TEC | Tecnológico de Costa Rica

Estudiantes: Ana Belén Murillo, Josué Quintana, Angie Esquivel López

10 de mayo de 2023

Tecnológico de Costa Rica

Introducción a la programación

Knight Tour

Eddy Ramírez Jiménez

Proyecto 2 de Taller

1. Resumen ejecutivo

- **Resumen del proyecto:**
  Este proyecto trata sobre el tour del caballo, en donde se debe implementar la técnica del backtracking. Dado una cantidad de filas, columnas y las coordenadas de la posición inicial del caballo, debe de retornar el camino hamiltoniano que realiza esta pieza de ajedrez. Se busca que el código sea el más eficiente posible, por lo que se implementó varias técnicas para que esto se lograse.

- **Nuestra solución:**
  Lo primero que se realizó fue el análisis del problema, en donde se estudió el comportamiento del caballo en un tablero. Esto hizo que se comprendiera de manera más fácil la implementación del backtracking y todo lo que esto conlleva.
  A partir del código que el profesor brinda para la resolución del proyecto, se codifica la función llamada "hijos"(lo que ingresa en la pila del backtracking), en general en este código se definen en una lista los movimientos que el caballo puede realizar y al llamar la función 'ordenar_movimientos' retorna una lista ordenada de los posibles movimientos que el caballo podría hacer.
  Para lograr uno de los objetivos del proyecto, la eficiencia, se llevaron a cabo varios códigos que hacían que esto fuera posible, que son los siguientes:
  Una de las técnicas ejecutadas fue **la distancia euclidiana** que se implementó usando las coordenadas "x" y "y" de la última posición del caballo y las dimensiones del tablero, itera los posibles movimientos que puede realizar el caballo, la distancia desde el cuadro actual hasta el centro del tablero debe calcularse antes de moverse. Este proceso determina qué maniobras son seguras y posibles para el próximo movimiento del caballo, y se calcula por medio de la siguiente fórmula:

$$dist = \left(\frac{X_1 - X_2}{2}\right)^2 - \left(\frac{Y_2 - Y_2}{2}\right)^2 \tag{1}$$

  El ordenamiento por inserción, fue otra técnica usada, esto se basa en recorrer una lista de elementos y en cada iteración, insertar un elemento en la posición correcta en la parte ya ordenada de la lista, en otras palabras, ordena la lista de forma creciente en base a cada tercer elemento de cada tupla, e inserta el nuevo elemento en la posición en la que se debe en la lista.

- **Resumen de los resultados de las pruebas:**
  Los resultados que se obtuvieron en este proyecto fueron muy favorables, ya que la duración de ejecución del programa es bajo, por lo que cumple con el objetivo de este, además da una respuesta correcta, en donde no se repiten casillas para que el caballo logre el recorrido completo del tablero. Se puede ver que existe una relación entre la duración de ejecución del programa y el tamaño del tablero, ambas incrementan, pero no sobrepasan el límite establecido por el profesor.

Introducción

Este documento consiste en la documentación técnica de la solución de un problema en Python del proyecto. El cual se enfoca en encontrar una solución programable sobre el Tour del caballo, el cual para un tablero de n*n y desde una posición arbitraria, se pueda hallar un camino para el cual el caballo pase por cada casilla sin repetir ninguna. Esto se debe analizar y generar un algoritmo que al ejecutarse pueda correrlo en el menor tiempo posible, dando una respuesta correcta, además que este algoritmo debe funcionar para inputs con matrices de 101 x 99.

Marco teórico

- **Python 3** Python es un lenguaje básico de programación, que permite crear programas de computación; y, su forma básica de utilizarlo se debe a que es un lenguaje similar al humano. Este lenguaje es multiplataforma y de código FOSS, y funciona para muchas aplicaciones, así como la IA, machine learning, etc. como se menciona a continuación por Visus (2020):

  > Python es un lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación. Además, posee una serie de características que lo hacen muy particular y que, sin duda, le aportan muchas ventajas. (párr. 6)[1]

  Siendo así, que python tiene múltiples ventajas para programar y diseñar en computación; sin embargo, esta herramienta de programación tiende a consumir mucha memoría debido a la flexibilidad en los tipos de datos que se utilizan, también, puede llegar a ser muy lenta y no tiende a ser muy eficiente en todas las áreas de programación. En síntesis, python es una herramienta muy útil para desarrollar y programas software, pero al no ser muy eficiente la vuelve un lenguaje muy básico, lleno de limitaciones.

