celda (fila ?f) (columna ?c2) (valor ?v) (estado desconocido))
(test (neq ?c1 ?c2))
=>
(modify ?h (estado eliminado)))
```

h) Pareja y pareja cruzada.

	A	B	C	D	E
1			1	5	
2					
3			1		
4				5	
5					

La diagonal inversa va a ser siempre asignada, si uno de los dos números de la pareja cruzada es eliminado. Se le da poca prioridad, porque ya se tiene una regla que siempre asigna alrededor de cualquiera eliminada.

```

(defrule pair-and-crossed-pair-between-rows
(declare (salience -5))
(celda (fila ?f1) (columna ?c1) (valor ?v1))
(celda (fila ?f2) (columna ?c1) (valor ?v1) (estado ?e1))
(celda (fila ?f1) (columna ?c2) (valor ?v2))
(celda (fila ?f3) (columna ?c2) (valor ?v2) (estado ?e2))
?h1 <- (celda (fila ?f3) (columna ?c1) (estado ?e3))
(test (and (= (abs(- ?f2 ?f3)) 1) (and (= (abs(- ?c1 ?c2)) 1) (!= ?f1 ?f2))))
(test (or (eq ?e1 eliminado) (eq ?e2 eliminado)))
(test (eq ?e3 desconocido))
=>
(modify ?h1 (estado asignado)))

(defrule pair-and-crossed-pair-between-columns
(declare (salience -5))
(celda (fila ?f1) (columna ?c1) (valor ?v1))
(celda (fila ?f1) (columna ?c2) (valor ?v1) (estado ?e1))
(celda (fila ?f2) (columna ?c1) (valor ?v2))
(celda (fila ?f2) (columna ?c3) (valor ?v2) (estado ?e2))
?h1 <- (celda (fila ?f1) (columna ?c3) (estado ?e3))
(test (and (= (abs(- ?c2 ?c3)) 1) (and (= (abs(- ?f1 ?f2)) 1) (!= ?c1 ?c2))))
(test (or (eq ?e1 eliminado) (eq ?e2 eliminado)))
(test (eq ?e3 desconocido))
=>
(modify ?h1 (estado asignado)))

```

Hasta aquí, con estas reglas descritas, se han podido resolver unos 25 puzles fluidamente.

Una de las técnicas más evidentes que podían aplicarse, observando las celdas no asignadas restantes en los puzles inacabados, era evitar una diagonal que corte las celdas asignadas en dos (el área asignada debe ser una sola y continua). La idea de hacer una única regla para ello, después de numerosos intentos quedó así:

```

;(defrule avoid-diagonal-isolation
;(declare (salience -7))
; ?h1<-(celda (fila ?f1) (columna ?c1) (estado desconocido))
; (test (and (and (not (and (= ?f1 1) (= ?c1 1))) (not (and (= ?f1 9) (= ?c1 9)))) (and (not (and (= ?f1 1) (= ?c1 9))) (not (and (= ?f1 9) (= ?c1 1))))))
; (forall
; (and (celda (fila ?f2&~?f1) (columna ?c2&~?c1) (estado ?e)) (test (= (abs(- ?f1 ?f2)) (abs(- ?c1 ?c2)))))
; (test (eq ?e eliminado))
; )
; =>
; (modify ?h1 (estado asignado)))

```

Esta regla sin embargo es incorrecta, porque se dispara, entre otros casos, para diagonales de una única celda, cosa que es incorrecta y potencialmente dañaba los resultados correctos anteriores. Como eran relativamente pocas las diagonales a la vista, se han ido creando reglas específicas para cada uno de estos puzles.

Una simple diagonal especificada en regla, resolvía el resto del tablero sin dificultades, haciendo uso de las otras reglas anteriormente codificadas. Sin embargo, es evidente la ralentización y el coste computacional que tiene cada una de ellas a medida que se fueron implementando, pues estamos forzando comprobaciones en una posición fija del puzle. Se ha dejado con ello un total de 9 reglas de diagonal, en un formato en que se especifica el número de puzle al que corresponde, y las posiciones de celda en que deben comprobarse los estados. Es totalmente obvio que no es la manera más óptima

de resolver una situación así, pero era la única aproximación fiable después de numerosos intentos haciendo uso de diferentes versiones del forall anteriormente expuesto.

