

# KNIGHT'S TOUR

## PIA INTELIGENCIA ARTIFICIAL

ENRIQUE ANDRÉS CHÁVEZ MARTÍNEZ 1810004

PEDRO RICARDO PARDO GAYTAN 1884124

JORDAN EDUARDO GARCÍA PECINA 1719825

# KNIGHT'S TOUR

- EN EL AJEDREZ SABEMOS QUE UN CABALLERO PUEDE SALTAR DE UNA MANERA PECULIAR. PUEDE MOVERSE YA SEA DOS CUADROS HORIZONTALMENTE Y UN CUADRADO VERTICALMENTE, O PUEDE MOVERSE 2 CUADROS VERTICALMENTE Y UNO HORIZONTALMENTE, EL MOVIENTO EN SI SE VE COMO UNA "L".
- EN ESTE PROBLEMA TENEMOS UN TABLERO DE AJEDREZ CASI VACÍO, EL CABALLERO EMPIEZA DESDE CAULQUIER ESPACIO EN EL TABLERO, Y NUESTRO OBJETIVOS ES HACER QUE EL CABALLERO PUEDA VISITAR TODOS LOS ESPACIOS DEL TABLERO TENIENDO EN CUENTA QUE NO PUEDES VISITAR POR SEGUNDA VEZ UN MISMO ESPACIO.
- LOS TABLEROS PUEDEN SER DE  $N \times N$ , CON  $N \geq 5$ , DE LO CONTRARIO EL CABALLERO ES INCAPAZ DE COMPLETAR EL PROBLEMA.



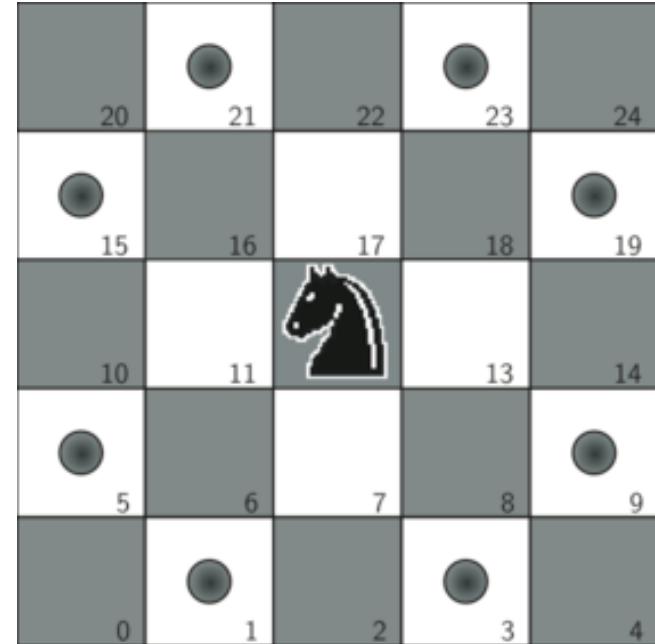
# KNIGHT'S TOUR

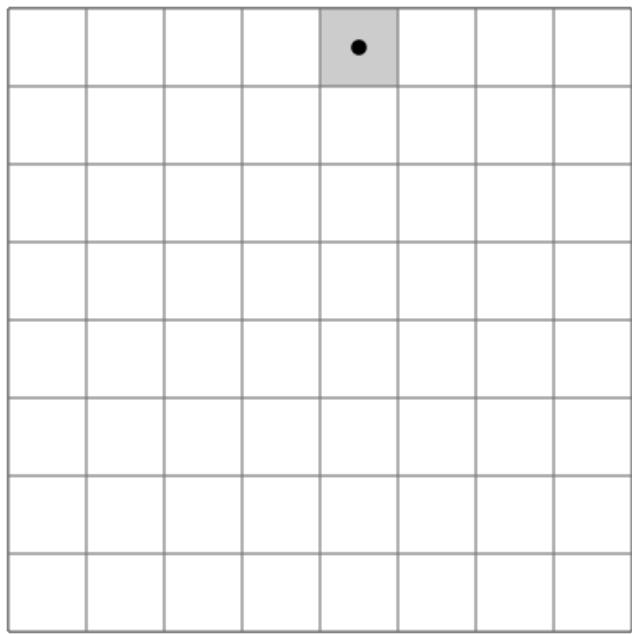
## PARÁMETROS

PARA NUESTRO PROBLEMA SE NECESITA UNA PIEZA DE CABALLO DE AJEDREZ Y UN TABLERO CON UNA CANTIDAD DE CASILLAS INDEFINIDA.