- **Bibliotecas utilizadas**

  Pygame:

  Pygame es un conjunto de módulos de Python diseñados específicamente para crear videojuegos y programas multimedia, que se caracteriza por ser gratuito y ser usado para crear juegos de código abierto. Al agregar funcionalidad a la biblioteca SDL, Pygame le permite crear programas multimedia y juegos completos utilizando el lenguaje de programación Python. Como menciona Pygame Community (s.f.):

  > Utiliza código optimizado en C y ensamblador para las funciones principales. El código en C suele ser de 10 a 20 veces más rápido que el código en Python, y el código en ensamblador puede ser fácilmente de 100 veces o más rápido que el código en Python. (párr. 6)[2]

Una de las grandes ventajas de Pygame es su portabilidad, ya que se ejecuta en una amplia variedad de sistemas operativos y plataformas, esto lo convierte en una herramienta muy accesible y versátil para desarrolladores de todo el mundo.

Math: Se utiliza la biblioteca math para realizar operaciones sencillas dentro del código a utilizar, esta funciona como se va a mencionar por avivcas (2022) "El módulo matemático en python es un módulo estándar en python que tiene funciones para hacer algunos cálculos bastante simples como la adición (+), la sustracción (-) a varias funciones de alto nivel como logarítmica, exponencial, funciones trigonométricas" (párr. 1).[3] Como se menciona anteriormente, gracias a esta biblioteca se pueden hacer operaciones matemáticas sin estar codificándolas, lo que facilita y agiliza la realización del proyecto.

- **IDE´S** El IDE utilizado para realizar el código de este proyecto fue Visual Studio Code (también conocido como VS code), este mismo es un editor de código diseñado por Microsoft, es altamente personalizable y extensible, además de que es muy sencillo de usar, lo que lo hace muy popular, ofrece varias herramientas para la edición de código, la depuración, el control de versiones y la integración de extensiones, en resumen es bastante fácil de usar, intuitivo y muy útil

- **Tour del caballo y demostraciones asociadas** El tour del caballo es un problema de backtracking, en el cual, con base a una pila, se van acumulando los resultados, y se van sacando los que no funcionan, hasta encontrar una solución del problema. como menciona GeeksforGeeks (2023) "Solo se pueden resolver probando todas las configuraciones posibles y cada configuración se intenta solo una vez. Una solución ingenua para estos problemas es probar todas las configuraciones y generar una configuración que siga las restricciones del problema dado" (párr. 1).[4] El tour del caballo consiste en que, en un tablero dado n*n, se coloca un caballo en el primer bloque del tablero vacío, y este se mueve con las reglas del ajedrez; se termina cuando se encuentra un camino que visita todas las casillas del tablero, sin repetir ninguna. Las demostraciones se presentan a continuación:

  El caballo debe buscar un camino desde la posición inicial, hasta visitar todas las casillas sin repetir hasta quedarse sin posibilidades de continuar. Existen dos tipos de tour, el abierto y el cerrado, el primero señala que en un tablero n*n existe tour abierto del caballo si y sólo si n>=5, pero si n es par, entonces el tour es cerrado. Así se menciona a continuación por Álvarez (2012) "Sin prueba alguna, esto es aceptado por la comunidad de pesquisa en esta temática. No es seguro si el problema del tour del caballo sea NP-Completo debido al gran número de algoritmos de solución de orden polinomial propuestos" (p. 2).[5] Esto causa que los problemas del tour del caballo sean casos especiales de caminos hamiltonianos de un grafo.

