CSC 311 - PROJECT 21

P15/141016/2020 - Alex Kimutai
P15/137032/2019 – Hamud Abdulrahman Abeid
P15/140000/2020 - Kitiabi Kofi
P15/137294/2019- Michael Nyandiri Omare

Q:

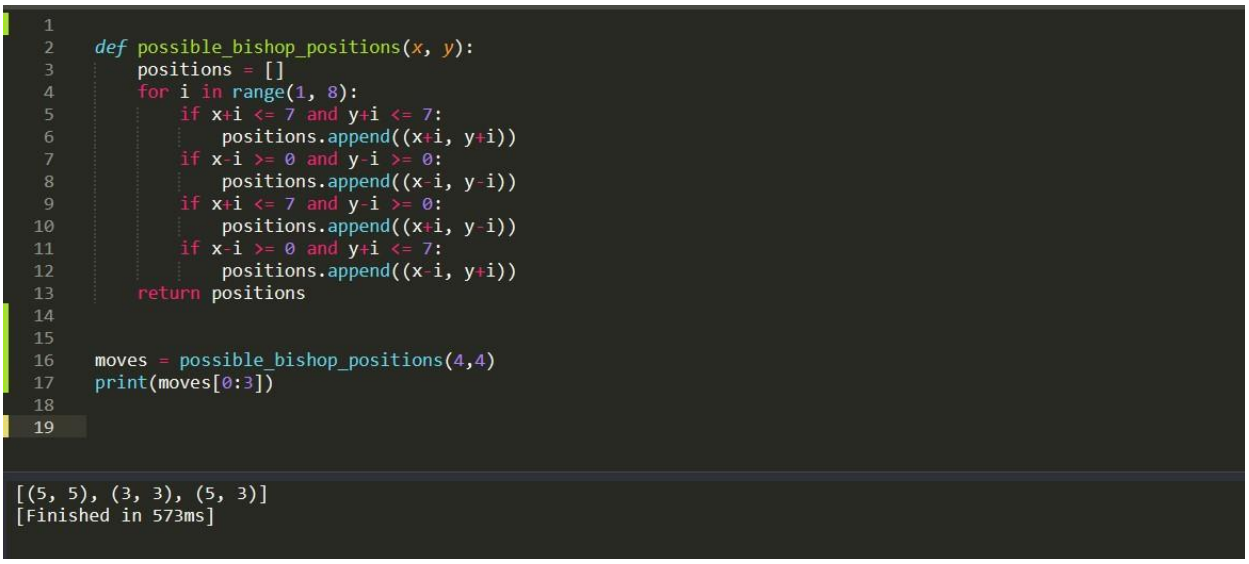
Formulate an algorithm when given a position of the bishop, it highlights three possible successive positions of a bishop on an 8x8 chess board. Find the complexity of the algorithm.

Ans:

1. Store the current position of the bishop in variables x and y (x represents the row and y represents the column).
2. Initialize a list, "positions", to store the possible positions.
3. Using a loop, iterate through a range of values starting from 1 to 8.
4. For each iteration, check if x+i and y+i are within the boundaries of the chess board (0<=x<=7, 0<=y<=7)
5. If the position is valid, append it to the positions list.
6. Repeat steps 4 and 5 for the other three diagonal positions by incrementing/decrementing both x and y by i.
7. Return the positions list

This algorithm takes in the current position of the bishop as parameters x and y (representing the row and column, respectively). It then uses if conditions to check for all possible positions that the bishop can move to. It checks the positions by adding and subtracting 1 from x and y, it also makes sure that the new positions are within the boundaries of the chess board (0 <= x <= 7 and 0 <= y <= 7) by using if conditions and appends the positions that are valid to the ‘positions’ list. Finally, it returns the list of valid positions. And since we only want three successive positions, we can print only three positions from the list of possible valid positions.

We can see in the python code below how this algorithm is implemented.



```python

def possible_bishop_positions(x, y):
    positions = []
    for i in range(1, 8):
        if x+i <= 7 and y+i <= 7:
            positions.append((x+i, y+i))
        if x-i >= 0 and y-i >= 0:
            positions.append((x-i, y-i))
        if x+i <= 7 and y-i >= 0:
            positions.append((x+i, y-i))
        if x-i >= 0 and y+i <= 7:
            positions.append((x-i, y+i))
    return positions


moves = possible_bishop_positions(4,4)
print(moves[0:3])


```

```
[(5, 5), (3, 3), (5, 3)]
[Finished in 573ms]
```

The time complexity of calculating possible moves of a bishop on an 8x8 chess board is O(n) where n is the number of possible moves of the bishop. This is because the algorithm needs to iterate through all the possible moves and check if they are valid. Since a bishop can move to a maximum of 32 squares on an 8x8 chess board, the algorithm will take a maximum of 32 iterations to calculate all the possible moves. Therefore, the time complexity is linear, O(n)

The space complexity of calculating three possible successive positions of a bishop on an 8x8 chess board is O(1) because the algorithm only uses a constant amount of memory to store temporary variables and the list used to store the positions. Since the size of the chess board is fixed and the algorithm only needs to store a constant number of variables, regardless of the size of the input or the initial position, the space complexity is considered to be O(1) or constant space complexity.