TRABALHO DE BUSCA COMPETITIVA COM O JOGO PACMAN

Alunos:  
Lucas Bauchspiess  
Rafael Ehlert  
Tiago Bodnar

1. **Descrição e Regras do Jogo**

O jogo escolhido é **Pacman**, com as seguintes regras:

1. O Pacman deve comer todas as comidas do mapa.
2. Cada comida: **10 pontos**.
3. Cada passo do Pacman: **-1 ponto**.
4. Se o Pacman for pego pelo fantasma: **-10000 pontos**. Utilizado um valor bem alto, para evitar ao máximo que isto aconteça
5. Há **apenas um fantasma**, que tenta impedir o Pacman de ter sucesso.
6. O ambiente é uma matriz 2D, com posições que podem conter:
7. Pacman (P)
8. Fantasma (F)
9. Comida (.)
10. Paredes (#)
11. Espaço vazio ( )

Obs.: Para facilitar a análise e compreensão, será utilizado apenas um fantasma que tentará capturar o Pacman. A matriz pode ser modificada conforme livre escolha de quem inicializa o programa.

1. **Formulação do Problema de Busca**
2. **Estado inicial**: posições iniciais do Pacman, fantasma e todas as comidas.
3. **Ações**: movimentos possíveis (cima, baixo, esquerda, direita), considerando as paredes.
4. **Função de transição**: movimenta o Pacman (e o fantasma), atualiza posições e pontuação.
5. **Função objetivo**: maximizar a pontuação total.
6. **Estado final**: todas as comidas foram comidas ou o Pacman foi pego.
7. **Tipo de Jogo (características)**
8. **Determinístico**: As ações têm efeitos previsíveis.
9. **Competitivo**: Pacman e fantasma têm objetivos opostos.
10. **Ambiente totalmente observável**: Todas as informações do mapa estão disponíveis.
11. **Algoritmo Escolhido**

Escolhemos o algoritmo **Minimax com poda alfa-beta e com profundidade limitada**, pois no Pacman:

1. Trata-se de um jogo competitivo de dois jogadores (Pacman e Fantasma).
2. É necessário tomar decisões estratégicas de ambos os lados.

Obs.: Ele tende a trazer soluções ótimas, porém, em mapas grandes, o algoritmo minimax poder ser muito custoso. Por isso, as podas alfa-beta ajudam a diminuir sua escalabilidade e limitamos sua profundidade para limitar a escalabilidade e torna-lo acessível. Entretanto, desta forma, é possível que não seja alcançado o melhor estado possível.

1. **Resultados**

O algoritmo tornou possível tanto o pacman comer todas as comidas quando o fantasma pegar o pacman antes disso acontecer, encerrando a execução do programa. Porém, dependendo de como o Pacman são inicializado na matriz e da configuração da matriz, podem entrar em loop infinito, onde o pacman foge do fantasma, que em seu turno tenta capturar o pacman. Após o fantasma se mover, a matriz faz o Pacman retornar ao estado anterior, consequentemente, fazendo o fantasma voltar à posição anterior também, resultando em um loop.

1. **Conclusão**

O algoritmo **Minimax** foi escolhido pois este é indicado por se tratar de um jogo competitivo e determinístico com dois jogadores. Apesar disso, não é possível utilizá-lo no contexto completo do pacman, pois a quantidade de estados possíveis tende a escalar muito rapidamente. Desta forma, limitamos a quantidade de níveis seguintes para a análise tornar-se possível. Desta forma, tanto o Pacman ser pego quanto todas as comidas serem capturadas pelo Pacman. Outro estado possível é o programa entrar em loop infinito, conforme ações do Pacman e do Fantasma