13-10-2016

## Adrián Mora Perera

## **Gregorio Baldomero Patino Esteo**

## **Óscar Pérez Galán**

# GRUPO: G4\_07

## Memoria Sistemas inteligentes

# Práctica 1

Memoria Sistemas Inteligentes

# Explicación Práctica

1. La práctica consiste en la reconstruccion de un puzzle tipo slide del cual nos proporcionaran una imagen completamente desordenada para que nosotros tengamos que ordenarla, esto corresponderia a la práctica final pero la vamos a dividir en 2 partes:
   * 1ª: Con la imagen dada ( desordenada ) haremos una comparativa con respecto a la imagen original del puzzle para ver si efectivamente las piezzas del puzzle son correspondientes en ambas imágenes.
   * 1.2: Segunda parte sin uso de algoritmo en el que crearemos todo lo necesario para el manejo del arbol de busqueda que se desarrollara en el apartado 2
   * 2ª: Una vez realizada la comparación se procedera a la resolución del puzzle a traves de un algoritmo que concluya con la imagen del puzzle ordenada tal como se muestra en la imagen original.

# Resolución del apartado 1 de la práctica :

### Cuestiones planteadas por el grupo:

### **¿Qué lenguaje hemos escogido para la realización de la práctica y sus librerías?**

## Para nuestra práctica hemos considerado como lenguaje de desarrollo python , ya que nos parecia más sencillo a la hora de hacer pruebas dado que no es necesaro compilación y como añadido decidimos este lenguaje para mejorar nuestro conocimiento acerca de este lengauje.

## Como recursos principales para el tratamiento de imágenes usamos Tkinter y PIL

**Tipo de desarrollo:**

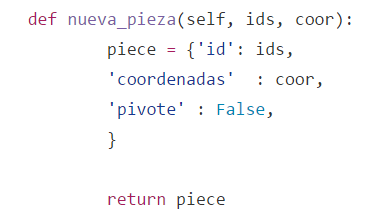
* Se ha elegido una metodología de desarrollo iterativa e incremental, por lo cual se irán haciendo revisiones del código en cada iteración para arreglar posibles fallos o decisiones equivocadas y añadir nuevas funcionalidades.

### **¿La imagen se partirá en un numero de piezas definido por el usuario?**

## Dado que no se especifica que el usuario tenga que definir el numero de piezas del puzzle y puesto que las imágenes proporcionadas para las pruebas (Aquellas que representan un estado inicial) vienen divididas ya en un determinado número de filas y columas, se ha optado por tener que poner esas filas x columnas en el codigo cada vez que se cambia de imagen.

## En posteriores versiones del programa se incluirá una interfaz con la que interactuar con el programa para seleccionar las imágenes que, una vez elegidas, dirá al usuario el número de filas y columnas recomendadas para dividir la imagen

**Detalles de la pieza:**



* Id sirve tanto como para saber qué número de pieza es, como para hacer de indice en una lista que contiene todas las imágenes, de forma que no haya que almacenar cada trozo de la imagen en una pieza, ahorrando así memoria.
* Coordenadas almacena las coordenadas x,y de la pieza en una matriz
* Pivote: determina qué pieza es la negra.

### **¿Las piezas de la imagen se guardaran en cada iteracción que trabaje con ellas?**

## Si, puesto que como cada pieza solo alberga un id, que sirve como indice en una lista de imágenes para indicar a qué trozo del puzzle hace referencia, y unas coordenadas (x,y) que representan su posición en una matriz, no ocupa mucho espacio en memoria ya que solo contiene números. Es por esto que en las siguientes tareas cuando se generen todos los posibles estados no se disparará el consumo de memoria, cosa que pasaría si se almacenace en cada pieza un trozo de imagen

### **¿Cómo definimos nuestro conjunto de operaciones básicas?**

## Se ha definido un método que recibe la pieza negra (Pivote) y un posible movimiento (arriba, abajo, izquierda y derecha) y determina si es un movimiento valido o no.

* Se ha definido un método que recibe dos piezas, el pivote y el destino, y las intercambia (sus coordenadas), para posteriormente repintarlas en la pantalla

### **¿Cómo planteamos la problemática de comparación de imágenes?**

## Esta comprobación la hacemos posible haciendo una división de la imagen original (ordenada) en piezas del mismo tamaño que las piezas de la imagen desordenada, para posteriormente ir comparando una a una cada pieza de ambas imágenes y ver que todas las piezas sean iguales en ambas imágenes.

# Resolución del apartado 1.2 de la práctica :

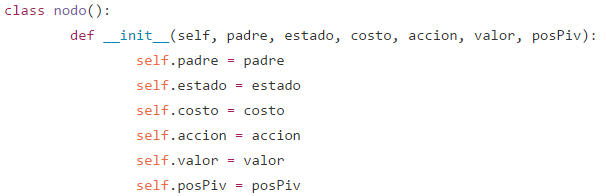
¿Cómo hemos definido el espacio de estados?

* Hemos considerado el espacio de estados como el conjunto de todos los posibles estados posibles que pueden ser generados dentro de nuestro problema.
* Un estado no es más que una representación del tablero en una iteración concreta, siendo nuestro estado inicial el tablero ordenado

## ¿Cómo hemos definido la frontera y sus correspondientes nodos?

* Consideramos la forntera junto con sus nodos como clases que deben interacuar.
* la clase forntera es una lista de nodos con un criterio de ordenacion según valor de los nodos

Estructura del nodo:



* El nodo es la estructura basica con la trabajamos y la que debemos evaluar para conseguir alcanzar el nodo objetivo.El numero de nodos estara limitado a una asignacion fija de memoria ,en este caso 1 gigabyte.Cuando la memoria este totalmente ocupada nos devolvera el numero de nodos maximos que puede almacenar

¿Cómo hemos considerado los nodos sucesores?

* Tras haber definido la frontera los nodos empezaran a ser insertados en ella y ordenándose conforme a un criterio(menor a mayor valor), Entonces escogeremos el nodo correspondiente para explorar sus posibles estados sucesores almacenándolos del siguiente modo:



¿Cómo controlamos el tiempo invertido en insertar un nodo en la frontera (Max, min y medio)?

* *Hemos colocado dos medidas de time para controlar del tiempo de inserción del nodo, seguidamente calculamos la diferencia, guardamos los máximos, mínimos y hacemos la media de tiempo*