  El Tour del caballo siempre contará entonces con un camino, porque se llega a obtener un grafo al reemplazar los movimientos del caballo por casillas adyacentes o nodos, en el orden i, j. Por lo tanto, dos casillas son adyacentes si desde una se puede alcanzar a la otra a través de los movimientos del caballo , esto se puede ver representado de la siguiente forma, en la que una casilla es adyacente a otra si y sólo si se cumple esto:

Figura 1

Algoritmo determinístico para resolver el problema del tour abierto del caballo sobre tableros de ajedrez n*n

$$1 \leq a,b,c,d \leq n \text{ y } (c,d) \in \begin{Bmatrix} (a-2,b+1),(a-1,b+2),(a+1,b+2),(a+2,b+1), \\ (a+2,b-1),(a+1,b-2),(a-1,b-2),(a-2,b-1) \end{Bmatrix}$$

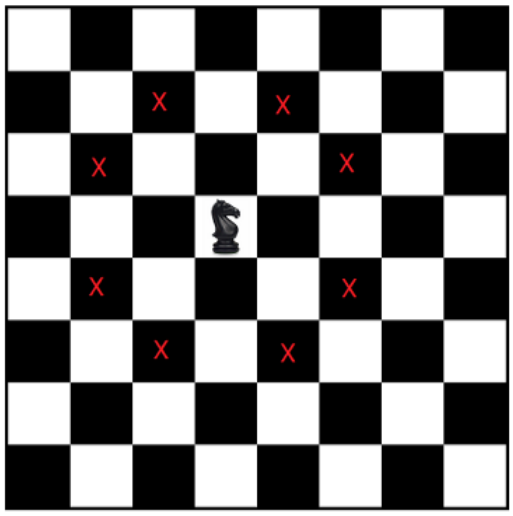
Nota. Adaptado de Algoritmo determinístico para resolver el problema del tour abierto del caballo sobre tableros de ajedrez n*n. Por Álvarez, D y Romero, R. 2012. Congreso Latino-Iberoamericano de Investigación Operativa (http://www.din.uem.br/sbpo/sbpo2012/pdf/arq0509.pdf)

Figura 1: [6]

Y se puede ver gráficamente en un tablero, así:

Figura 2

Knight's Tour



Nota. Adaptado de Knight's Tour. Por IB Maths Resources, 2013. IB Maths Resources. (https://ibmathsresources.com/2013/11/19/knights-tour/)

Figura 2: [7]

Descripción de la solución

Para la solución de este proyecto, se utilizaron 5 funciones, la primera de ellas llamada kt (knight tour), toma como entrada las dimensiones (Filas por columnas) de la matriz o tablero, se establece una pila que va almacenando en una lista las rutas que se van encontrando de la forma (ii, jj), luego se hace un bucle (while) mientras la pila no esté

vacía se extrae una ruta de la pila y si la longitud de la ruta extraída es igual a las dimensiones del tablero (Filas por columnas), entonces se retorna esa ruta encontrada, si no pasa esto, entonces se itera sobre la lista que retorna la función llamada hijos y se agrega a la pila otro de los hijos.

La función llamada hijos recibe como entrada una lista llamada ruta y que es la secuencia de movimientos que ha hecho el caballo hasta el momento. Primero se define una lista de los movimientos que puede realizar el caballo, en forma de L o mejor dicho a alguna de las casillas en diagonal no adyacentes más cercanas, de esta forma "[[2,1], [1,2], [-1,2], [-2,1], [-2,-1], [-1,-2], [1,-2], [2,-1]]" se le especifica al programa que el caballo se puede mover dos casillas hacia arriba y una a la derecha, o una casilla hacia arriba y dos a la derecha y así mismo con los demás movimientos, luego se obtienen las coordenadas del último movimiento hecho por el caballo en la ruta que se está haciendo utilizando slicing (ruta[-1]). Luego se llama a la función *ordenar_movimientos* con los movimientos que puede hacer el caballo, la coordenada "x" y "y" de la última posición que tomó el caballo, y las dimensiones del tablero, esta función retorna una lista ordenada de los posibles movimientos que el caballo podría hacer.

Luego la función hijos, itera sobre esta lista ordenada y comprueba si los movimientos son válidos, o sea si el movimiento no se sale de las dimensiones del tablero y si aún el caballo no ha pasado por esa casilla, si esto pasa entonces se agrega el movimiento a la ruta y además la ruta se agrega a la lista de hijos y ya luego se retorna la lista con todos los hijos resultantes.