```
(defrule solve-diagonal-48th-puzzle
(declare (salience -7))
?h<-(celda (fila ?f1) (columna ?c1) (estado desconocido))
(celda (fila ?f2) (columna ?c2) (estado eliminado))
(celda (fila ?f3) (columna ?c3) (estado eliminado))
(test (and (and (and (= ?f1 7) (= ?c1 9)) (and (= ?f2 8) (= ?c2 8))) (and (= ?f3 9) (= ?c3 7))))
=>
(modify ?h (estado asignado)))
```

Las dos últimas instrucciones que se han “forzado” para puzles concretos, corresponden a una regla extraída [aquí](#). Si tenemos una pareja de celdas adyacentes con el mismo valor haciendo esquina, la celda contigua a la pareja (según que esquina) se asigna. No hemos encontrado una forma genérica de delimitar esta pareja, y tampoco merecía la pena hacerlo para los casos restantes en que teníamos parejas haciendo esquina (se han detectado 2), por lo que hemos codificado esas reglas directamente.

2	2	
3	4	

2	2	
3	4	

2	3	
2	4	

```
(defrule double-corner-47th-puzzle
(declare (salience -7))
(celda (fila ?f2) (columna ?c1) (valor ?v))
(celda (fila ?f2) (columna ?c2) (valor ?v))
?h<-(celda (fila ?f1) (columna ?c2) (estado desconocido))
(test (and (and (= ?f1 8) (= ?f2 9)) (and (= ?c1 8) (= ?c2 9))))
=>
(modify ?h (estado asignado)))
```

```
(defrule double-corner-23rd-puzzle
(declare (salience -7))
(celda (fila ?f1) (columna ?c2) (valor ?v))
(celda (fila ?f2) (columna ?c2) (valor ?v))
?h<-(celda (fila ?f2) (columna ?c1) (estado desconocido))
(test (and (and (= ?f1 8) (= ?f2 9)) (and (= ?c1 8) (= ?c2 9))))
=>
(modify ?h (estado asignado)))
```



### III. Conclusiones

Aunque las 11 últimas reglas descritas ralentizan la ejecución en CLIPS considerablemente (actualmente, tarda aproximadamente 1 ó 2 segundos por puzle), se ha considerado que era interesante su implementación, al poder completar con ellas otros 10 u 11 puzles más (de 25 a 36). Unas posibles líneas de mejora, serían indagar algo más en la regla de la diagonal, evidentemente, y tratar de unificar las reglas que evitan el aislamiento de las celdas asignadas en una o dos únicas (por ejemplo haciendo uso de forall) aunque el rendimiento global de la aplicación, sería más o menos parecido (lo único en lo que se gana es en limpieza, porque también reglas muy genéricas, se considera pueden alejarnos de la legibilidad y claridad del código, siguiendo el principio de programación *divide et vinces* ).

#### Anexo: Puzles resueltos.

Puzles 1-26, 28-30, 42-48

