SISTEMAS INTELIGENTES
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante **Curso 2023/2024**

Práctica 1. Satisfacción de restricciones

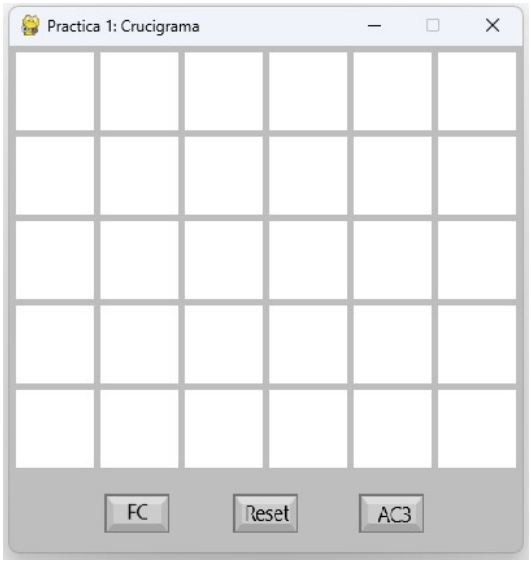
Objetivos:

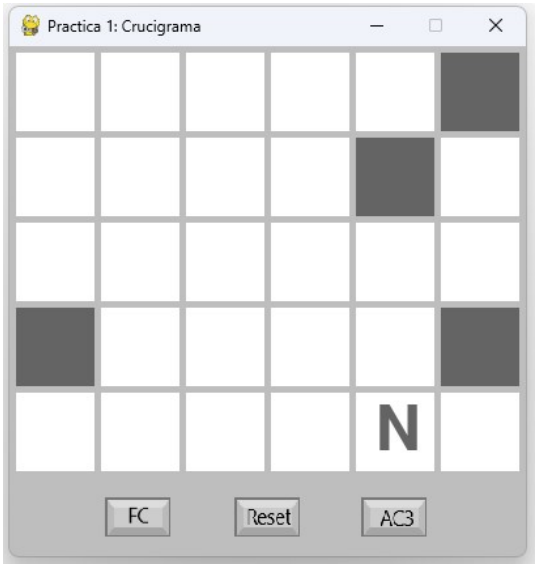
- Implementar dos algoritmos básicos en satisfacción de restricciones: AC3 y Forward Checking.
- Aplicar dichos algoritmos a un problema en concreto, en este caso la resolución de crucigramas.

Sesión 1: Introducción y entorno de trabajo.

En esta primera práctica de la asignatura se deben desarrollar algoritmos de búsqueda en problemas de satisfacción de restricciones para generar un crucigrama, es decir, encontrar las palabras que se ajustan a los huecos del crucigrama, tanto en horizontal como en vertical.

Se proporciona un entorno gráfico desarrollado en Python en el cual es posible establecer tanto las casillas ocupadas como caracteres concretos en cualquier casilla. Haciendo click con el botón izquierdo del ratón introduciremos una casilla ocupada (cuadrado relleno en negro) o bien dejaremos vacía una casilla que no lo estuviera. Con el botón derecho se nos presentará un dialogo para introducir un carácter en la casilla.





El tamaño del crucigrama se puede modificar cambiando los valores de las variables FILS y COLS

Cuando una palabra colocada en horizontal en el crucigrama se cruza con otra vertical, la posición en la que se cruzan debe contener el mismo carácter. Hay que tener en cuenta que en cada fila o columna del tablero puede haber más de una palabra dependiendo de las casillas negras que tenga el crucigrama

Funcionamiento

Al ejecutar el programa aparecerá una cuadrícula preparada para marcar casillas con caracteres o rellenarlas de negro. En la parte inferior aparecen tres botones que cuando se pulsa en ellos se ocupan de llamar a los s algoritmos que hay que implementar.