## REGLAS

EL PROBLEMA TIENE REGLAS MUY SIMPLES, PRIMERO COMENZAREMOS CON EL AGENTE O MEJOR CONOCIDO COMO CABALLO EL CUAL PODRÁ MOVERSE SIEMPRE Y CUANDO HAYA SUFICIENTE ESPACIO; EL ESPACIO NECESARIO Y LA REGLA BÁSICA DEL AGENTE ES QUE SOLO PUEDE MOVERSE DOS CASILLAS HACIA EL FRENTE DESDE CUALQUIERA DE LOS 4 PUNTOS DE LA CASILLA ACTUAL Y DESPUÉS SOLO PUEDE MOVERSE UNA CASILLA DE LADO CUALQUIERA MIENTRAS NO SEA HACIA EL FRENTE O DE REGRESO, EXPLICADO DE MANERA MÁS BÁSICA ESTA PIEZA SOLO PUEDE MOVERSE DE MANERA SIMILAR A UNA “L”. EL CABALLERO NO PODRÁ VISITAR UNA CASILLA PREVIAMENTE VISITADA EN EL RECORRIDO. EL TABLERO DEBERÁ SER SIEMPRE UN CUADRADO CON UNA CANTIDAD TOTAL DE CASILLAS PAR.





35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21	 03	
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01

# KNIGHT'S TOUR



# KNIGHT'S TOUR

BÚSQUEDA EN PROFUNDIDAD  
(DEPTH-FIRST SEARCH (DFS))

# PSEUDOCÓDIGO

OFFSET\_MOVIMIENTOS = (

DESCRIBE EL MOVIMIENTO DE EL AGENTE EN EL TABLERO

)

---

DEF MOVIMIENTOS\_DEL\_CABALLERO(FILA, COLUMN, TAMANO\_DEL\_TABLERO)

ITERAR LAS COMBINACIONES DE MOVIMIENTOS POSIBLES

VERIFICAR QUE EL MOVIMIENTO SEA VALIDO

SI ES VALIDO GUARDAMOS EN UN YIELD MOVER\_FILA, MOVER\_COLUMN

---

DEF AGREGAR\_POSIBLE\_MOVIMIENTO(TABLERO, VERTICE\_A, VERTICE\_B):

CREAMOS LA CONEXION DE LOS VERTICES EN EL TABLERO

# PSEUDOCÓDIGO

DEF CONSTRUIR\_TABLERO(TAMAÑO\_DEL\_TABLERO):

