PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

Ciência de Dados e Inteligência Artificial

Enzo Ambrosio da Costa - 24008773

Gabbriel Vicente Hiroshi Nagano - 24005804

Gabriel Benevides Bosso - 24013653

João Vitor Guedes del Ducca - 24012015

“Projeto Integrador”

“IA no 8-Puzzle”

Introdução

O projeto integrador tem como objetivo criar um 8-puzzle, um quebra cabeça criado em uma matriz 3x3 que para ser resolvido deve ser organizado de forma na qual a posição dos números de 1 a 8 estejam na “ordem crescente” e a posição do número 0 (casa vazia) seja a última. O projeto deve ter 3 modos de jogo, 2 com uma IA resolvendo o puzzle e 1 no qual o jogador resolve manualmente. Os modos nos quais a IA resolve foram feitos pelos métodos de busca profundidade limitada iterativa e A*, cada um com abordagens distintas para encontrar a solução.

Desenvolvimento do Projeto

Um pouco sobre o trabalho:

Durante o desenvolvimento do trabalho, na primeira parte, acabamos começando a tentar implementar a busca em largura. E isso acarretou em diversos problemas relacionados à memória e também não conseguimos achar um jeito mais rápido e que não gastasse tanta memória usando a busca em largura. Então, acabamos mudando para a busca em A*.

Ao pesquisar sobre o assunto chegamos a conclusão que o melhor método seria o Manhattan, nesse momento, nosso grupo se dividiu e uma parte acabou cuidando apenas de fazer o código do A* usando o método Manhattan e outra parte do grupo ficou responsável por fazer a parte do código de busca em profundidade limitada iterativa.

Porém, após cada equipe finalizar o seu próprio código, decidimos juntar os dois códigos, junto do nosso código base do primeiro trabalho, e ao fazer isso encontramos diversas funções que tinham a mesma linha de raciocínio, porém cada um fez de seu modo e tinha certas diferenças. Então optamos por colocar as funções que se parecem no código para não ter problemas futuros.

Dificuldades:

- Para que sempre houvesse uma solução possível para o 8-puzzle ao invés de aleatorizar os números na matriz, utilizamos a matriz original e fomos aleatorizando os movimentos da casa vazia pela matriz por 50 vezes, desse modo sempre é gerado um estado que é solucionável.
- Começamos a fazer por largura e ao tentar rodar o código a memória era insuficiente, por isso decidimos optar pelo método A* utilizando a Distância de Manhattan como heurística.
- Ao implementar a busca por profundidade limitada iterativa até a IA achar uma solução possível poderia demorar ou não. Nesse caso não havia muito o que consertar, já que a busca é influenciada pela maneira na qual a matriz está montada.
- Ao rodar a parte do A* várias vezes notamos que em alguns casos até mesmo solucionáveis o código dava erro, notamos que o vetor dos “visitados” ficava cheio e então o código não conseguia gerar estados para solucionar o estado inicial. Para resolver isso utilizamos a função memset da biblioteca <string.h>.

Funções Extras:

- Utilizamos o método de Hash básico no A* para a verificação dos estados que foram visitados, gerando uma chave para cada um desses estados e pela chave que foi gerada guarda-se se foi visitado no vetor “visitados[chave]”.
- Utilizamos o método de Manhattan para a verificação do custo de um estado ao objetivo. Esse cálculo é feito na função “distman”.
- Utilizamos a funções “Profundo” e “EstadoEstrela* aheurist” para fazer as operações propriamente de ambas as buscas de profundidade interativa e A* respectivamente.

- As funções “gerarchildren” e “gerarSucessores” possuem a mesma linha de raciocínio e são funções que tem como o objetivo gerar mais tabuleiros porém o primeiro é usado na busca A* e o segundo na busca em profundidade.
- A função “verificar_vitoria” é usada para verificar se o atual tabuleiro é igual ao tabuleiro solucionado.
- As funções “modo_a_estrela”, “modo_profundidade_limitada” e “modo_manual” são as funções onde criamos meio que o “menu” de cada modo e aparece
- A função memset da biblioteca <string.h> é para reinicializar o vetor “visitados”, fazendo com que ele sempre tenha espaço para visitar outros estados.

Bibliografia

- https://www.cs.princeton.edu/courses/archive/fall15/cos226/assignments/8puzzle.html - acesso em 30/11/2024
- https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/ - acesso 30/11/2024
- https://www.geeksforgeeks.org/hash-functions-and-list-types-of-hash-functions/ - acesso 30/11/2024
- https://stackoverflow.com/questions/52622561/calculating-manhattan-and-euclidean-distance - acesso 30/12/2024
- https://www.geeksforgeeks.org/maximum-manhattan-distance-between-a-distinct-pair-from-n-coordinates/ - acesso 30/12/2024
- https://stackoverflow.com/questions/13053455/8-puzzle-solution-executes-infinitely - acesso 01/12/2024
- https://gist.github.com/juniorcesarabreu/ff81bac6dd8510bcd1a816f270223168 - acesso 04/12/2024
- https://www.ibm.com/docs/pt-br/i/7.5?topic=functions-memset-set-bytes-value - acesso 04/12/2024

- https://pt.stackoverflow.com/questions/296121/d%C3%BAvidas-em-rela%C3%A7%C3%A3o-a-uso-das-fun%C3%A7%C3%B5es-memset-e-memcpy - acesso 04/12/2024