La función *ordenar_movimientos*, que toma como entrada los movimientos que puede hacer el caballo, la coordenada "x" y "y" de la última posición que tomó el caballo, y las dimensiones del tablero, itera sobre los movimientos que puede hacer el caballo y verifica si son válidos, para esto el movimiento tiene que estar dentro de los límites del tablero, se calcula la distancia desde la casilla desde la cual se va a realizar el movimiento hasta el centro del tablero. Para esto usamos la fórmula para calcular la distancia euclidiana. Por ejemplo, para dos puntos A y B con coordenadas (X1, Y1) y (X2, Y2), la distancia se puede calcular como: dist = $(X1 - X2//2)^2 - (Y2 - Y2//2)^2$.

Luego la lista con los movimientos se ordena en función de la distancia hasta el centro del tablero para esto se utiliza otra función llamada *ordenamiento_por_insercion*. Esta función *ordenar_movimientos* lo que retorna es una lista ordenada de tuplas que contiene las coordenadas del movimiento en "x" y "y".

La función llamada *ordenamiento_por_insercion*, implementa el algoritmo de ordenamiento por inserción. Primero se comprueba si la lista está vacía o no, si está vacía entonces se agrega el elemento a la lista, si la lista no está vacía se establece una variable i en 0 y luego se itera con un bucle while sobre los elementos de la lista, y se compara el tercer elemento de la lista en la posición i, con el tercer valor de lo que se quiere insertar, el bucle se repite hasta que se llega al final de la lista o hasta que se encuentra un elemento cuyo tercer valor es mayor que el tercer valor del elemento que se quiere insertar, cuando se encuentra la posición correcta en la que se debe insertar el elemento, entonces se inserta usando el método insert, que inserta el elemento en la primera posición de la lista. Básicamente esta función ordena la lista de forma creciente

en base a cada tercer elemento de cada tupla, e inserta el nuevo elemento en la posición en la que se debe en la lista.

Por último, la función *imprimir_tablero*, recibe como parámetros la solución encontrada después de aplicar todo el algoritmo y también las dimensiones del tablero, primero que nada, se crea una matriz nula donde se va a almacenar la solución de manera consecutiva según se va moviendo el caballo. Para esto se establece la variable i en 1, y se itera sobre cada movimiento de la solución encontrada, el valor de la variable corresponde a la posición del caballo en ese respectivo movimiento, el valor de i se va incrementando de uno en uno según cada movimiento del caballo, para así marcarlos en orden consecutivo. Por último, se imprime la matriz o tablero fila por fila y se usa la función format para agregar ceros al inicio del número para que cada número tenga cuatro dígitos y se deja un espacio entre cada valor.

Algo importante a recalcar en la última parte del código, es que se hace una condición para que si el producto de filas por columnas es impar y la suma de las coordenadas i y j es impar entonces no hay solución para esa posición inicial dada si esto pasa entonces se imprime un mensaje que dice que no hay solución y si no pasa entonces se ejecuta normalmente el algoritmo.

Para la interfaz gráfica, primero que nada, importamos el módulo pygame, y establecemos las dimensiones de la ventana de la interfaz en 600 por 600 para que quepa en el monitor, y se establecen los colores blanco y negro que se van a utilizar para hacer el tablero, se establecen también los valores de Fil y Col, que son las dimensiones de la matriz, y se establece que el tamaño del tablero va a ser igual que Fil, o sea si Fil es 6 entonces las dimensiones del tablero serían 6x6.

Se inicializa pygame, y se crea la ventana usando la función display y se le dan las dimensiones de la ventana, se importa la imagen del caballo y se escala según el mínimo entre la altura y la anchura de la ventana dividido entre el tamaño del tablero, lo mismo se hace con la imagen del cuadrado rojo.

Luego se crea una función llamada crear_tablero, en la que primero se establece el tamaño de cada casilla, que va a ser el mínimo entre la altura y la anchura de la ventana dividido entre el tamaño del tablero, luego se itera sobre e tamaño del tablero para crear las filas y columnas, se crea la variable x que va a ser la cantidad de columnas por el tamaño de la casilla y lo mismo con otra variable y, se establece también una variable llamada colores y esto va a ser que si la casilla está en posición par, entonces va a ser blanca, y si es impar entonces va a ser negra.