    CREAR DICCIONARIO LLAMADO TABLERO

    ITERAMOS FILA EN TAMANO\_TABLERO

        ITERAMOS COLUMA EN TAMANO DE TABLE

            AGREGAMOS LOS POSIBLES MOVIMIENTOS AL TABLERO

    REGRESAMOS EL TABLERO

---

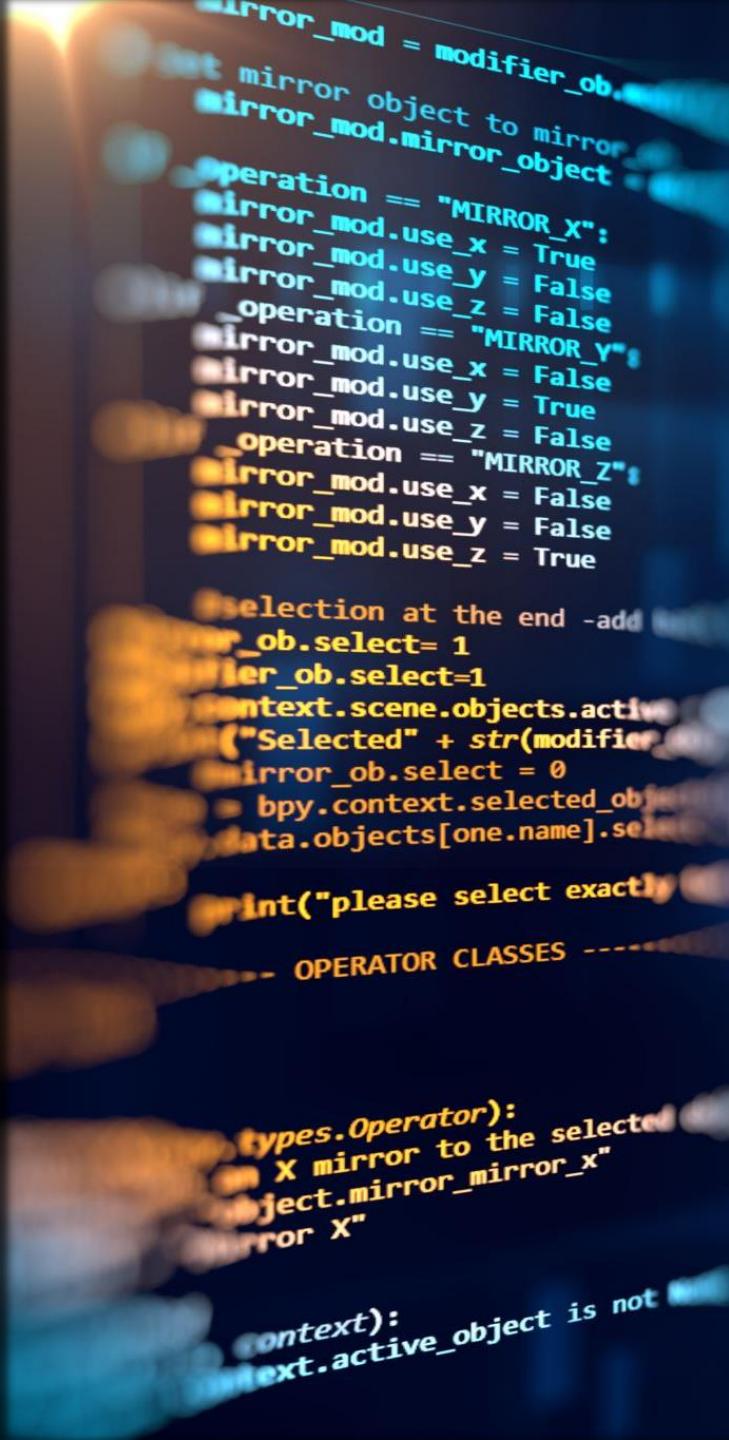
DEF PRIMER\_VERDADERO(SECUENCIA):

    ITERAMOS ITEMS EN SECUENCIA

        SI EXISTE ITEM

            RETORNAMOS ITEM

        SINO RETORNAMOS NONE



# PSEUDOCÓDIGO

DEF DFS(TAMAÑO\_TABLERO, TABLERO):

CALCULAMOS EL TOTAL DE ESPACIOS

DEF EXPLORAR(CAMINO, VERTICE\_ACTUAL):

# UNA VEZ QUE SE HAN EXPLORADO EL TOTAL DE ESPACIOS

# SE REGRESA LA SOLUCIÓN

SI EL CAMINO+1 ES IGUAL AL TOTAL DE ESPACIOS

RETORNAMOS CAMINO+ VERTICE ACTUAL

OBTENEMOS LA LISTA DE LOS NODOS "AUN POR VISITAR"

# SI EN EL VERTICE ACTUAL NO HAY OTRO ESPACIO QUE SE PUEDE VISITAR # ESTAMOS EN UN CAMINO MUERTO

SI NO HAY MAS NODOS POR VISITAR

RETORNAMOS FALSE

# CHECAMOS TODOS LOS VERTICES VALIDOS QUE AUN NO SE VISITAN

REECURSAREMOS LA FUNCION EXPLORAR PARA RECORRER EL TABLERO E IREMOS CREANDO EL CAMINO

AÑADIENDO LOS VERTICES QUE NO SEAN UN CALLEJON SIN SALIDA

RETORNAMOS LA LISTA GENERADA

# PSEUDOCÓDIGO

DEF MAIN():

