Atividade T1 - INE5430 - Inteligência Artificial

Caique Rodrigues Marques
c.r.marques@grad.ufsc.br

Fernando Jorge Mota
contato@fjorgemota.com

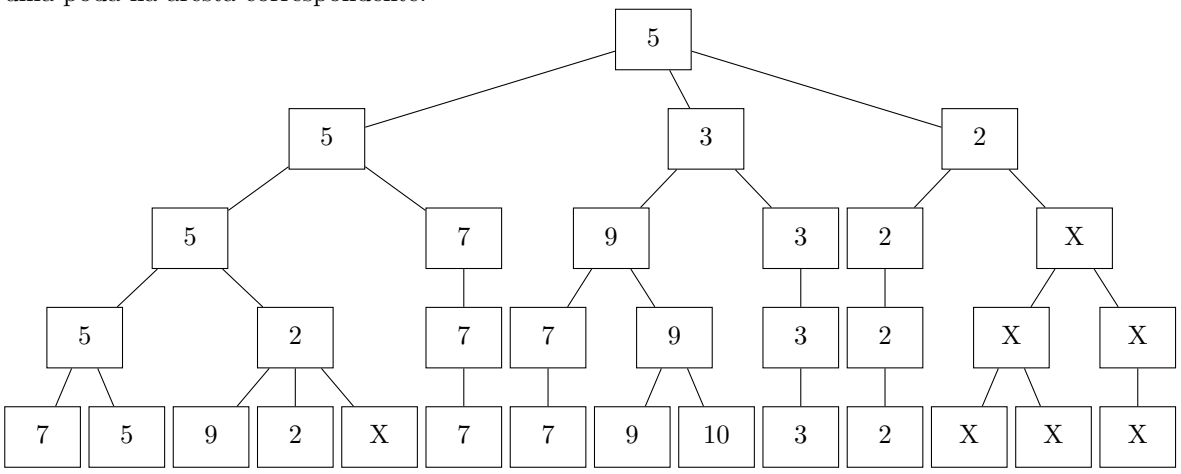
02 de setembro de 2016

Questão 1

Como é de se esperar para o algoritmo do minimax com podas alfa e beta, temos que, inicialmente, vai ser associado, primeiramente, a todos os nodos os valores $\alpha = -\infty$ e $\beta = \infty$. Depois disso, é considerado o algoritmo do minimax e configurado os valores α e β de forma a permitir descobrir que nodos podem ser cortados ou não. Para facilitar, resolvemos separar em duas subseções os valores obtidos após aplicar o algoritmo (no caso, os valores finais dos nodos e os valores alfa e beta associados a cada nodo). Boa leitura!

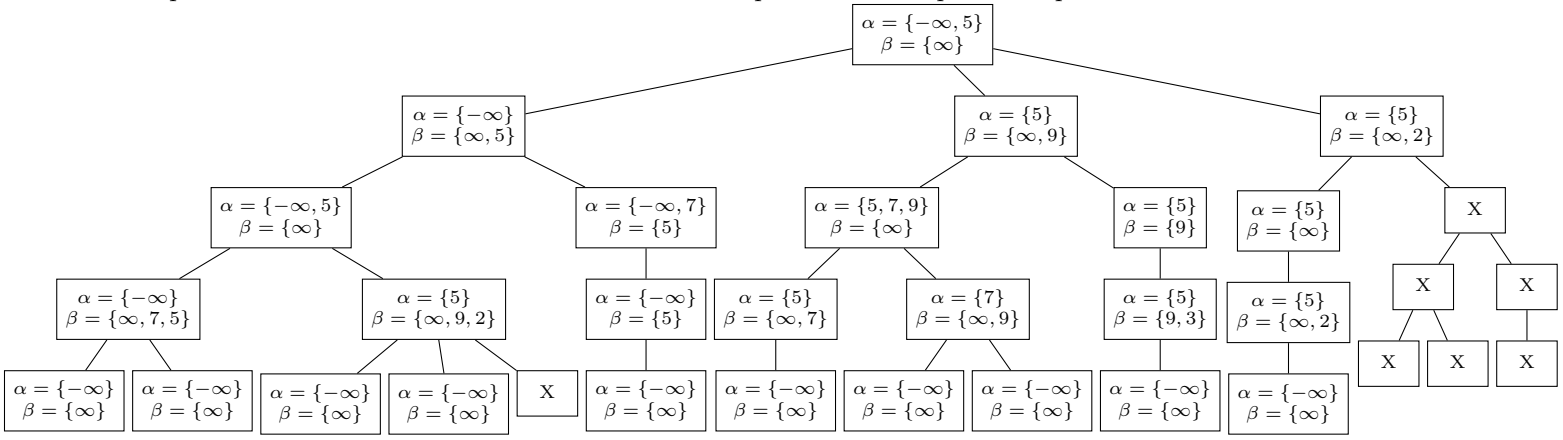
Valores finais dos nodos

A árvore abaixo representa os valores finais dos nodos. Os nodos marcados com um "X"representa que houve uma poda na aresta correspondente.