Se crea otra función llamada dibujar_caballo, que recibe de entrada la posición del caballo, se establece el tamaño de las casillas y unas variables x y y, la variable x va a ser el elemento en la posición 1 de la lista "posición" multiplicado por el tamaño de la casilla, y la variable y va a ser el elemento en la posición 0 de la lista "posición" multiplicado por el tamaño de la casilla, estas variables van a ser las coordenadas de la esquina superior izquierda de la casilla en la que se encuentra el caballo, luego se establece el tamaño del caballo restando un margen de 10 píxeles a cada lado de la casilla y se obtiene un rectángulo con la imagen del caballo y se ajusta el tamaño en base al tamaño calculado antes. Esta función se encarga de posicionar y ajustar la

imagen del caballo en cada casilla. Todo esto mismo, se realiza también con la imagen del cuadrado rojo.

Por último, con lo que tiene que ver con la interfaz, se crea una función llamada int_grafica, se establece que la solución va a ser una llamada a la función kt, para que se ejecute todo el algoritmo de la solución al problema del tour del caballo, luego si hay solución, entonces se llama a la función para crear el tablero y se itera sobre la solución, por cada paso o movimiento del caballo, se dibuja el caballo y se actualiza o resetea en la pantalla, y por cada paso que da el caballo también se dibuja el cuadro rojo pero este no se actualiza para que así quede marcado en rojo cada movimiento que hizo el caballo, se esperan 500 milisegundos cada vez que esto pasa para que se pueda apreciar mejor y que no sea tan instantáneo, por último se finiquita pygame cuando ya se completó toda la ruta y el caballo visitó todas las casillas.

Resultados de las pruebas

El código es ejecutado en el sistema operativo macOs Ventura, con una computadora marca Apple, en el momento en el que se realizan las pruebas no se va a correr nada más en el computador. Como se aprecia en la gráfica, los tiempos que tarda el algoritmo

algunos constantes y otros relativos independientemente del caso que se ejecutará, esta gráfica fue realizada de acuerdo a un promedio de casos que se desarrollaron que van desde una matriz 6*6 hasta una matriz 101*99. Se adjuntan los números de prueba y los tiempos a continuación:

6*6 2 3	0.01
6*6 0 0	0.02
6*6 5 5	0.03
8*8 0 0	0.19
8*8 2 3	0.02
8*8 4 7	0.04
20*20 10 15	0.03
20*20 19 19	0.03
20*20 14 8	0.03
80*80 50 30	3.34
80*80 70 79	7.23
80*80 40 65	3.44
101*99 30 50	9.53
101*99 20 60	9.39
101*99 43 75	9.5

En términos de exhaustividad, el código es completo, debido a que devuelve correctas las salidas y no tarda mucho tiempo en dar una respuesta, por lo que se puede afirmar que es bastante eficiente.

Conclusiones

Como análisis general de los resultados obtenidos, se tiene que este se ejecuta de manera rápida con cualquier cantidad de filas y columnas dadas, así como con cualquier posición inicial del caballo, esto se debe a que se están ordenando los posibles movimientos que puede realizar el caballo desde una casilla, de manera que los que tengan menor distancia hasta el centro del tablero se prueben primero.
En lo que respecta a la eficiencia del proyecto, este podría ser más eficiente implementando una estructura de backtracking de cola, en vez de una de pila, junto a esto si se utilizan funciones como lambda o .sort, se logra disminuir el tiempo de ejecución del algoritmo de manera significativa.

Aprendizajes

- **Josué Quintana**
  Durante el proyecto, se aprendió mucho sobre cómo abordar problemas y trabajar en equipo para encontrar soluciones efectivas. Además, antes de comenzar el proyecto, se invirtió tiempo y esfuerzo en comprender claramente el problema que se estaba tratando de resolver para así intentar encontrar una solución más fácil y efectivamente. El equipo aprendió a comunicarse claramente, escuchar activamente y proporcionar retroalimentación constructiva para garantizar que todos los miembros del equipo aportaran ideas válidas para la resolución del proyecto. El proyecto fue una oportunidad valiosa para aprender y desarrollar habilidades clave, desde la comprensión del problema hasta la resolución de problemas y la

comunicación efectiva. Estos aprendizajes no solo fueron útiles para el proyecto en sí, sino que también pueden ser aplicados a otros proyectos en el futuro.

