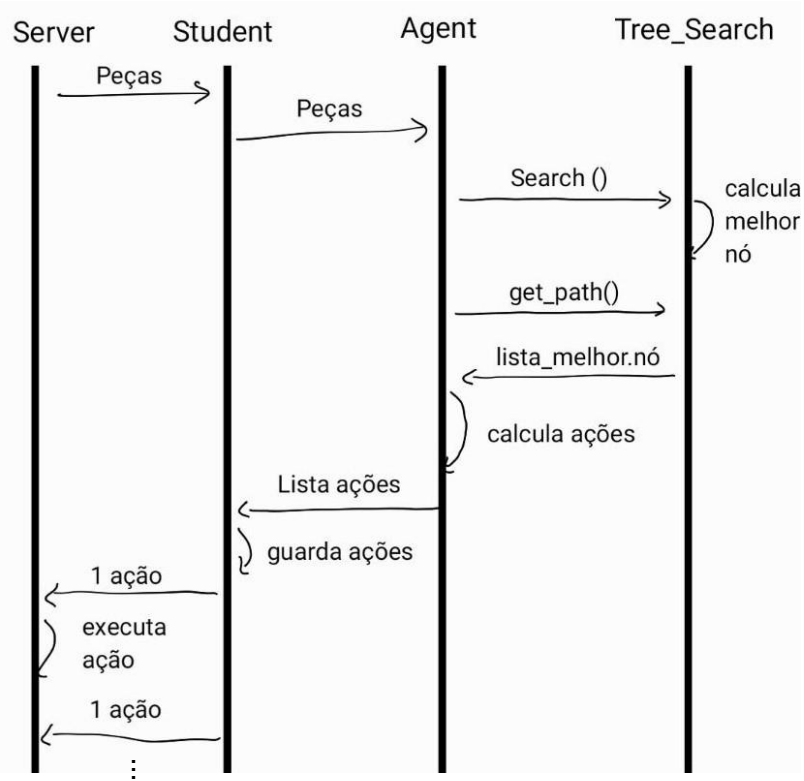
# Agente

O nosso agente (agent.py) interage com o server através do student.py. Ao receber uma peça, o student chama o agente. Este normaliza a peça (e peças seguintes se tivermos lookahead) e calcula a melhor posição e rotação para essa peça através de uma pesquisa gulosa efetuada no ficheiro tree_search.py.

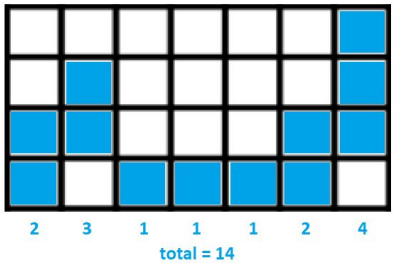
Após saber a melhor posição e rotação, o agente envia um array com as respetivas ações para que a peça se possa mover para lá (ex.: pos=-3 rot=0 significa ir 3 vezes para a esquerda e rodar 0 vezes).

O student ao receber este array, envia as ações uma a uma para o server (temos assim um movimento por frame).

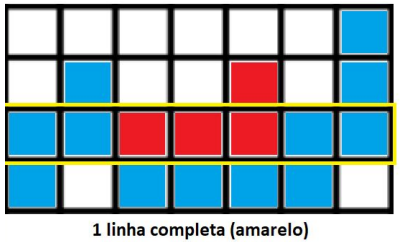


# Heurística

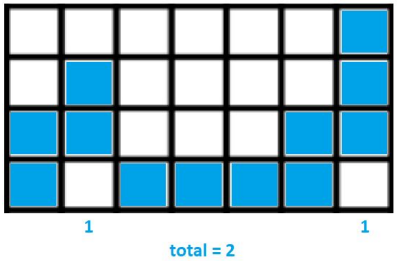
- Altura combinada (coef: -0.510066)



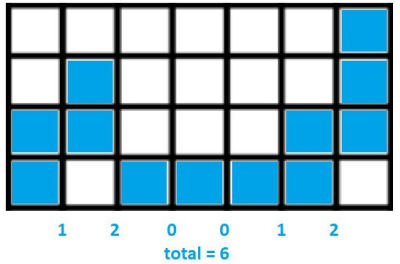

- Linhas completas (coef: 0.760666)




- Número de buracos (coef: -0.35663)




- Instabilidade (coef: -0.184483)




$$H = \sum_{i=0}^{3} parametro_i * coef_i$$

# Tree Search

**Nó** - peça em determinada posição e rotação

**Filled** - simulação do campo do jogo em um determinado momento

**Método *search()*** - calcula a heurística para diferentes nós; atualiza o filled (uma matriz binária); cria nós filhos; adiciona nós à lista de pesquisa dependendo do lookahead (não vamos expandir nós folhas).

**Método *intersect()*** - verifica se uma peça colide com outra peça ou com os limites do campo

**Método *simulate_lines()*** - simula o número de linhas feitas

**Método *calculate_heuristic()*** - calcula a heurística

**Método *get_path()*** - retorna o caminho desde o melhor nó folha até ao nó cujo parent é a raiz

# Resultados

Escolhemos o lookahead e o número de nós segundo um compromisso entre a média e o desvio padrão dos valores após 10 execuções.

| | | la=1,nodes=5 | la=1,nodes=4 | la=1,nodes=3 | la=1,nodes=2 | la=0 |
|---|---|---|---|---|---|---|
| Andreia | diferença | 138 | 192 | 326 | 452 | 260 |
| | média | 273,9 | 338,6 | 429,3 | 331,1 | 162,2 |
| | desvio | 41,02695591 | 59,74798927 | 97,99211986 | 188,8870621 | 72,18002186 |
| Miguel | diferença | 57 | 86 | 254 | 673 | 295 |
| | média | 417,7 | 475,5 | 599,5 | 645,2 | 147,2 |
| | desvio | 17,61974902 | 25,78220747 | 68,98510306 | 259,3683781 | 91,53602812 |

la: representa o lookahead
nodes: representa o número de nós expandidos para cada profundidade

# Bibliografia

https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/
https://thawsitt.me/files/Tetris_AI_CS221_final_paper.pdf
https://christophm.github.io/interpretable-ml-book/tree.html
https://en.wikipedia.org/wiki/Monte_Carlo_tree_search
https://www.geeksforgeeks.org/array-copying-in-python/
https://www.geeksforgeeks.org/python-get-indices-of-true-values-in-a-binary-list/

Conversamos com os seguintes grupos:

1) 98323 - 98546 - 97814
2) 96123 - 97606 - 100055