Para poder generar un crucigrama es necesario disponer de una lista de palabras que se usarán para rellenar las celdas. Estas palabras estarán en un fichero de texto que se cargará al iniciar el entorno.

Botones:

- FC: aplicar el algoritmo Forward Checking. Este algoritmo puede aplicarse tanto para una cuadrícula con su configuración inicial como tras la ejecución de AC3. En este último caso los dominios de las variables al comenzar a ejecutar FC serán los devueltos por AC3.
- AC3: aplicar el algoritmo AC3. Este algoritmo permite reducir los dominios de las variables y elimina inconsistencias de arista.
- Reset: deja el tablero en su estado inicial con todas las casillas vacías

Script

El entorno se ha desarrollado en Python usando la librería pygame. Pygame permite la creación de videojuegos en dos dimensiones, permite programar la parte multimedia (gráficos, sonido y manejo de eventos) de forma sencilla.

El entorno proporcionado está compuesto por 3 ficheros:

- main: es el fichero que hay que ejecutar para lanzar el entorno. Tiene el bucle principal de manejo, así como el desarrollo del interfaz gráfico
- tablero: permite construir un objeto tablero que contiene las dimensiones del mismo, así como la matriz que representa las celdas del crucigrama.

- dominio: permite crear una estructura adecuada para guardar las palabras del fichero que se usan como diccionario. Un dominio está compuesto por dos campos: tamaño y lista de palabras de dicho tamaño que están presentes en el fichero de texto.

Cuando se lanza el entorno, el módulo creaAlmacen() se ocupa de abrir el fichero de texto y crear el almacen de palabras, es decir, una lista de objetos de tipo Dominio. De esa forma las palabras están organizadas por su tamaño y es más práctico para poder aplicar los algoritmos. Se puede probar con distintos ficheros cambiando para crear el almacen de palabras simplemente abriendo otro fichero en este módulo.

Ejecución del proyecto

Ejecutar el fichero main.py

Tarea a realizar en esta primera sesión:

- Prueba y analiza el entorno. Averigua cómo acceder a una celda del tablero
- Crea una clase Variable con los campos que consideres adecuados.

Sesiones 2, 3, 4 y 5: Implementación del algoritmo Forward Checking. Hito intermedio de entrega

El algoritmo Forward Checking asigna valores a las variables que componen el problema. Cada vez que asigna un valor a una variable (variable actual), comprueba esa asignación con las variables que todavía están sin instanciar (variables futuras) que están restringidas con la variable actual. Los valores de las variables futuras que son inconsistentes con esa asignación son eliminados temporalmente de los dominios de las variables. Si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con otro valor.

1. Seleccionar x_i.
2. Instanciar $x_i \leftarrow a_i : a_i \in D_i$.
3. Razonar hacia adelante (forward-check):
   - Eliminar de los dominios de las variables $(x_{i+1}, \ldots, x_n)$ aún no instanciadas, aquellos valores inconsistentes con respecto a la instanciación (x_i, a_i), de acuerdo al conjunto de restricciones.
4. Si quedan valores posibles en los dominios de todas las variables por instanciar, entonces:
   - Si $i < n$, incrementar i, e ir al paso (1).
   - Si $i = n$, salir con la solución.
5. Si existe una variable por instanciar, sin valores posibles en su dominio, entonces retractar los efectos de la asignación $x_i \leftarrow a_i$. Hacer:
   - Si quedan valores por intentar en D_i, ir al paso (2).
   - Si no quedan valores:
     - Si $i > 1$, decrementar i y volver al paso (2).
     - Si $i = 1$, salir sin solución.

```
funcion FC(i variable): booleano
    para cada a∈ factibles[i] hacer
        Xi ← a
        si i=N solución retorna CIERTO
        sino
                si forward (i,a)
                        si FC(i+1) retorna CIERTO
                restaura (i)
    retorna FALSO
funcion forward(i variable, a valor): booleano
    para toda j=i+1 hasta N hacer
        Vacio ← CIERTO
        para cada b ∈ factibles[j] hacer
                si (a,b) ∈ Rij vacio ←FALSO
                sino eliminar b de factible[j]
                        Añadir b a podado[j]
        si vacio retorna FALSO
    retorna CIERTO
procedimiento restaura(i variable)
    para toda j=i+1 hasta N hacer
        para todo b ∈ podado[j] hacer
                si Xi responsable filtrado b
                        Eliminar b de podado[j]
                        Añadir b a factible[j]
```