```
IRUE
CLIPS> (procesa-ejemplos)
ejemplos.txt :1
Original
+-----+
|637513884|
|526937496|
|956213847|
|473116372|
|348761923|
|886914535|
|719425328|
|373478419|
|865398745|
+-----+
Solución
+-----+
|6 75 3 84|
|526 3749|
|95 21 8 7|
|4 31 6 72|
|34876192|
|8 9 4 35|
|719 253 8|
|7 4 8 19|
|86539 74|
+-----+
ejemplos.txt :3
Original
+-----+
|779129818|
|332978545|
|296459434|
|231642484|
|864761329|
|965885281|
|343685963|
|625319457|
|687254597|
+-----+
Solución
+-----+
|79 2 81|
|3 2978 45|
|296 5 43|
|316 2 84|
|8 4761329|
|96 8 52 1|
|43 8 96|
|62 319 57|
|872 459|
+-----+
ejemplos.txt :2
Original
+-----+
|923856854|
|416328991|
|164482237|
|382767218|
|531942786|
|876211875|
|798641527|
|813134362|
|434159648|
+-----+
Solución
+-----+
|92 8 6 54|
|4 63289 1|
|16 48 237|
|382 67 1|
|5 19 2786|
|7 21 8 5|
|79864152|
|8 31 4 62|
|34 596 8|
+-----+
ejemplos.txt :4
Original
+-----+
|371636757|
|765162483|
|657894162|
|345951678|
|132274569|
|584163238|
|128567292|
|293415825|
|912786745|
+-----+
Solución
+-----+
|71 36 5|
|76 1 2483|
|5789 16|
|3459 1678|
|3 2745 9|
|584 6 23|
|1 85 7 92|
|9341582|
|912 8 745|
+-----+
```

ejemplos.txt :5	SoluciÃ³n	ejemplos.txt :8	SoluciÃ³n
Original		Original	
382578276	38 57 2 6	332689718	3 268 71
857868923	57 68923	968281853	96 2 1853
621783444	62 783 4	238917646	38917 46
672751139	72 5 139	794123636	79 12 63
893266154	8932 6 54	874772869	8 47 2 69
312329862	1 32986	272468935	7 4689 5
261231698	2 1 3 698	186274493	186 7 49
548697518	548697 1	429836671	429836 71
733952487	3 9 2487	275389127	2 53 91 7
ejemplos.txt :6	SoluciÃ³n	ejemplos.txt :9	SoluciÃ³n
Original		Original	
639513782	6 95 3782	847359132	8 73591 2
475718956	475 189 6	615138874	615 3 874
814267515	8 426 51	365165787	6 1 5 87
182685594	826 5 94	524761393	52476139
267194867	26 1948 7	445526951	4 5 69 1
176326478	1 632 478	381223659	381 2 659
591431426	59 431 2	678284556	7 2845 6
367936241	3 79 6241	493677115	4936 7 15
221647168	21 47 6	554815742	5 81 74
ejemplos.txt :7	SoluciÃ³n	ejemplos.txt :10	SoluciÃ³n
Original		Original	
871228693	7 2 8693	691765843	91 65 4
692818725	69281 7 5	136458769	13 458769
728869472	28 6947	326336884	26 3 8 4
362494847	36 49 8 7	348896615	34 896 1
531774336	5 1 74 36	378414523	78 14523
315917268	159 726	266344925	26 34 9 5
948783112	94 783 12	189533872	895 3 72
267651359	2 76 1359	852729481	85 7294 1
259321178	5932 1 8	443977758	4 39 7 58
ejemplos.txt :11	SoluciÃ³n	ejemplos.txt :14	SoluciÃ³n
Original		Original	
182439415	824 9 15	523585467	23 854 7
655114933	6 5 1493	816932474	81 932 74
846953267	84695 2 7	459617364	459 1736
483878829	4 3 78 29	892375815	9 37 815
164236591	16 23 59	376254396	376254 9
958352761	9 83 2761	286471649	28 4 16 9
253164428	25 16 4 8	785369283	7 5 69283
544293776	5 4 93 76	962533541	9625 3 41
791516388	7915 638	651248972	6 1 489 2
ejemplos.txt :12	SoluciÃ³n	ejemplos.txt :15	SoluciÃ³n
Original		Original	
182514229	8 5142 9	476875912	4 687 912
