INTELIGENCIA ARTIFICIAL
Puzzle 3 x 3 Hill Climbing

El juego del puzzle se representa, sobre un tablero de 3x3 casillas. 8 de las casillas contienen una pieza o ficha que se puede deslizar a lo largo del tablero horizontal y verticalmente. Las fichas vienen marcadas con los números del 1 al 8, y el 0 representa a la casilla vacía, que permite los movimientos de las fichas. El objetivo del problema es partiendo de un tablero inicial con las fichas desordenadas alcanzar un tablero en el que todas las fichas estén ordenadas en orden creciente, dejando el hueco en la primer casilla del tablero. El número total de estados o tableros posibles que se pueden generar en el juego del puzzle es 9! = 362,880 estados.

Tu labor es resolver este problema empleando la idea de **Hill Climbing**.

El estado al que se desea llegar para cualquier puzzle dado será:

0	1	2
3	4	5
6	7	8

El criterio para elegir el siguiente estado a visitar será el siguiente:

Para cada estado que puedas generar partiendo del actual, siempre verás si lo puedes visitar o si es posible generarse, y de ser así lo guardarás temporalmente en un vector de estados.

Ya que tengas la lista de posibles estados a los que puedes ir partiendo del actual, deberás aplicarle a cada uno la siguiente función F (e: estado).

F (e) = La sumatoria del valor absoluto de la diferencia de los valores que se encuentran indicados por cada posición (i, j), (para 1 <= i, j <= 3) donde i son las filas y j las columnas, de las matrices del estado actual y el estado final.

$$\sum_{i=1}^{n=3} \sum_{j=1}^{n=3} \left| actual\ [i][j] - final\ [i][j] \right|$$

Por ejemplo si se da el estado

1	2	3
4	0	5
6	7	8

F (e) para ese estado es igual a:

F (e) = |a[1][1]-f[1][1]|+|a[1][2]-f[1][2]|+|a[1][3]-f[1][3]|+|a[2][1]-f[2][1]|+|a[2][2]-f[2][2]|+|a[2][3]-f[2][3]|+|a[3][1]-f[3][1]|+|a[3][2]-f[3][2]|+|a[3][3]-f[3][3]|, es decir.

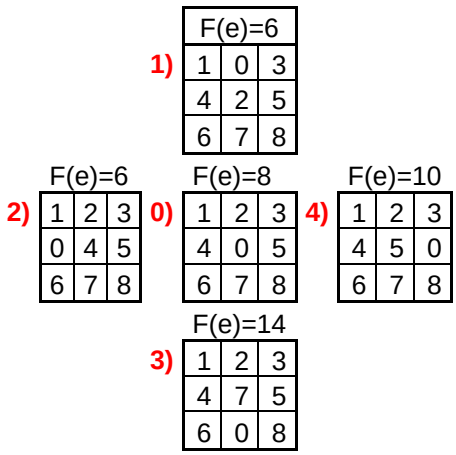
F (e) = |1-0|+|2-1|+|3-2|+|4-3|+|0-4|+|5-5|+|6-6|+|7-7|+|8-8| = 1+1+1+1+4+0+0+0+0 = 8.

En la pila debes insertar cada estado en el orden de F (e) de mayor a menor, es decir si tuvieras los valores 8 y 16, primero debes insertar en la pila el 16 y luego el 8, que al quedar en el tope de la pila, posteriormente sería tu estado actual.

En caso de que dos o más estados se puedan visitar tuvieran el mismo valor F (e), deberás insertar al final el estado con el menor orden lexicografico, quedando este en el tope y siendo posteriormente el estado actual.

El dibujo que se muestra abajo muestra un estado inicial **0** el cual deberás insertar en una pila, y dado que este estado no es el destino, deberás empezar a realizar los movimientos aplicando la idea de **Hill Climbing**.

Del estado **0** puedes partir hacia 4 estados. Para este caso el estado **3** donde F (e) = 14, es el primero que se ingresa en la pila, después el estado **4** donde F (e) = 10 se inserta en la pila, y en tercer lugar se inserta el estado **2**, ya que aunque hay empate en el valor de F (e) entre **1** y **2**, el **2** tiene un mayor orden lexicográfico, por lo que se inserta este, y finalmente se inserta el estado **1**.