Tareas a realizar en estas sesiones:

- Implementar el algoritmo Forward checking.
- Antes de la sesión 4 se debe entregar el proyecto

Sesiones 6 y 7: Implementación el algoritmo AC3

En estas sesiones se va a implementar el algoritmo AC3. Este algoritmo transforma un problema en otro sin inconsistencias de arco

$Q = \{c(e_p) = \langle V_i, V_j \rangle | e_p \in E, i \neq j\}$
Mientras $Q \neq \emptyset$ hacer
    $\langle V_k, V_m \rangle$ = seleccionar_y_borrar(Q)
    cambio = falso
    Para todo $v_k \in D_k$ hacer
        Si no_consistente (v_k, D_m) entonces
            borrar (v_k, D_k)
            cambio = cierto
        FinSi
    FinPara
    Si Dk = $\emptyset$ entonces salir_sin_solución FinSi
    Si cambio = cierto entonces
        $Q = Q \cup \{c(e_r) = \langle V_i, V_k \rangle \mid e_r \in E, i \neq k, i \neq m\}$
    FinSi
FinMientras

Cuando se pulse en el botón del AC3 se debe mostrar por pantalla los dominios de cada variable antes y después de ejecutar el algoritmo.

Ejemplo:

```
DOMINIOS ANTES DEL AC3
Nombre 0 Posición 0 0 Tipo: horizontal Dominio:  ['TOTEM', 'OSERA', 'SETOS', 'RETOS', 'ESOPO']
Nombre 1 Posición 1 0 Tipo: horizontal Dominio:  ['ESTO', 'PARA', 'COMO', 'ROSA', 'OLOR', 'LALA', 'PERO', 'OSOS', 'PERA']
Nombre 2 Posición 1 5 Tipo: horizontal Dominio:  ['L', 'A', 'B']
Nombre 3 Posición 2 0 Tipo: horizontal Dominio:  ['PRUEBA', 'VARIAS', 'LOTERO', 'LOTERA', 'ROMANO', 'ROMANA']
Nombre 4 Posición 3 1 Tipo: horizontal Dominio:  ['ESTO', 'PARA', 'COMO', 'ROSA', 'OLOR', 'LALA', 'PERO', 'OSOS', 'PERA']
Nombre 5 Posición 4 0 Tipo: horizontal Dominio:  ['ROMANO', 'ROMANA']
Nombre 6 Posición 0 0 Tipo: vertical Dominio:  ['CON', 'LAL', 'ROL', 'OLA', 'SOL', 'ARA', 'RON']
Nombre 7 Posición 0 1 Tipo: vertical Dominio:  ['TOTEM', 'OSERA', 'SETOS', 'RETOS', 'ESOPO']
Nombre 8 Posición 0 2 Tipo: vertical Dominio:  ['TOTEM', 'OSERA', 'SETOS', 'RETOS', 'ESOPO']
Nombre 9 Posición 0 3 Tipo: vertical Dominio:  ['TOTEM', 'OSERA', 'SETOS', 'RETOS', 'ESOPO']
Nombre 10 Posición 2 4 Tipo: vertical Dominio:  ['CON', 'LAL', 'ROL', 'OLA', 'SOL', 'ARA', 'RON']
Nombre 11 Posición 1 5 Tipo: vertical Dominio:  ['ES', 'UN', 'DE', 'LA', 'NO', 'AR']
DOMINIOS DESPUES DEL AC3
Nombre 0 Posición 0 0 Tipo: horizontal Dominio:  ['OSERA', 'SETOS', 'RETOS']
Nombre 1 Posición 1 0 Tipo: horizontal Dominio:  ['OSOS']
Nombre 2 Posición 1 5 Tipo: horizontal Dominio:  ['L', 'A']
Nombre 3 Posición 2 0 Tipo: horizontal Dominio:  ['LOTERA']
Nombre 4 Posición 3 1 Tipo: horizontal Dominio:  ['PERO']
Nombre 5 Posición 4 0 Tipo: horizontal Dominio:  ['ROMANO', 'ROMANA']
Nombre 6 Posición 0 0 Tipo: vertical Dominio:  ['ROL', 'SOL']
Nombre 7 Posición 0 1 Tipo: vertical Dominio:  ['ESOPO']
Nombre 8 Posición 0 2 Tipo: vertical Dominio:  ['TOTEM']
Nombre 9 Posición 0 3 Tipo: vertical Dominio:  ['OSERA']
Nombre 10 Posición 2 4 Tipo: vertical Dominio:  ['RON']
Nombre 11 Posición 1 5 Tipo: vertical Dominio:  ['LA']
```