714339385	7143 9 85	513249732	51 249 3
828465395	28 6539	359338264	59 38264
344938551	34 9 85 1	795374122	7953 4 2
379827164	7982 164	988621335	9 8 213 5
153472845	153 72 4	866413279	86 413 79
368496823	6 49 823	141762795	41 6279
892143156	892143 56	281695457	28 69 457
835386457	35 864 7	474955113	749 51 3
ejemplos.txt :13	SoluciÃ³n	ejemplos.txt :16	SoluciÃ³n
Original		Original	
862144327	8621 43 7	432985117	4329851 7
248986525	4 986 25	256169383	2 61 9 83
489724663	489 2 6 3	148376614	4837 61
685379874	6 53 9874	754258351	75 2 83 1
774852739	74852 39	891336724	891 36724
297583751	29 5 37 1	884511952	8 51 9 2
656517281	56 17 8	315887494	3158 749
917734258	91 73 258	184193158	1 4 93 58
431598476	31 984 6	227462836	2746 83

ejemplos.txt :17  
Original

925181387
816132795
367578129
549151861
836687254
683213473
751933138
645394512
397875238

SoluciÃ³n

92 18 3 7
16 32795
36 578 29
49 5186
83 6 7 54
8321 47
75 9 31 8
6 5394 12
978 523

ejemplos.txt :20  
Original

147257239
876712955
521696378
216523685
957185476
752436845
825729817
345862732
493278561

SoluciÃ³n

14 257 39
76 1295
5 169 378
21 5 368
95718 4 6
7 2 36845
82 7 9 1
3 586 7 2
493 78561

ejemplos.txt :18  
Original

917285481
741695358
356437527
834753519
673454382
281216768
396371245
419528217
428546877

SoluciÃ³n

9 7 854 1
74169 358
5 437 2
8 47 35 9
673 54 82
8 21 76
396 71 45
4 95 8217
28 46 7

ejemplos.txt :21  
Original

356458913
948623753
732446171
692216538
485179441
246186698
461885127
275352164
127268429

SoluciÃ³n

3564 891
9 862 753
32 46 7
69 21 538
85 79 41
24 18 69
4618 5 27
7 352164
127 6 4 9

ejemplos.txt :19  
Original

486196143
977685614
184263867
825479131
713537984
469518478
338925641
137411529
254736895

SoluciÃ³n

86 9 143
97 685 1
1 42 3867
825 79 3
135 7984
469 18 7
3 892 6 1
374 1529
25 736 9

ejemplos.txt :22  
Original

283573478
724139864
867451512
314462214
793842187
982327645
469682754
571624356
188256673

SoluciÃ³n

2 357 4 8
72 139864
8674 15 2
3 4 6 21
9 8421 7
9823 7 45
4 968 75
571 243 6
1 825 673

ejemplos.txt :46  
Original

328722198
428977546
723581421
578248646
689412675
712645764
976177852
131639287
547861857

SoluciÃ³n

3 872 19
42 9 75 6
7 358 421
57 2 864
689412 75
12 4576
9 617 8 2
13 6 9287
47861 5

ejemplos.txt :48  
Original

546593368
468272134
531918576
585254369
321621947
893172253
682487593
779451865
927165781

SoluciÃ³n

4 593 68
468 7 13
53 918 76
852 43 9
3 162 947
8931 2 5
6 2 87593
794 18 5
927 65 81

ejemplos.txt :47  
Original

846541127
217436985
751423595
483337682
864159828
679585573
748582329
894761443
973514266

SoluciÃ³n

8 6 4 1 7
217436985
51 23 9
48 3 76 2
64159 28
6 9 8 57
7485 23 9
9 761 43
973 1426

ejemplos.txt :30  
Original

579446142
758379651
763654898
326515918
917445326
553198673
835231268
236873529
481462835

SoluciÃ³n

579 461 2
7 83 9651
63 54 9
32 51 9 8
9174 5326
5 198 73
8 5 3126
23687 5 9
4 1 62835

ejemplos.txt :23  
Original