Para observar mejor el orden lexicografico, puedes ver a las matrices como un vector de la siguiente manera:

1) 1 0 3 4 2 5 6 7 8
2) 1 2 3 0 4 5 6 7 8
3) 1 2 3 4 7 5 6 0 8
4) 1 2 3 4 5 0 6 7 8

Y si las ordenamos solo lexicograficamente quedarían así:

1) 1 0 3 4 2 5 6 7 8
2) 1 2 3 0 4 5 6 7 8
4) 1 2 3 4 5 0 6 7 8
3) 1 2 3 4 7 5 6 0 8

Y ordenándolas bajo ambos criterios (orden ascendente en cuento a F(e) y descendente lexicograficamente en caso de empates) quedarían así:

3) 1 2 3 4 7 5 6 0 8
4) 1 2 3 4 5 0 6 7 8

2) 1 2 3 0 4 5 6 7 8
1) 1 0 3 4 2 5 6 7 8

Una vez que te encuentres en el estado 1, dado que este no es el destino, nuevamente debes checar a que estados puedes ir (si es posible el movimiento y aun no ha sido visitado). El proceso se repite hasta que encuentres el estado final.

Recuerda que es búsqueda a lo profundo, y aunque en los ejemplos hay estados a los que se podía llegar, estos ya se encuentran en la pila, por lo que eventualmente podrías generar sus hijos.

Recuerda que para este caso todas las inserciones en una pila se hacen en el tope de la misma así como las extracciones.

Tu labor será mostrar en un archivo llamado puzzle3x3hill.txt todos los estados por los que pasa tu algoritmo antes de llegar a la solución, así como el numero de estos.

Tu programa fuente deberá llamarse puzzle3x3hill.(pas, for, c, cpp, java, etc).

Especificaciones de la entrada

La entrada es solo una matriz de tamaño 3x3 con 9 valores (entre el 0 y 8, sin repetirse cada uno), donde el 0 indica casilla vacía.

Todos los valores dados pueden estar separados por uno o más espacios (incluyendo saltos de línea o tabulaciones), es decir, la entrada podría estar de las siguientes maneras:

Ejemplo 1:
1 2 3 4 0 5 6 7 8

Ejemplo 2:
1 2 3 4 0 5 6 7 8

Ejemplo 3:
1 2 3
4 0 5
6 7 8

Especificaciones de la salida (puzzle3x3hill.txt)

El programa deberá generar un archivo denominado "puzzle3x3hill.txt" (sin las comillas), el cual contendrá todos los estados visitados por el algoritmo antes de llegar a la solución así como el número de los mismos:

Cada programa se probará con 10 casos de entrada (como los mostrados en los ejemplos de entrada 1, 2 y 3), cada salida correcta* vale 1 punto.

Fecha de entrega: **Miércoles 11 de noviembre de 2009**.

No se reciben programas fuera de esta fecha, más que para derecho a exámen, pero ya sin derecho a calificación en el programa.

Modo de entrega: enviar un correo con el código fuente puzzle3x3hill.(pas, for, c, cpp, java, etc), al mail **sergio10barca@gmail.com**.

Ejemplo de entrada 1

```
0 1 2
3 4 5
6 7 8
```

Ejemplo de salida 1 (puzzle3x3hill.txt)

```
0 1 2
3 4 5
6 7 8

1
```

Ejemplo de entrada 2

```
1 2 0
3 4 5
6 7 8
```

Ejemplo de salida 2 (puzzle3x3hill.txt)

```
1 2 0
3 4 5
6 7 8

1 0 2
3 4 5
6 7 8

0 1 2
3 4 5
6 7 8

3
```

Ejemplo de entrada 3

```
4 2 5
1 7 8
3 6 0
```

Ejemplo de salida 3 (puzzle3x3hill.txt)

```
4 2 5
1 7 8
3 6 0

4 2 5
1 7 0
3 6 8

4 2 0
1 7 5
3 6 8

4 0 2
1 7 5
3 6 8

0 4 2
1 7 5
3 6 8

1 4 2
0 7 5
3 6 8
```

```
1 4 2
3 7 5
0 6 8

1 4 2
3 7 5
6 0 8

1 4 2
3 0 5
6 7 8

1 0 2
3 4 5
6 7 8

0 1 2
3 4 5
6 7 8

11
```

* **NOTA:** Se considera salida correcta aquella que es igual en su totalidad a la salida que el evaluador genera, en caso de imprimir valores o caracteres demás, o los datos incompletos, todo el caso de prueba se considerará como incorrecto y tendrás 0 puntos para ese caso. Además para cada caso de prueba tu programa no debe tardar más de 1 minuto en arrojar la respuesta.

ANA LILIA C. LAUREANO CRUCES, SERGIO LUIS PÉREZ PÉREZ,
UAM AZCAPOTZALCO, 2009