GENERAMOS EL TAMAÑO DEL TABLERO DE FORMA ALEATORIA

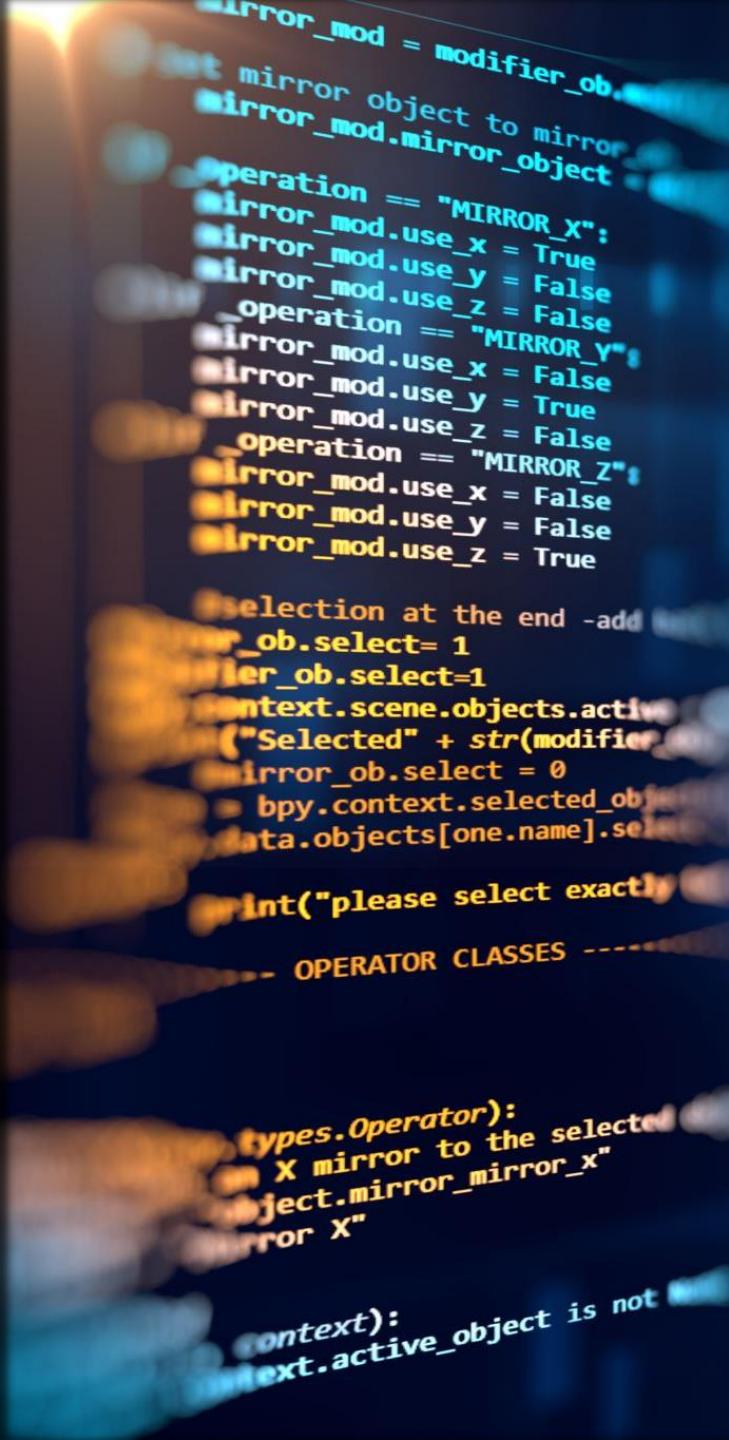
CONSTRUIMOS EL TABLERO

GENERAMOS EL TABLERO MATRIZ

EJECUTAMOS EL DFS

IMPRIMIMOS EL CALCULADO A CAMINO A SEGUIR

IMPRIMIMOS LA MATRIZ(TABLERO) CON EL NUMERO DE MOVIMIENTO  
EN EL QUE FUE SELECCIONADO



```
mirror_mod = modifier_ob
# mirror object to mirror
mirror_mod.mirror_object = mirror_object
operation = "MIRROR_X"
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation = "MIRROR_Y"
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation = "MIRROR_Z"
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

selection at the end -add
modifier.select= 1
modifier.select=1
context.scene.objects.active = ("Selected" + str(modifier))
modifier.select = 0
bpy.context.selected_objects[one.name].select = 1
data.objects[one.name].select = 1
print("please select exactly one object")
- OPERATOR CLASSES -
types.Operator:
  X mirror to the selected object.mirror_mirror_x" or X"
context):
  context.active_object is not None
```