Valores Alfa e Beta

A árvore abaixo representa os valores alfa e beta na árvore de acordo com o tempo. Os valores mais à direita representam o valor final. Os nós com o valor "X"representam nós que foram podados.



Questão 2

Para definir uma boa função heuristica e uma boa função utilidade para o jogo de damas, precisamos definir as condições que cada função deve considerar e isso depende do objetivo de cada função.

Sabemos que, para o caso da função heurística, temos como meta determinar **o quão próximo** estamos da vitória ou da derrota, de forma aproximada. Para isto, a função deve retornar valores positivos quando o estado considerado representar uma chance de vitória e valores negativos quando o estado considerado representar uma chance de derrota.

Dado essas informações, podemos facilmente idealizar algumas regras que a função heuristica **deve** considerar. No caso do jogo de damas, sabemos que:

- O jogador está mais perto de vencer conforme a quantidade de damas;
- Da mesma forma, está mais perto de perder conforme a quantidade de damas do adversário;
- Para ter damas, o jogador deve alcançar a última linha do tabuleiro, logo, quanto mais perto da última linha, maior a probabillidade de vencer;
- Da mesma forma, o jogador deve evitar que o adversário alcance a última linha do tabuleiro;
- O jogador tem mais chance de ganhar se tiver mais peças que o jogador adversário;
- Da mesma forma, ele tem mais chance de perder se tiver menos peças que o jogador adversário;
- Quando o jogador tiver a possibilidade de capturar peças do adversário, ele está alcançando passos da vitória. As chances aumentam quando o jogador consegue capturar mais de uma peça adversária, numa sequência e em uma mesma jogada, aumentando ainda mais as chances de vitória;
- A estratégia utilizada pelo jogador também deve considerar a posição em que suas peças estão em relação ao adversário, de forma em que este não consiga vantagem capturando várias peças do jogador em uma sequência;
- Em caso de jogadas positivas ao jogador, como por exemplo, a captura de peças, é desejável em que ele consiga capturar o maior número de peças no menor tempo possível, inclusive, é desejável que ele tente ao máximo de jogadas possíveis para evitar que suas peças sejam capturadas.

O algoritmo para uma função heuristica seria algo mais ou menos assim:

```
def heuristica(estado):
    resultado = 0
    # Nos casos a favor do computador, soma..
    resultado += quantidade_damas(estado, cor_computador)
    resultado += distancia_ultima_linha(estado, cor_computador)
    resultado += quantidade_pecas(estado, cor_computador)
    resultado += captura(estado, cor_computador) * n_pecas_capt

    # Nos casos a favor do adversario, decrementa..
    resultado -= quantidade_damas(estado, cor_adversario)
    resultado -= distancia_ultima_linha(estado, cor_adversario)
    resultado -= quantidade_pecas(estado, cor_adversario)
    resultado -= captura(estado, cor_adversario) * n_pecas_perdidas

    # Por fim, divide o resultado pelo numero de jogadas feitas de forma
    # que seja possível priorizar um jogo que termine no menor tempo possível..
    return resultado/numero_jogadas(estado)
```

Já para o caso da função utilidade, sabemos que devemos retornar um valor preciso, indicando apenas se o estado considerado pelo nodo folha em questão representa vitória ou derrota (empate não aparenta acontecer no jogo de damas, segundo as regras apresentadas pelo enunciado do problema). Para determinar isto, basta ver as condições no qual você ganha ou perde o jogo:

- Você ganha o jogo se o seu adversário ficar sem peças no tabuleiro;
- Você perde o jogo se você ficar sem peças para jogar.

Logo, a função utilidade deve se basear nesses aspectos para retornar o valor correspondente, portanto, positivo se representa vitória para o computador, negativo se representa vitória para o adversário:

```
def utilidade(estado):
    resultado = 0
    if numero_pecas(cor_computador) == 0:
        resultado = -1
    elif numero_pecas(cor_adversario) == 0:
        resultado = 1
    return resultado
```