# GUIÃO VÍDEO DE INTELIGÊNCIA ARTIFICIAL

## Apresentação do grupo e do problema

Apresentação-grupo: Olá, sou o Duarte Cruz e eu sou o André Pagaime do grupo 32 e neste vídeo vamos explicar o nosso projeto para a cadeira de Inteligência Artificial.

nuruomino-explicacao: Nuruomino, ou LITS, é um jogo que consiste em preencher com tetraminos (mostrar fotos dos tetraminos, com reflexões e rotações) uma grelha quadrada dividida em regiões seguindo algumas regras, tais como:  
  
regra1 - Cada região tem apenas 1 tetramino

regra2 - Não pode haver tetraminos quadrados nem pode haver quadrados formados pela junção de tetraminos (mostrar fotos)

regra3 - Dois tetraminos iguais não podem ser ortogonalmente adjacentes, considerando reflexões e rotações como iguais

regra4 - Todas as regiões têm de estar ortogonalmente ligadas, formando um único polinómio

objetivo- O objetivo deste projeto é desenvolver um programa em python que dada uma instância de Nuruomino, retorne a sua solução única utilizando técnicas de procura e resolução de problemas de Inteligência Artificial.

## Abordagem

Abordagem-inicio: Deste modo, a nossa abordagem para este problema é guardar todas as possíveis ações dentro de cada região, tendo como objetivo restringir ao máximo o número de possibilidades até se atingir a solução, isto é, eliminar o maior número de ações antes de realizar uma procura.

Acao-contexto: Uma ação no contexto deste problema é colocar uma peça numa região, de forma a preenchê-la, de maneira a restringir ao máximo as regiões adjacentes, para que os ramos de procura sejam minimizados.

Pre-inicio: Deste modo, tivemos de realizar um pré-processamento do tabuleiro recebido no input, em que se analisa todas as ações possíveis em cada região, eliminando precocemente as que não tocavam em nenhuma região adjacente. Desta maneira, remove-se as peças que não tornariam as regiões ortogonalmente ligadas.

possibleShapes (cortar pausa)- Para isto, recorremos a uma procura DFS aplicada a cada célula da região. Por não ser possível calcular os tetraminos T diretamente com esta procura, fez-se uma verificação adicional para os detetar.

grafoInicio - Além disso, o pré-processamento constrói um grafo bi-direcional com peso cujos nós são as regiões e cada região está ligada às suas adjacentes. O peso da conexão região A -> região B corresponde ao número de ações da região A cujos tetraminos tocam na região B.

ilhasCheck - Deste modo, o pré processamento verifica se há alguma região que apenas tem uma região adjacente. Em caso afirmativo, significa que ambas regiões têm obrigatoriamente de se tocar, pelo que eliminamos da região adjacente todas as ações que não tocam na região isolada, ou que apenas tocam na região isolada.

Consequentemente - Consequentemente, o programa verifica ainda se a região adjacente se tornou isolada devido à restrição de ações possíveis. Caso se confirme, o programa realiza o mesmo tratamento que fez anteriormente, desta vez considerando a região adjacente como a região isolada.

Overlap-início : Além disso, o pré-processamento realiza ainda um overlap das ações possíveis de cada região, verificando se há quadrados comuns a todas, preenchendo-os.

Overlap-meio: O overlap pode ser resultado de ações com diferentes tetraminos ou então com o mesmo tetramino. Caso seja com o mesmo tetramino, então removemos das regiões adjacentes todas as possibilidades que contêm tetraminos iguais e que os tornavam ortogonalmente adjacentes. Eliminam-se ainda todas as possibilidades das regiões adjacentes que, em conjunto com os quadrados preenchidos da região original, fariam um quadrado 2 por 2, aplicando-se o **Forward Checking** e o **AC-3** para atualizar o domínio.

DFS-Tree: Após o pré-processamento, se ainda existirem regiões com várias ações possíveis, o nosso programa recorre ao algoritmo de procura **DFS Tree Search** para explorar as opções.

Backtracking: O nó atual é desenvolvido o mais profundamente possível e, caso se chegue um estado completamente preenchido que não corresponda a uma solução, é feito **backtracking**. Se for encontrada uma região ainda por preencher, o programa expande todas as possibilidades dessa região.

MRV: No processamento, o programa seleciona a região com menor número de possibilidades para desenvolver, aplicando o MRV, e, em caso de empate, escolhe a região com mais adjacentes de forma a introduzir mais restrições no problema, para assim, reduzir a profundidade da árvore de procura.

Após-acao: Assim, após aplicar a ação, o programa realiza o mesmo tratamento que realizou no pré-processamento, mas desta vez apenas para esta região e para as suas adjacentes.  
  
GrafoRemove: A cada possibilidade removida, tanto no pré-processamento como durante o processamento, caso se tenha removido a última possibilidade de uma região que tocava numa adjacente, então remove-se do grafo as conexões Região - Região Adjacente e Região Adjacente – Região. De seguida, verifica-se se ambas as regiões ficaram isoladas. Caso isso se confirme, aplica-se o tratamento de **regiões isoladas** descrito anteriormente.

FimYapping: Tendo isto em conta, após realizar a ação, o nosso programa verifica se o tabuleiro é o objetivo, verificando se todas as regiões estão preenchidas e

ortogonalmente ligadas. Caso não seja objetivo, o algoritmo de procura faz backtracking, explorando a próxima possibilidade do nó pai.

## Procura utilizada

Grafico: Tal como referido anteriormente, o algoritmo de procura utilizado foi a DFS tree search, já que após comparar os tempos de procura, foi a que teve menor média. Para isto construímos este gráfico em que se observa de forma clara que é este o algoritmo com gráfico de menor área, isto é, é este o algoritmo com menor tempo médio de procura da solução.

## Fim

BotFim: Por fim, foi desenvolvido um script capaz de resolver automaticamente puzzles disponibilizados no site referido no enunciado, com o objetivo de demonstrar, de forma prática, a eficácia da solução implementada.

## Onde usamos ChatGpt

- resolução de bugs no código e ajuda no desenvolvimento de algumas funções básicas

- uso de Claude e Gemini