- **Angie Esquivel**
  Con este proyecto aprendí a tener perseverancia, tema que me va a ayudar durante todo el trayecto de mi carrera universitaria y futura vida laboral como programadora. Además aprendí la importancia del buen trabajo en equipo, colaborar con mis compañeros y compartir nuestras ideas nos ayudó a un éxito en el desarrollo del proyecto. Y por último, aprendí a tener un enfoque más creativo y abierto con lo respecta a ideas para la resolución del problema.
- **Ana Murillo**
  En este proyecto, aprendí a escuchar a mis compañeros y a tener paciencia, también a ser comprensiva con la forma en la que trabajan otros compañeros. Entendí que este trabajo no se hubiese podido lograr sin alguno de los tres, porque a pesar de que a los tres nos cuesta un poco (mucho), nos supimos apoyar en los otros para que este trabajo diera un buen resultado, y cada uno aportó de forma diferente en este proyecto. Pero no solo aprendí eso, sino también logré entender algunos de los contenidos del curso, y cómo trabajar con estos para la realización del proyecto.

Referencias bibliográficas

[1] Visus, A. (2020). ¿Para qué sirve Python? Razones para utilizar este lenguaje de programación. esic. https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

[2]Pygame Community. (s.f.). About - wiki. pygame wiki. https://www.pygame.org/wiki/about

[3]avivcas. (2022). Import math python para que sirve. Todo sobre JAVA. https://quejava.com/import math-python-para-que-sirve/

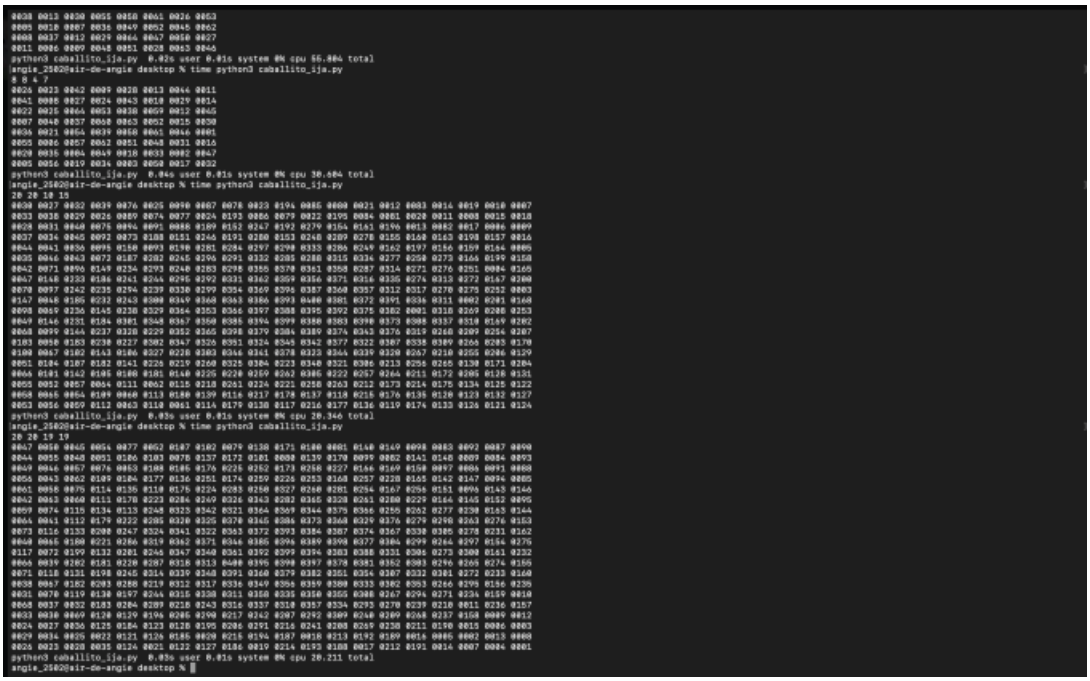
[4]GeeksforGeeks. (2023). The knight's tour problem. GeeksforGeeks https://www.geeksforgeeks.org/t knights-tour-problem