Tareas a realizar en estas sesiones:

- Implementar el algoritmo AC3

Sesión 8: Documentación y pruebas. Hito final

Esta última sesión está dedicada a realizar pruebas y terminar la documentación que se deberá haber ido elaborando de manera continua durante todo el desarrollo de la práctica.

La documentación es la parte más importante de la práctica (60%). **Como mínimo** debe contener:

- Explicación detallada de los algoritmos implementados
- Especificar un problema pequeño (5 variables máximo) y diccionario de palabras pequeño
- Dibujar el grafo de restricciones del problema especificado.
- Traza del problema pequeño de FC.
- Traza del problema pequeño de AC3
- Sección de experimentación en la que se describan las diferentes pruebas realizadas a los algoritmos, dejando bien claro el objetivo de las pruebas. Debe haber pruebas con distintos tamaños de crucigramas y distintas casuísticas: sin solución ya determinado por el AC3, AC3 deja valores en el dominio, pero el problema no tiene solución, y con solución
- Estudio de tiempos de ambos algoritmos para distintos tamaños de crucigramas y distintos diccionarios.
- Se valorará el empleo de gráficas para el análisis de tiempos.

Entrega de la práctica

La fecha límite de entrega de la práctica es el 5 **de noviembre de 2023** a las 23:55h. La entrega se realizará a través de Moodle.

Formato de entrega del proyecto para el hito final (entrega final)

La entrega debe consistir en un **fichero comprimido zip** con dos carpetas:

- /Fuente: todos los ficheros .py que constituyen la práctica

- /Doc: documentación en **pdf**

¡¡¡AVISO IMPORTANTE!!!

No cumplir cualquiera de las normas de formato/entrega puede suponer un suspenso en la práctica.

Recordad que **las prácticas son INDIVIDUALES** y NO se pueden hacer en parejas o grupos.

Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia y, como indica el Reglamento para la Evaluación de Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015) y el documento de Actuación ante copia en pruebas de evaluación de la EPS, **se informará a la dirección de la Escuela Politécnica Superior para la toma de medidas oportunas**.

Plan de entrega por hitos

Durante el periodo de ejecución de la práctica se realizarán dos hitos de entrega. Es obligatorio cumplir las fechas de las entregas correspondientes:

Hito	Entrega	Fecha tope
1 (intermedio)	Todos los ficheros .py (no es necesario entregar documentación)	15 de octubre
2 (final)	Práctica completa siguiendo la estructura del formato de entrega	5 de noviembre

- La no entrega del hito 1 en la fecha prevista supone una penalización del 20%.

La nota de la práctica sufrirá una penalización de dos puntos si no se cumple rigurosamente con los requisitos de la entrega