# ALGORITMO GENÉTICO EN THE KNIGHT'S TOUR

# EL ALGORITMO GENÉTICO

- EN GENERAL EL ALGORITMO NO SE DIFERENCIA MUCHO DEL PSEUDOCÓDIGO QUE VIMOS, LA DIFERENCIA FUNDAMENTAL ES COMO GENERAMOS ESA POBLACIÓN Y COMO LA EVALUAMOS.

**¿CÓMO DEMONIOS PODEMOS GENERAR UN STRING DE NÚMEROS BINARIOS PARA RESOLVER ESTE PROBLEMA?**

```
BEGIN /* Algoritmo Genetico Simple */
  Generar una poblacion inicial.
  Computar la funcion de evaluacion de cada individuo.
  WHILE NOT Terminado DO
    BEGIN /* Producir nueva generacion */
      FOR Tamaño poblacion/2 DO
        BEGIN /*Ciclo Reproductivo */
          Seleccionar dos individuos de la anterior generacion,
          para el cruce (probabilidad de seleccion proporcional
          a la funcion de evaluacion del individuo).
          Cruzar con cierta probabilidad los dos
          individuos obteniendo dos descendientes.
          Mutar los dos descendientes con cierta probabilidad.
          Computar la funcion de evaluacion de los dos
          descendientes mutados.
          Insertar los dos descendientes mutados en la nueva generacion.
        END
      IF la poblacion ha convergido THEN
        Terminado := TRUE
    END
  END
```

Figura 1: Pseudocódigo del Algoritmo Genético Simple

# GENERAR UNA POBLACIÓN

- QUEREMOS GENERAR UN STRING DE BINARIOS QUE PODAMOS MUTAR EN BASE A TODAS LAS REGLAS DEL ALGORITMO GENÉTICO.
- DEPENDIENDO DEL LUGAR DONDE ESTÁ EL CABALLERO, ESTE PUEDE TENER DE 2 A 8 MOVIMIENTOS QUE PUEDE TOMAR EN EL TABLERO.
- CADA MOVIMIENTO LO PODEMOS REPRESENTAR DE LA SIGUIENTE MANERA:

• 4 • 3 •	• 100 • 011 •
5 • • • 2	101 • • • 010
• • X • •	• • • X • •
6 • • • 1	110 • • • 001
• 7 • 0 •	• 111 • 000 •

# GENERAR UNA POBLACIÓN

- YA PODEMOS VER DE DONDE SACAR BITS Y CAMBIARLOS PERO, ¿QUÉ PODEMOS HACER CON ESTO?
- TOMEMOS COMO EJEMPLO UN TABLERO DE AJEDREZ DE 8x8, EL CABALLERO TIENE QUE HABER VISITADO TODOS LOS ESPACIOS DEL JUEGO SIN TENER QUE VOLVER A LOS ESPACIOS YA VISITADOS, Y SIN SALIRSE DEL TABLERO. EL CABALLERO TIENE QUE HABER VISITADO 64 ESPACIOS EN EL TABLERO Y HABERSE **MOVIDO** 64 VECES.
- COMO YA HABÍAMOS HABLADO EN LA DIAPOSITIVA ANTERIOR LOS **MOVIMIENTOS** SE PUEDEN EXPRESAR CON 3 BITS, Y CON ESO PODEMOS CREAR UN STRING DE **MOVIMIENTOS**, DE TAL FORMA QUE  $64 \times 3 = 192$ , UN INDIVIDUO TENDRÁ UN TAMAÑO DE 192 BITS.