[5]IB Maths Resources. (2013). Knight's Tour. Intermathematics https://ibmathsresources.com/2013/ tour/

[6]Álvarez, D y Romero, R. (2012). Algoritmo determinístico para resolver el problema del tour abierto del caballo sobre tableros de ajedrez n*n. Congreso Latino-Iberoamericano de Investigación Operativa. http://www.din.uem.br/sbpo/sbpo2012/pdf/arq0509.pdf

[7]Techie Delight. (s. f.). Print all possible Knight's tours on a chessboard. Techie Delight. https://www.techiedelight.com/es/print-possible-knights-tours-chessboard/

Anexos


0038 0013 0030 0055 0058 0061 0026 0063
0005 0018 0007 0036 0049 0052 0045 0062
0008 0037 0012 0029 0064 0047 0050 0027
0011 0006 0009 0048 0051 0028 0053 0046
python3 caballito_ija.py 0.02s user 0.01s system 0% cpu 56.804 total
angie_2502@air-de-angie desktop % time python3 caballito_ija.py
8 0 4 7
0026 0023 0042 0009 0038 0013 0044 0011
0041 0006 0027 0024 0043 0010 0029 0014
0022 0025 0044 0053 0038 0059 0012 0045
0007 0040 0037 0058 0063 0052 0015 0030
0036 0021 0054 0039 0050 0061 0046 0001
0055 0006 0057 0062 0051 0048 0031 0016
0020 0035 0004 0049 0018 0033 0002 0047
0005 0056 0019 0034 0003 0060 0017 0032
python3 caballito_ija.py 0.04s user 0.01s system 0% cpu 30.604 total
angie_2502@air-de-angie desktop % time python3 caballito_ija.py
20 20 18 15
0038 0027 0032 0039 0076 0025 0090 0067 0078 0023 0194 0085 0008 0021 0012 0003 0014 0019 0010 0007
0033 0036 0029 0026 0089 0074 0077 0024 0193 0086 0079 0022 0195 0084 0081 0020 0011 0008 0015 0018
0028 0031 0040 0075 0096 0095 0088 0189 0192 0247 0192 0279 0164 0161 0196 0013 0082 0017 0086 0009
0037 0034 0045 0092 0073 0188 0191 0246 0191 0280 0163 0248 0289 0278 0185 0160 0163 0198 0157 0016
0044 0041 0056 0095 0150 0093 0190 0281 0284 0297 0298 0335 0286 0249 0162 0197 0156 0159 0144 0085
0035 0046 0043 0072 0187 0282 0245 0296 0291 0332 0285 0288 0315 0334 0277 0258 0273 0166 0199 0158
0042 0071 0096 0149 0234 0293 0248 0283 0295 0355 0370 0361 0358 0287 0314 0271 0276 0251 0084 0165
0047 0148 0233 0146 0241 0244 0296 0292 0331 0342 0359 0356 0371 0316 0335 0274 0313 0273 0167 0200
0070 0097 0242 0235 0294 0239 0338 0379 0354 0360 0396 0387 0360 0357 0312 0317 0270 0275 0252 0083
0147 0048 0166 0232 0243 0300 0349 0368 0363 0366 0393 0400 0381 0372 0391 0336 0311 0002 0201 0168
0098 0069 0236 0145 0238 0329 0364 0353 0366 0397 0388 0395 0392 0375 0382 0001 0310 0269 0288 0253
0049 0146 0231 0184 0381 0349 0367 0360 0385 0396 0399 0380 0383 0398 0373 0388 0337 0310 0149 0202
0068 0099 0164 0237 0328 0229 0352 0365 0398 0379 0384 0389 0374 0343 0376 0317 0268 0289 0256 0287
0103 0050 0183 0238 0227 0302 0347 0326 0391 0324 0345 0342 0377 0322 0387 0338 0389 0266 0285 0170
0100 0067 0102 0143 0186 0327 0228 0303 0346 0341 0378 0323 0344 0339 0320 0267 0210 0255 0286 0129