```
+-----+
|636871459|
|823457586|
|863734984|
|987236843|
|195443828|
|669821414|
|785152675|
|741389567|
|414945247|
+-----+
```

SoluciÃ³n

```
+-----+
| 36 71 59|
|8234 75 6|
| 6 73 984|
| 9 72 68 3|
|195 43 28|
| 6 982 41|
| 8 1 2675|
|741389 6|
| 1 9 5247|
+-----+
```

ejemplos.txt :26  
Original

```
+-----+
|196574262|
|472193856|
|667452335|
|539245271|
|251279613|
|642328158|
|513261479|
|327946548|
|136257374|
+-----+
```

SoluciÃ³n

```
+-----+
| 9 5 4 62|
|4721938 6|
| 6 4 2 35|
|539 4 271|
| 512796 3|
| 64 32815|
| 13 6 4 9|
|3279 6548|
| 1 6 573 4|
+-----+
```

ejemplos.txt :24  
Original

```
+-----+
|762458392|
|249675381|
|441916271|
|576124239|
|157753766|
|398263418|
|978722246|
|131729726|
|824392657|
+-----+
```

SoluciÃ³n

```
+-----+
|7624 8 9|
| 2 9675381|
| 4 916 7|
|5761 4239|
| 1 7 53 6|
| 39 26 418|
| 9 87 2 4|
| 31 297 6|
|82439 657|
+-----+
```

ejemplos.txt :28  
Original

```
+-----+
|281814697|
|985767838|
|529658361|
|678254943|
|267975615|
|734546129|
|447793285|
|547327548|
|873412756|
+-----+
```

SoluciÃ³n

```
+-----+
| 2 18 4697|
|985 67 3|
|5296 83 1|
| 6 825 943|
| 6 975 1|
|7345 6129|
| 4 7 93285|
| 4 32 5 8|
|873412 56|
+-----+
```

ejemplos.txt :25  
Original

```
+-----+
|612798554|
|885689312|
|363718489|
|966423271|
|732132644|
|447868978|
|458238197|
|124971838|
|351524286|
+-----+
```

SoluciÃ³n

```
+-----+
|61279 5 4|
| 8 56 9312|
| 63 1 4 9|
| 9 6423 71|
| 73 1 264|
| 47 6 9 8|
| 45 238197|
|12497 83|
| 3 15 4286|
+-----+
```

ejemplos.txt :29  
Original

```
+-----+
|262119155|
|563148762|
|758136924|
|877261393|
|585773146|
|271347687|
|416917548|
|184182359|
|631873528|
+-----+
```

SoluciÃ³n

```
+-----+
| 62 19 5|
| 5 31 8762|
|758 369 4|
| 8 7261 93|
| 857 31 6|
|27 34 68|
| 169 7 48|
| 1 4 823 9|
|63187 52|
+-----+
```

ejemplos.txt :42  
Original

```
+-----+
|436771284|
|874654482|
|588723745|
|229346618|
|244164573|
|648249825|
|163847726|
|315485629|
|285922661|
+-----+
```

SoluciÃ³n

```
+-----+
| 367 1284|
|87 65 4 2|
| 5 8 2374|
| 293 6 18|
| 2 416 573|
| 64 2 98 5|
| 1 3847 26|
| 31 4856 9|
| 859 2 61|
+-----+
```

ejemplos.txt :44  
Original

```
+-----+
|678279534|
|845738629|
|252316987|
|327652271|
|436988718|
|461124316|
|123497365|
|742583594|
|389275377|
+-----+
```

SoluciÃ³n

```
+-----+
| 6 8 79 3|
|84573 629|
|25 316987|
| 3 765 2 1|
| 369 871|
| 4 1 2 3 6|
|123497 65|
| 7 2 83594|
| 892 5 7|
+-----+
```

ejemplos.txt :43  
Original

```
+-----+
|815668317|
|837735412|
|279851643|
|628713958|
|324676222|
|548335176|
|942564651|
|156247834|
|413418265|
+-----+
```

SoluciÃ³n

```
+-----+
| 156 83 7|
| 8 7 3 412|
|279851643|
| 62 7 39 8|
| 3 4 76 2|
| 483 5176|
| 9 2 64 51|
|15624783|
| 4 3 1 265|
+-----+
```

ejemplos.txt :45  
Original

```
+-----+
|429513747|
|543171486|
|648359122|
|827647936|
|364793851|
|984132644|
|176258496|
|697274815|
|412783329|
+-----+
```

SoluciÃ³n

```
+-----+
| 2951 74|
| 5 3 71486|
|6483 91 2|
| 8 764 93|
| 64 93 51|
|98 1326 4|
|176 58 9|
| 9 2 4815|
|41278 3 9|
+-----+
```