CSC384H1F

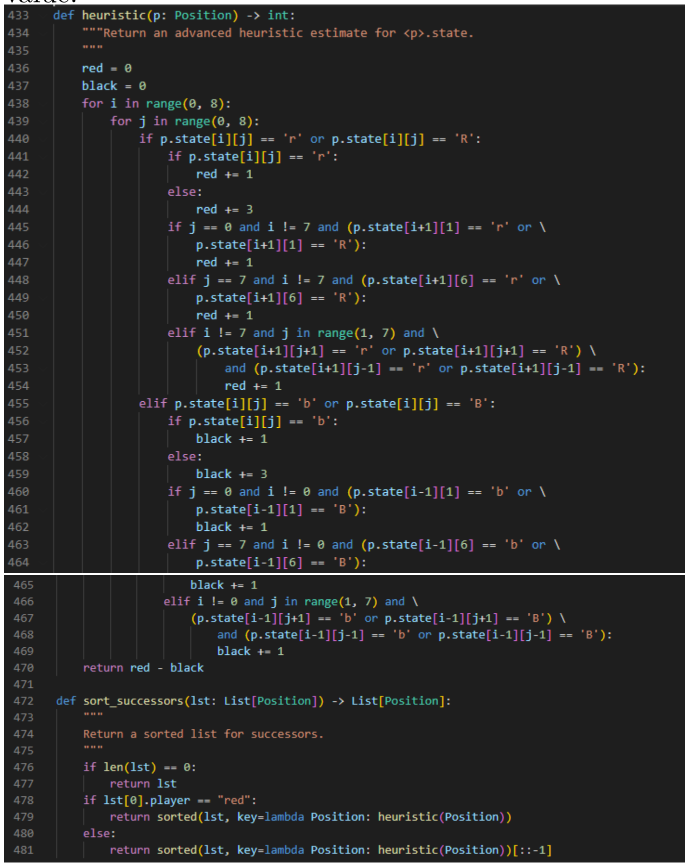
Assignment 2: Game Tree Search

Junrui Chen - chenj591

Due: October 13, 2022

Heuristic Function

Code. The following is my code for my heuristic function and how to order nodes based on the heuristic value:



```python
def heuristic(p: Position) -> int:
    """Return an advanced heuristic estimate for <p>.state.
    """
    red = 0
    black = 0
    for i in range(0, 8):
        for j in range(0, 8):
            if p.state[i][j] == 'r' or p.state[i][j] == 'R':
                if p.state[i][j] == 'r':
                    red += 1
                else:
                    red += 3
                if j == 0 and i != 7 and (p.state[i+1][1] == 'r' or \
                    p.state[i+1][1] == 'R'):
                    red += 1
                elif j == 7 and i != 7 and (p.state[i+1][6] == 'r' or \
                    p.state[i+1][6] == 'R'):
                    red += 1
                elif i != 7 and j in range(1, 7) and \
                    (p.state[i+1][j+1] == 'r' or p.state[i+1][j+1] == 'R') \
                        and (p.state[i+1][j-1] == 'r' or p.state[i+1][j-1] == 'R'):
                        red += 1
            elif p.state[i][j] == 'b' or p.state[i][j] == 'B':
                if p.state[i][j] == 'b':
                    black += 1
                else:
                    black += 3
                if j == 0 and i != 0 and (p.state[i-1][1] == 'b' or \
                    p.state[i-1][1] == 'B'):
                    black += 1
                elif j == 7 and i != 0 and (p.state[i-1][6] == 'b' or \
                    p.state[i-1][6] == 'B'):
                    black += 1
                elif i != 0 and j in range(1, 7) and \
                    (p.state[i-1][j+1] == 'b' or p.state[i-1][j+1] == 'B') \
                        and (p.state[i-1][j-1] == 'b' or p.state[i-1][j-1] == 'B'):
                        black += 1
    return red - black

def sort_successors(lst: List[Position]) -> List[Position]:
    """
    Return a sorted list for successors.
    """
    if len(lst) == 0:
        return lst
    if lst[0].player == "red":
        return sorted(lst, key=lambda Position: heuristic(Position))
    else:
        return sorted(lst, key=lambda Position: heuristic(Position))[::-1]
```

Description. This heuristic function is improved based on the utility function given in handout.
Note that the player's colour is red and the opponent's colour is black.
With the simple utility function, regular piece worth 1 and king piece worth 2, and the result of the utility function is the scores gotten by red minus the scores gotten by black.
My heuristic function also considers about the difference between regular and king pieces. The score for regular piece was set as 1, and the score for king was 3. Another thing that this heuristic function considers is whether the piece can be captured by the opponent's pieces. The article *Tips to win Checkers* says, "Try to form a pyramid shape with your pieces." Obviously, if three pieces with same colour form a pyramid shape, the one at the top of the pyramid is impossible to be captured. Notice that one of the two pieces at the bottom of the pyramid can be replaced by the

edge of the board. Thus, a bonus point is given to a piece which cannot be captured.
In other words, a regular piece which cannot be captured worth 2, a king piece which cannot be captured worth 4, the rest regular pieces worth 1 each, and the rest king pieces worth 3 each.
The result of this heuristic function is still the scores gotten by red minus the scores gotten by black.

Node Ordering Heuristic. With the heuristic function, the successors of each position can be sorted, which is a list of possible positions for a position.
If this list is empty, return an empty list.
If the positions in this list have a red player, this list contains the possible positions as the next step for a position having black player, vice versa.
If this is a list of results for red players, then these results should be sorted from largest to smallest according to heuristic value to trigger (value $\geq$ beta) as soon as possible.
If this is a list of results for black players, then these results should be sorted from smallest to largest according to heuristic value to trigger (value $\leq$ beta) as soon as possible.