0051 0104 0107 0182 0141 0226 0219 0260 0325 0304 0223 0340 0321 0306 0213 0256 0265 0130 0171 0284
0066 0101 0162 0185 0180 0101 0148 0225 0220 0259 0262 0305 0222 0257 0264 0211 0172 0285 0128 0131
0065 0052 0057 0054 0111 0062 0115 0218 0261 0224 0221 0258 0263 0212 0173 0214 0175 0134 0125 0122
0068 0066 0064 0109 0060 0113 0180 0139 0116 0217 0178 0137 0118 0216 0176 0135 0120 0123 0132 0127
0053 0056 0059 0112 0063 0110 0061 0114 0179 0138 0117 0215 0177 0136 0119 0174 0133 0126 0121 0124
python3 caballito_ija.py 0.03s user 0.01s system 0% cpu 20.346 total
angie_2502@air-de-angie desktop % time python3 caballito_ija.py
20 20 19 19
0047 0058 0045 0054 0077 0052 0187 0102 0079 0130 0171 0180 0001 0148 0169 0098 0083 0092 0087 0090
0044 0055 0048 0051 0186 0103 0078 0137 0172 0181 0008 0139 0170 0099 0082 0141 0148 0089 0084 0093
0049 0046 0067 0076 0063 0108 0105 0176 0226 0262 0173 0258 0227 0146 0169 0150 0097 0086 0091 0088
0056 0043 0062 0109 0104 0177 0136 0251 0174 0259 0226 0253 0168 0257 0228 0165 0142 0147 0094 0085
0061 0058 0075 0114 0135 0110 0176 0224 0203 0260 0327 0260 0281 0294 0167 0256 0151 0096 0143 0146
0042 0063 0060 0111 0178 0223 0204 0249 0326 0343 0282 0365 0328 0261 0280 0229 0144 0145 0152 0095
0059 0074 0115 0134 0113 0248 0325 0342 0321 0364 0369 0344 0375 0366 0255 0262 0277 0230 0163 0144
0064 0061 0112 0179 0222 0205 0320 0325 0370 0345 0386 0373 0368 0329 0376 0279 0240 0263 0276 0153
0073 0116 0133 0208 0247 0324 0341 0322 0363 0372 0393 0384 0387 0374 0367 0330 0385 0278 0231 0162
0048 0066 0100 0221 0206 0319 0362 0371 0346 0385 0394 0389 0398 0377 0384 0299 0264 0297 0154 0276
0117 0072 0199 0132 0201 0246 0347 0340 0361 0392 0399 0394 0383 0388 0331 0306 0273 0300 0161 0232
0066 0039 0202 0181 0220 0287 0318 0313 0408 0395 0398 0397 0378 0381 0362 0303 0296 0269 0274 0155
0071 0118 0131 0198 0245 0314 0339 0348 0391 0360 0379 0382 0351 0354 0307 0332 0301 0272 0233 0160
0050 0057 0182 0293 0288 0219 0312 0317 0336 0340 0396 0359 0309 0333 0302 0303 0266 0298 0156 0235
0031 0070 0119 0130 0197 0244 0315 0338 0311 0358 0335 0350 0355 0308 0267 0294 0271 0234 0159 0010
0068 0037 0032 0183 0284 0289 0218 0243 0316 0337 0310 0357 0334 0293 0270 0239 0210 0011 0236 0157
0033 0030 0069 0124 0139 0196 0286 0290 0217 0242 0297 0292 0309 0240 0289 0268 0237 0158 0000 0012
0024 0027 0050 0125 0184 0123 0128 0195 0206 0291 0216 0241 0208 0269 0238 0211 0190 0015 0096 0003
0029 0034 0028 0022 0121 0126 0185 0020 0215 0194 0187 0018 0213 0192 0189 0014 0005 0002 0013 0008
0026 0023 0038 0035 0124 0021 0122 0127 0186 0019 0214 0193 0188 0017 0212 0191 0016 0007 0004 0001
python3 caballito_ija.py 0.03s user 0.01s system 0% cpu 20.211 total
angie_2502@air-de-angie desktop %