# INTERPRETAR UNA POBLACIÓN

• 4	• 3	•	• 100	• 011	•	
5	•	•	2	101	•	• 010
•	•	X	•	•	X	•
6	•	•	1	110	•	• 001
• 7	• 0	•	• 111	• 000	•	

11011000101010101...

110-110-001-010-101...

- TENDREMOS UN STRING COMO EL SIGUIENTE:
- ESTE STRING LO PODEMOS DIVIDIR ASÍ:
- EMPEZANDO EN EL TABLERO EN LA POSICIÓN (5, 4) PODEMOS INTERPRETAR LOS PRIMEROS 5 TRIOS DE BITS DE LA SIGUIENTE FORMA:

...**(5,4)**-110->**(3,3)**-110->**(1,2)**-001->**(3,1)**-010->**(5,2)**-101->**(3,3)**

- Así podemos obtener una lista de movimientos que el caballero puede hacer.
- Hay que denotar que (3,3) se repite dos veces en la lista, invalidando así el individuo, ya es con una **EVALUACIÓN** que podemos darle un valor a un string de bits para determinar que tan cerca esta de ser un tour valido.

# EVALUAR UNA POBLACIÓN

- UN TOUR COMPLETO Y VALIDO EN UN TABLERO DE 8x8 TIENE UN TOTAL DE 64 POSICIONES QUE HA TOCADO EL CABALLERO, INCLUYENDO DONDE EMPEZÓ. ESTA FUNCIÓN QUE VA EVALUAR LA POBLACIÓN LE VA A ASIGNAR UN **VALOR** AL STRING DE ACUERDO A CUANTOS **POSICIONES VALIDAS** TIENE, EN BASE A LAS SIGUIENTES RESTRICCIONES:

35	40	47	44	61	08	15	12
46	43	36	41	14	11	62	09
39	34	45	48	07	60	13	16
50	55	42	37	22	17	10	63
33	38	49	54	59	06	23	18
56	51	28	31	26	21		03
29	32	53	58	05	02	19	24
52	57	30	27	20	25	04	01

# RESTRICCIÓN, ESTAR DENTRO DEL TABLERO:

- POR LA NATURALEZA DEL ALGORITMO EL CABALLERO PUEDE EXPLORAR CASILLAS MUY FUERAS DEL TABLERO, EN UN TABLERO 8x8 UN INDIVIDUO PUEDE HABER VISITADO CASILLAS COMO (-1,3), QUE TIENE UN NEGATIVO Y POR LO TANTO ESTA FUERA DEL TABLERO, Y (8,8) QUE ESTA FUERA DEL TABLERO PORQUE LAS COORDENADAS TIENEN UN RANGO DE 0 A 7 Y TENER UN NUMERO MAYOR A ESTE SIGNIFICA QUE SE SALIÓ DEL TABLERO.



# RESTRICCIÓN, NO REPETIDOS

- PUEDE PASAR QUE EL CABALLERO HALLA EXPLORADO UNA CASILLA VALIDA DOS VECES, HACIENDO QUE ESA SEGUNDA CASILLA SEA INVALIDA. ES BASTANTE SIMPLE EN CONCEPTO, SI EN LA LISTA DE MOVIMIENTOS DEL INDIVIDUO SE ENCUENTRA UNA CASILLA QUE SE HA VISITADO MAS DE UNA VEZ, ENTONCES HAY UN REPETIDO Y POR LO TANTO SE TIENE QUE INVALIDAR EL VALOR QUE LOS REPETIDOS DARÍAN.



# EVALUAR UNA POBLACIÓN

- SI EL MOVIMIENTO DE UN INDIVIDUO HA CUMPLIDO CON LAS DOS RESTRICCIONES ANTERIORES ENTONCES EL INDIVIDUO GANA UN PUNTO.
- SI EL INDIVIDUO LLEGA A EXPLORAR TODOS LOS ESPACIOS DE UN TABLERO, 64 EN UN TABLERO 8x8, SU VALOR SERIA IGUAL A ESOS ESPACIOS, O EN OTRAS PALABRAS “SI VALOR\_INDIVIDUO == 64: ES LA SOLUCIÓN.”

